Akquisition von Integritätsbedingungen in Datenbanken

Dissertation

zur

Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Ingenieurwissenschaften
der Universität Rostock

vorgelegt von

Meike Klettke, geb Albrecht, geb. am 27. Mai 1969 in Rostock
aus Rostock

Rostock, 23. September 1998

Gutachter: Prof. Dr. Bernhard Thalheim, Technische Universität Cottbus
Prof. Dr. Andreas Heuer, Universität Rostock
Prof. Dr. Joachim Biskup, Universität Dortmund

Zum Geleit

Datenbanken haben heute einen festen Platz in jeder software-determinierten Infrastruktur. Ihre Bedeutung ist im Internet-Zeitalter noch mehr gewachsen. Datenbanken werden durch Datenbank-Management-Systeme (DBMS) verwaltet und dem Benutzer zum Stellen von Anfragen aufbereitet. Die heutige Technologie basiert im wesentlichen auf den theoretischen Vorarbeiten der 70er Jahre, die das relationale Datenbankmodell als Grundlage nutzten. Das relationale Modell zeichnet sich durch eine sehr einfache Strukturierung aus. Um auch komplexere Anwendungen darstellen zu können, wurde eine umfangreiche Theorie der Integritätsbedingungen entwickelt. Im Verlaufe weniger Jahre wurden weit über 100 verschiedene Klassen von statischen und dynamischen Integritätsbedingungen eingeführt. Damit war jedoch sogar ein in der Datenbanktechnologie vorgebildeter Benutzer sehr schnell überfordert. Man suchte Mitte der 70er Jahre nach anderen Modellierungshilfsmitteln und glaubte bis etwa Mitte der 90er Jahre mit dem Entity-Relationship-Modell und seinen Erweiterungen ein adäquates Modell zur Darstellung komplexerer Zusammenhänge gefunden zu haben. Mit seinen drei Konstrukten (Attribute, Entitäten, Beziehungen) erlaubte dieses Modell eine schnelle, einfache, intuitiv nachvollziehbare Darstellung von Strukturen und z.T. auch von semantischen Beziehungen zwischen Objekten. Verführt von der Einfachheit und dem intuitiven Verständnis der graphischen Notation vernachlässigte man sehr lange die Ausarbeitung einer Theorie dieses Modelles.

Mit der Ausarbeitung der Theorie des Entity-Relationship-Modelles (ER-Modell) wurde klar, daß unterschiedliche semantische Interpretationen nicht ohne weiteres vereinbar sind, daß auch das ER-Modell einer Normalisierungstheorie bedarf, daß aufgrund der Ausdrucksstärke der Konstrukte des Modelles eine Normalisierungstheorie um einiges komplexer ist als die für das relationale Modell und daß aufgrund der Ausdrucksstärke des Modelles gleiche Anwendungen in so unterschiedlicher Art und Weise modelliert werden können, daß nicht einmal herausgefunden werden kann, ob die entwickelten Schemata zur Darstellung der Anwendung äquivalent sind. Parallel dazu wurde ein Ausweg durch objektorientierte Datenbankmodelle gesucht, wobei allerdings wiederum eine theoretische Grundlegung vernachlässigt wurde. Damit wurde wiederum sichtbar, daß eines der ungelösten Probleme der Datenbanktheorie nicht auf diese Art gelöst werden kann.

Jede Modellierung von Struktur und Funktionalität einer Datenbankanwendung erfordert auch eine Erfassung der Semantik von Strukturen und Funktionen. Die Korrektheit einer Datenbank hängt in starkem Maße davon ab, inwieweit die Semantik in DBMS explizit oder implizit gepflegt werden kann und wird. Damit muß zur Modellierung auch die Semantik *vollständig* erfaßt werden. Integritätsbedingungen werden darüber hinaus benutzt, um mit Normalisierungsansätzen das Verhalten der Datenbank zu optimieren. Alle bekannten Algorithmen zur Erzeugung einer solchen 'Normalform' gehen jedoch von einer vollständigen Menge von Integritätsbedingungen aus. Wurde die Semantik nur unvollständig erfaßt, dann wird die Datenbank falsch strukturiert, ist nicht mehr ausreichend pflegbar und die Qualität der Daten sinkt kontinuierlich mit jeder falsch behandelten Aktion.

Integritätsbedingungen können jedoch sehr schnell komplex werden. Selbst in einfachen Anwendungen müssen sehr viele Integritätsbedingungen erfaßt werden. Sie werden meist durch abstrakte Formeln dargestellt, deren Semantik nicht einfach auf umgangssprachliche Formulierungen abgebildet werden können. Damit ist oft der Modellierer in seinen Formulierungsfähigkeiten, in seiner Übersicht über die Gesamtmenge der Bedingungen und in seinem Abstraktionsvermögen überfordert. Empirische Untersuchungen haben außerdem gezeigt, daß Integritätsbedingungen oft 'vergessen' werden, weil sie zu offensichtlich sind. Modellierungswerkzeuge ver-

sprechen oft eine einfache Abhilfe, bieten aber selten adäquate Hilfen an.

Komplexere Anwendungen wurden bereits nach einer relativ kurzen Laufzeit mit einem anderen (meist nur auch logischen oder physischen Niveau repräsentierten) Schema versehen, weil die Semantik nicht adäquat erfaßt und in Effizienzbetrachtungen einbezogen wurde. Zugleich stellen die Daten dieser Datenbanken erhebliches Kapital dar. Dadurch kann man meist nicht einfach die infragekommenden Anwendungen neu strukturieren, sobald Versäumnisse des Entwurfes bekannt werden. Es entstehen damit Datenbanksysteme, die als 'Altsysteme' weiter benutzt werden, obwohl erheblicher Überarbeitungsbedarf existiert. Dieses *Legacy-Problem* wurde in der Literatur breit diskutiert. Eine Lösung scheint sich mit Reengineering-Ansätzen aufzutun, die jedoch wiederum eine adäquate Erfassung der Semantik voraussetzen. Um ein Vielfaches komplexer wird das Legacy-Problem bei der Einbeziehung von Altdatenbeständen in Datenbank-Warenhäuser. Datenbank-Warenhäuser sind jedoch die inhaltliche Voraussetzung für eine sinnvolle Verwendung der Infrastruktur, die z.Z. im Rahmen des 'Information Highway' geschaffen wird.

Das *Problem der adäquaten Semantikerfassung* für Datenbanken ist auch heute noch nicht vollständig gelöst. In dieser Arbeit wird ein Ansatz verfolgt, der die Erfassung von formal formulierten Integritätsbedingungen erweitert um die Diskussion von mehr oder weniger abstrakten Beispielen, mit denen Integritätsbedingungen in einfacher, weniger abstrakter und leicht erfaßbarer Form dargestellt und validiert werden können. Mit Beispielen können zusätzliche Integritätsbedingungen leicht bestätigt oder auch abgelehnt werden. Da einem Modellierer nicht zugemutet werden kann, alle sinnvollen Beispiele selbst zu generieren, ist eine Werkzeugunterstützung erforderlich.

Datenbankanwendungen können selbst bezüglich der Anzahl der verwendeten Typen sehr schnell komplex werden. Anwendungen werden oft von einer Gruppe von Entwerfern entwickelt, die sich einer einheitlichen Methodik und eines vereinheitlichten Verständnisses bedienen müssen, um sich nicht in Details zu unterscheiden und damit zusätzliche Widersprüche in den Entwurf einzubringen. Komplexere Anwendungen fordern deshalb eine entsprechende Werkzeugunterstützung. Es existiert eine Vielzahl von Werkzeugen zur Erfassung der Datenbankstruktur, einige Werkzeuge erlauben auch die Erfassung der Prozesse. Nur wenige Werkzeuge erlauben eine - zumeist dann rudimentäre - Erfassung der Semantik. Die Lösungen zur Semantikakquisition wurden in das RADD-System (Rapid Application and Database Development) voll integriert. Damit ist erstmalig in integrierter Form eine Erfassung von Struktur, Funktionalität und Semantik für Datenbankanwendungen möglich.

Erfahrene Modellierer nutzen Lösungen, die im Zusammenhang mit anderen Anwendungen entstanden, um auf schnelle Art zu einer neuen Lösung zu gelangen. Ein guter Ansatz ist die *Wiederverwendung* von Datenbanken, der auch zur *Wiederverwendung vorhandener Lösungen* erweitert werden kann. Damit wird aber das Legacy-Problem zu einem *Wiederverwendungsproblem*. Dessen Hauptkomplexität wird wiederum durch die statische und dynamische Semantik diktiert.

Verdienst der Arbeit von Frau Klettke ist auch, diese miteinander verwobenen Probleme im Zusammenhang behandelt zu haben und eine integrierte Lösung für diese Probleme vorzulegen:

- Effiziente Erfassung, Validierung und Kontrolle von Integritätsbedingungen;
- Entwicklung eines adaptiven, auf Lernverfahren beruhenden heuristischen Rahmens zur Generierung von Kandidaten für geltende bzw. abzuweisende Integritätsbedingungen;

- Entwicklung eines Zugangs zur benutzerfreundlichen Gestaltung von Akquisitionsdialogen;
- Entwicklung eines Wiederverwendungszuganges;
- Nachweis der Praktikabilität der eigenen Ideen durch Integration der der Algorithmen in die intelligente Datenbankentwurfsumgebung RADD.

Cottbus, September 1998 Bernhard Thalheim

Vorwort

Beim Erstellen und Verändern von Datenbank-Schemata benötigt man Informationen über die Bedeutung der Daten in der Datenbank, diese Informationen werden formal durch Integritätsbedingungen des Datenbank-Schemas spezifiziert. Die Angabe der Integritätsbedingungen ist für viele Datenbank-Entwerfer und Datenbank-Administratoren nicht einfach. Eine Unterstützung der Benutzer bei dieser Aufgabe durch entsprechende Tools ist deshalb wünschenswert. Es wird deshalb in dieser Arbeit der Versuch unternommen, einen einfach verständlichen Zugang für diese Aufgabe zu entwickeln.

Basis für eine Unterstützung der Semantikakquisition sind relationale Datenbank-Schemata, man benötigt die Integritätsbedingungen dieser Datenbank-Schemata in formaler Form. In dieser Arbeit wird dazu ein Zugang gewählt, bei dem einem Benutzer mögliche Integritätsbedingungen vorgeschlagen und über die Gültigkeit von Beispiel-Datenbanken erfragt werden. Dieses Vorgehen eignet sich auch für Benutzer, die die Integritätsbedingungen eines Datenbank-Schemas nicht formal spezifizieren können. Es ist jedoch zu beachten, daß die Anzahl der zu untersuchenden Integritätsbedingungen und damit auch die Anzahl der zu diskutierenden Beispiele sehr groß sein kann. Es wird also versucht, durch verschiedene Methoden wie den Einsatz von verschiedenen Heuristikregeln, die Auswertung von Metainformationen und die Wiederverwendung von Informationen aus anderen Datenbanken die Menge der zu untersuchenden Integritätsbedingungen einzuschränken. Anschließend wird die Menge der zu untersuchenden Integritätsbedingungen durch eine intelligente Erfragungsreihenfolge sortiert, um auf diese Weise die Semantikakquisition effizient ausführen zu können.

Integritätsbedingungen müssen beim Erstellen und Verändern von Datenbank-Schemata angegeben werden, das in dieser Arbeit beschriebene Vorgehen unterstützt deshalb eine Teilaufgabe des Entwurfes und des Reverse-Engineerings von Datenbanken.

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftliche Mitarbeiterin an der Universität Rostock und der Technischen Univeristät Cottbus. Ich möchte mich an dieser Stelle bei allen Kollegen bedanken, mit denen ich die Arbeit diskutieren konnte, die mir Teile, die in irgendeiner Form eingeflossen sind, erklärt haben oder die einfach nur zugehört haben.

An erster Stelle möchte ich mich ganz herzlich bei Prof. Thalheim für die Betreuung dieser Dissertation bedanken. Durch zahllose Diskussionen über das Thema, die besonders in der Anfangsphase erfolgten, weckte er in mit das Verständnis und auch die Begeisterung für das Gebiet.
Ich glaube, daß die Betreuung durch Prof. Thalheim nicht nur Auswirkungen auf diese Arbeit hat, durch seine Art, Dinge zu hinterfragen, Zusammenhänge zu suchen und daraus neue Ideen abzuleiten, werde ich wahrscheinlich immer beeinflußt bleiben – und das meine ich im positivsten Sinne.
Zuletzt möchte ich mich bei ihm für das Verständnis bedanken, das er meiner starken Ortsgebundenheit und den Einschränkungen durch meine Familie entgegenbrachte, sodaß Regelungen für mein Arbeitsverhältnisses getroffen werden konnten, die mir sehr entgegenkamen.

Bei Prof. Biskup und Prof. Heuer möchte ich mich für die Übernahme der Gutachten und die hilfreichen Bemerkungen zu Vorversionen der Arbeit bedanken.

Bei Antje Düsterhöft möchte ich mich für zahllose Diskussionen bedanken.

Wolfram Clauss möchte ich für die Diskussionen zum Thema Wiederverwendung und die zahlreichen Bemerkungen, die er zu diesem Kapitel gemacht hat, bedanken.

Susanne Lange möchte ich für das Lesen der vollständigen Arbeit und die Bemerkungen aus einem anderen Blickwinkel danken, die hoffentlich dazu geführt haben, daß die Arbeit nicht nur für Leser verständlich ist, die sich sehr gut auf dem Gebiet Integritätsbedingungen in Datenbanken auskennen.

Den im Kapitel 10 beschriebenen Graphmatching-Algorithmus hätte ich ohne fremde Hilfe nicht in so kurzer Zeit verstanden. Ich möchte mich deshalb bei Uli Köthe, Prof. Röck und Van Bang Le bedanken, die mir geholfen haben, an dieser Stelle von der vagen Idee zur konkreten Lösung zu gelangen.

Weiterhin möchte ich Prof. Forbrig für Gespräche und Erläuterungen über das Reverse-Engineering von Software danken.

Bei der organisatorischen Durchführung des Dissertationsverfahrens haben mir Prof. Forbrig, Frau Prof. Schumann, Prof. Thalheim und Prof. Heuer sehr geholfen.

Bei meiner Familie und bei allen Freunden und Bekannten, die alle Höhen und Tiefen der Arbeit – ob sie wollten oder nicht – miterleben mußten, möchte ich mich an dieser Stelle ebenfalls bedanken.

Rostock, September 1998 Meike Klettke

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Die Datenmenge, die jährlich weltweit digital gespeichert wird, nimmt ständig zu. Ein großer Teil der Daten, die in Rechnersystemen gespeichert werden, ist in *Datenbanken* gespeichert, da die strukturierte Speicherung, die durch Datenbanken unterstützt wird, die Auswertung und Weiterverwendung von Daten erleichtert. Für die korrekte Speicherung von Daten in Datenbank-Systemen werden Informationen über die Bedeutung der Daten benötigt. Diese werden als *Integritätsbedingungen* der Datenbanken bezeichnet.

In dieser Arbeit werden Methoden vorgestellt, mit denen die Integritätsbedingungen der Datenbanken ermittelt werden können.

1.1 Bedeutung von Datenbanken

Der Einsatz von Datenbank-Systemen bietet Vorteile bei der Speicherung und Auswertung der Daten. Datenbank-Systeme erfordern und unterstützen eine Strukturierung der Daten, die bei der Auswertung der Daten hilfreich ist. Datenbank-Systeme ermöglichen eine effiziente physische Speicherung der Daten. In Datenbank-Systemen gibt es Funktionen, die Speicherung, Änderung und Löschen der Daten, sowie Zugriffe auf die Daten unterstützen. Dabei werden Kriterien wie Datenschutz und Datensicherheit berücksichtigt.

Für den fehlerfreien und effizienten Einsatz der Datenbanken sind der korrekte Datenbank-Entwurf beziehungsweise die korrekte Wiederherstellung von Datenbanken sind besonders bedeutsam. Im folgenden sollen die Anforderungen an diese Phasen kurz erläutert werden.

1.1.1 Bedeutung des Datenbank-Entwurfes

Der Entwurf von Datenbanken ist vor dem Abspeichern von Daten in eine Datenbank erforderlich. Dem Datenbank-Entwurf kommt *besondere Bedeutung* zu, man muß bereits vor der Arbeit mit der Datenbank optimale Datenbank-Strukturen finden, um eine spätere effektive Nutzung der Datenbanken zu gewährleisten. Ein schlechter Entwurf kann die Ursache für Probleme wie Inkonsistenz, Instabilität, Ineffektivität oder auch fehlende Robustheit eines Systems sein. Die Qualität eines Datenbank-Schemas hängt sehr stark von der Professionalität des Entwerfers und der Qualität der Unterstützung ab.

Nachträgliche Änderungen eines Datenbank-Schemas sind zeitaufwendig und kostenintensiv, da meist große Datenmengen an veränderte Strukturen angepaßt und häufig auch ergänzt oder modifiziert werden müssen. Aufgrund der dabei anfallenden Kosten wird meist von nachträglichen Änderungen abgesehen und die Datenbanken, die nicht optimal den Anforderungen gerecht werden, weiterverwendet. Der Aufwand, der bei Eingabe, Änderung und Löschen von Daten, sowie bei Anfragen auf der Datenbank entsteht, ist dann wesentlich höher, da die Transaktionen aufwendiger sind und konsistenzerhaltende Maßnahmen explizit durchgeführt werden müssen. Damit kann die Verwendbarkeit der Datenbank eingeschränkt werden. Dem Datenbank-Entwurf kommt deshalb besondere Bedeutung zu, da Fehler in dieser Phase große Auswirkungen auf die Verwendbarkeit der Datenbank haben.

Folgende allgemeine Gütekriterien von Datenbanken versucht man beim Datenbank-Entwurf zu erreichen (nach [Nav85]):

- Verständlichkeit
- Integrität und Sicherheit
- Erweiterbarkeit
- Vollständigkeit
- Korrektheit und Konsistenz
- Berücksichtigung gesetzlicher Regelungen und innerbetrieblicher Standards

In [HeS95] werden diese Kriterien in ähnlicher Form aufgezählt, dabei ist ein weiterer Punkt enthalten:

- Vermeidung überflüssiger Daten

Diese Kriterien sind größtenteils nicht formal überprüfbar.

Obwohl die Bedeutung des Datenbank-Entwurfes allgemein anerkannt ist, gibt es kaum Leitfäden und Tools zur Unterstützung des Entwurfes. Das bringt einen Entwerfer sehr oft zu einem *intuitiven, informalen Entwurf* in einem Inspirationsstil. Dabei sind nur geübte Entwerfer in der Lage, gute Datenbanken zu entwerfen. Die *Entwicklung von Tools*, die auch ungeübte Datenbank-Entwerfer bei allen Entwurfsaufgaben unterstützen, ist deshalb nach wie vor von Bedeutung.

1.1.2 Bedeutung des Reverse-Engineerings von Datenbanken

Nach [Ewa95] können aus folgenden Gründen Veränderungen einer Datenbank notwendig werden: Änderungen des Produktionsprozesses, Änderungen der Technologie, Einführung neuer Produkte, Änderungen der Unternehmensstruktur, Änderungen der staatlichen Reglementierungen, usw. Diese Ursachen können erfordern, daß eingesetzte Datenbanken geändert werden müssen. Das führt zu einer notwendigen *Wartung* bestehender Datenbanken.

Da Datenbanken oft über sehr lange Zeiträume eingesetzt werden, ist eine Wartung der Datenbanken während ihres Einsatzes in den meisten Fällen erforderlich, weil sich die Anforderungen an die Datenbank ändern können. Die verlustfreie Wartung und Weiterentwicklung von Datenbanken ist eine besonders wichtige Aufgabe, da die vorhandenen Daten in den Datenbanken große Werte darstellen und bei der Umstellung oder Änderung von Datenbanken verlustfrei übernommen werden sollen. Es gibt nach [KlG95] verschiedene Möglichkeiten, eine Wartung von Datenbanken (Anpassung an neue Anforderungen) vorzunehmen. Abbildung 1.1 und Abbildung 1.2 zeigen diese für allgemeine Softwaresysteme.

Abbildung 1.1: “quick-fix“-Wartungsmodell (nach [KlG95])

Bei der “quick-fix“-Wartung erfolgt eine Anpassung des Source-Codes an geänderte Anforderungen. Die anderen Dokumente werden (wenn überhaupt) erst nachträglich angepaßt.
Durch diese Form der Wartung kommt es zu Inkonsistenzen zwischen den verschiedenen Dokumenten. Bereits erfolgte Änderungen im Source-Code sind schwer nachvollziehbar. Diese Art der Wartung bewirkt, daß die gewünschte Anpassung des Codes an die veränderten Anforderungen schnell erreicht wird, führt aber langfristig dazu, daß die Software nicht mehr beherrschbar ist.

Bei der “iterative enhancement“-Wartung erfolgt eine schrittweise Anpassung aller Dokumente, dabei wird auf dem höchsten Abstraktionsniveau begonnen. Diese Form der Wartung ist zwar bei kleinen Änderungen kurzfristig etwas aufwendiger, sichert aber langfristig konsistente Dokumente.

Bei bereits seit langer Zeit eingesetzten Datenbanken kann es sein, daß “quick-fix“-Wartungen durchgeführt wurden. *Konzeptionelle Entwürfe* sind dann *nicht mehr aktuell*. Es ist auch möglich, daß zu einer Datenbank *keine konzeptuellen Entwürfe* bestanden haben. Der Entwurf dieser Datenbanken erfolgte im logischen Datenmodell, Änderungen und Ergänzungen wurden ebenfalls im logischen Modell durchgeführt.

Bei der Transformation bestehender Datenbanken in andere Systeme oder bei großen Änderungen in bereits existierenden Datenbanken ist es notwendig, ein *konzeptuelles Modell* für die Datenbank zu *erstellen* bzw. *wiederherzustellen*. Das bedeutet, daß ein *Reverse-Engineering* von Datenbanken durchgeführt werden muß. Datenbank-Reverse-Engineering kann als Teil des Wartungprozesses gesehen werden, bei dem durch ein ausreichendes Verständnis des existierenden Systems und dessen Anwendungsgebietes Design-Spezifikationen für eine existierende Datenbank wiederermittelt werden. Beim relationalen Datenbank-Reverse-Engineering wird ein konzeptuelles Schema für eine relationale Datenbank erstellt.

Das Reverse-Engineering von Datenbanken transformiert Source-Code in abstraktere Darstellungen, meist sollen diese weiterentwickelt und verändert werden. Die Verbindung einer Reverse-Engineering-Phase (zur abstrakteren Darstellung des Systems) und einer Forward-Engineering-Phase zur Veränderung und Erweiterung der Datenbank wird als *Re-Engineering der Datenbank* bezeichnet.

1.2 Akquisition und Verwendung von Integritätsbedingungen

Die Ermittlung von geltenden Integritätsbedingungen ist sowohl beim Entwurf als auch im Reverse-Engineering von Datenbanken erforderlich, um korrekte Datenbanken zu erstellen. In dieser Arbeit soll eine Unterstützung der Semantikakquisition gezeigt werden, die im Datenbank-Entwurf und im Reverse-Engineering von Datenbanken einsetzbar ist. Dazu erfolgt in Abschnitt 1.2.1 zunächst ein Überblick, wie die Semantik von Datenbanken dargestellt werden kann. Anschließend folgt die Einordnung der Semantikakquisition in den Datenbank-Entwurf (Abschnitt 1.2.2) und in das Reverse-Engineering von Datenbanken (Abschnitt 1.2.3). Darauf aufbauend wird in Abschnitt 1.2.4 die Semantikakquisition als Teilaufgabe des Datenbank-Entwurfes und des Reverse-Engineerings von Datenbanken verglichen. Es wurde schon mehrfach erwähnt, daß Integritätsbedingungen notwendig sind, um korrekte Datenbanken zu finden. Zum Ende des Kapitels wird dargestellt, für welche Entwurfsaufgaben Integritätsbedingungen benötigt werden.

1.2.1 Darstellungsmöglichkeiten von Semantik in Datenbanken

Es gibt verschiedene Möglichkeiten zur Darstellung der Semantik von Datenbanken:

- **allgemeine prädikatenlogische Formeln**
  Prädikatenlogische Formeln sind ausdrucksstarke Möglichkeiten, um die Semantik von Datenbanken darzustellen. Viele verschiedene Zusammenhänge sind darstellbar, in [Lip89] wurde gezeigt, wie über prädikatenlogische Formeln (erweitert um temporale Ausdrücke) dynamische Integritätsbedingungen dargestellt werden können. Der Nachteil der prädikatenlogischen Formeln ist, daß sie aufgrund der Ausdrucksstärke schwer handhabbar sind. Die Erfüllbarkeit und Widerspruchsfreiheit von prädikatenlogischen Formeln ist nicht effizient zu testen.

- **Integritätsbedingungen**
  Viele Entwurfsaufgaben benötigen nur eine spezielle eingeschränkte Menge von prädikatenlogischen Ausdrücken, die als Integritätsbedingungen bezeichnet werden. Durch die Beschränkung auf wenige Integritätsbedingungen erreicht man eine größere Handhabbarkeit.

Die Auswahl, welche Integritätsbedingungen in dieser Arbeit berücksichtigt werden, erfolgt nach der Überlegung, welche Integritätsbedingungen in welchen Datenmodellen strukturell dargestellt werden können und welche Integritätsbedingungen für welche Entwurfsaufgaben benötigt werden.

Es sollen die Integritätsbedingungen gesucht werden, die Auswirkungen auf die *strukturelle Darstellung* von Datenbanken in verschiedenen konzeptuellen Datenmodellen haben. Im Entity-Relationship Modell werden *Schlüssel* und *Kardinalitäten* strukturell dargestellt. Bei der Übersetzung von Datenbanken im Entity-Relationship Modell in relationale Datenbanken entstehen *Inklusionsabhängigkeiten* durch ergänzte Fremdschlüssel in den Relationen-Schemata.
In objektorientierten Modellen werden *Schlüssel* und *Kardinalitäten* zur strukturellen Darstellung benötigt. Is-A-Beziehungen lassen sich ebenfalls strukturell darstellen, diese bedingen Inklusionsabhängigkeiten. Über Is-A-Beziehungen können Disjunktheitsbedingungen (darstellbar

durch die Kombination von *Inklusions- und Exklusionsabhängigkeiten* $X \subseteq Z, Y \subseteq Z$ und $X \mid\mid Y$) und Überdeckungsbedingungen ($X \subseteq Z, Y \subseteq Z$ und $X \cup Y = Z$) gelten.

Daraus resultiert die Auswahl der Integritätsbedingungen (Schlüssel, Inklusions- und Exklusionsabhängigkeiten und Kardinalitäten), für die in dieser Arbeit eine Methode zur Unterstützung der Akquisition gezeigt werden soll. Überdeckungsbedingungen können, sofern sie nicht durch die bekannten Daten ausgeschlossen werden, durch Nachfragen nach der Ermittlung von Inklusionsabhängigkeiten gefunden werden.

Die für *Normalisierungsoperationen* auf relationalen Datenbanken notwendigen Integritätsbedingungen sollen ebenfalls einbezogen werden. Bei der Normalisierung werden *Schlüssel, funktionale Abhängigkeiten* und *mehrwertige Abhängigkeiten* ausgewertet. Mehrwertige Abhängigkeiten sind schwer verständlich und dadurch schlecht anhand von Beispieldaten darstellbar und diskutierbar, deshalb kann man die Erfragung dieser Integritätsbedingungen in einem nicht formalen Ansatz kaum unterstützen. Aufgrund der Normalisierung von Datenbanken wird also in diesem Ansatz ebenfalls die Akquisition von Schlüsseln und funktionalen Abhängigkeiten vorgestellt.

Aufgrund der Normalisierung und der Darstellbarkeit von Semantik im Entity-Relationship Modell und in objektorientierten Datenmodellen wird in dieser Arbeit gezeigt, wie die informale Akquisition von Schlüsseln, funktionalen Abhängigkeiten, Inklusions- und Exklusionsabhängigkeiten sowie Kardinalitäten unterstützt werden kann.

Die Akquisition dieser Integritätsbedingungen wird im Datenbank-Entwurf und im Reverse-Engineering von Datenbanken eingesetzt.

1.2.2 Integritätsbedingungen im Datenbank-Entwurf

Eine der schwierigsten Teilaufgabe des Datenbank-Entwurfes ist die formale Spezifikation der Semantik der Datenbank, um die Integrität der Datenbanken zu überprüfen und zu sichern.

Der Datenbank-Entwurf erfolgt in der Regel in einem *konzeptuellen Modell* (z.B. im Entity-Relationship Modell), dieses wird in ein logisches Datenmodell (z.B. in das relationale Modell) übersetzt. Durch die Übersetzung erhält man strukturelle Informationen der relationalen Datenbank sowie einige explizite Integritätsbedingungen, die im konzeptuellen Modell strukturell dargestellt wurden.

[scale=0.25]wart$_i t$

Abbildung 1.2: “iterative enhancement“-Wartungsmodell (nach [KlG95])

Die *Semantikakquisition* soll auf dem *relationalen Datenmodell* erfolgen. Grund dafür ist die Tatsache, daß konkrete Beispieldatenbanken für Entwerfer leichter zu verstehen sind als abstrakte Integritätsbedingungen [Man96b]. Deshalb soll die Semantikakquisition anhand der Diskussion von konkreten Daten erfolgen. Das relationale Datenmodell wird in der Regel gut verstanden und ist für die Diskussion von Beispielen übersichtlich, dadurch ist es für die hier vorgestellte Semantikakquisition geeignet. Die Arbeit auf dem relationalen Datenmodell stellt keine Einschränkung dar, da für die meisten konzeptuellen Datenmodelle eine Übersetzung ins relationale Datenmodell existiert.

Für die Semantikakquisition durch eine Beispieldiskussion werden Testdaten im Datenbank-Entwurf benötigt. Testdaten können für viele Entwurfsaufgaben angewendet werden, deshalb sind sie in den meisten realen Datenbank-Entwürfen vorhanden. Es ist also realistisch, die Existenz von Daten beim Datenbank-Entwurf zu erwarten und diese auszuwerten.

Schwerpunkt dieser Arbeit ist die Ermittlung von Integritätsbedingungen für relationale Datenbanken im *Datenbank-Entwurf*.

1.2.3 Akquisition von Integritätsbedingungen im Reverse-Engineering von Datenbanken

Die Identifikation von Beziehungen zwischen den Datenbank-Komponenten ist eine notwendige Teilaufgabe des Reverse-Engineering von Datenbanken. In dieser Arbeit soll eine mögliche Lösung dieser Aufgabe erläutert werden. Es wird dazu die Semantik der bestehenden Datenbank im logischen Datenmodell ermittelt (geltende Integritätsbedingungen der Datenbanken), diese Informationen sind Voraussetzung für das korrekte Ausführen des Datenbank-Reverse-Engineerings.

In bestehenden und teilweise seit langer Zeit eingesetzten Datenbanken sind oft Integritätsbedingungen nicht oder nur unvollständig bekannt. Man kann auch nicht davon ausgehen, daß die Datenbanken bestimmten Restriktionen entsprechen, z.B. daß sich alle Relationen in einer bestimmten Normalform befinden. Trotzdem sind die Daten in diesen Datenbanken in der Regel korrekt, die notwendigen konsistenzerhaltenden Maßnahmen wurden über Anwendungsprogramme gesteuert. Im Zuge der Umstellung solcher bestehender Datenbanken auf andere (semantisch ausdrucksstärkere) Datenmodelle gehen diese Informationen aus den Anwendungsprogrammen verloren, und es kann dadurch zu falschen oder ineffizienten Datenbank-Schemata kommen. Bekannte Methoden des Reverse-Engineering gehen jedoch davon aus, daß entweder alle Integritätsbedingungen bekannt sein müssen [MaM90] oder aber bestimmte Namenskonventionen eingehalten werden müssen und die Relationen sich mindestens in 2.Normalform befinden [BCN92]. Die bekannten Reverse-Engineering Ansätze für Datenbanken benötigen also Voraussetzungen, die nicht in jedem Fall gegeben sind.

Um eine korrekte Darstellung der Datenbank im konzeptuellen Modell zu erreichen, muß im Reverse-Engineering an erster Stelle die Ermittlung der geltenden Integritätsbedingungen erfolgen.
Zu Beginn des Reverse-Engineering erfolgt deshalb ein *Analyseprozeß*, bei dem die *Semantik der Datenbanken* ermittelt werden muß. Auf diese Weise kann man versuchen, die Bedeutung der Daten in den Datenbanken festzustellen. In [HTJ93] wird auf die Bedeutung dieser Phase hingewiesen.

Die Semantikakquisition im Reverse-Engineering kann ähnlich wie die Semantikakquisition im Datenbank-Entwurf erfolgen. Sie wird ebenfalls auf einem logischen Datenmodell durchgeführt, um die vorhandenen Daten auswerten zu können. Erst nach der Analyse der Datenbanken und dem Finden der Integritätsbedingungen dieser Datenbanken ist die Transformation in ein konzeptuelles Modell möglich. Abbildung 1.4 zeigt die Einordnung der Semantikakquisition in das Reverse-Engineering von Datenbanken.

Die Semantikakquisition für das Reverse-Engineering wird in dieser Arbeit für *relationale Datenbanken* erläutert.

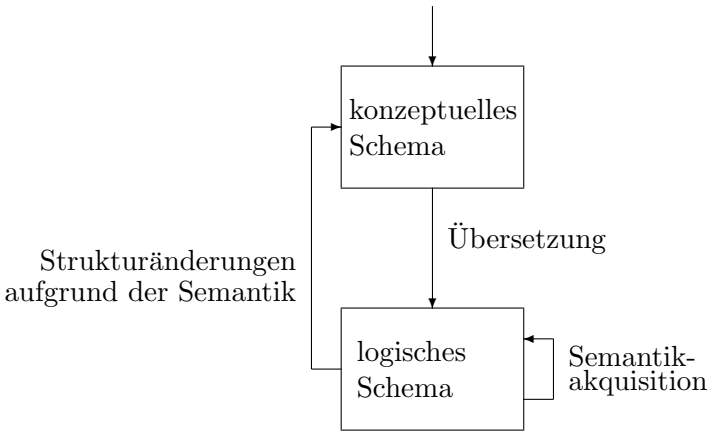


Abbildung 1.3: Semantikakquisition im Datenbank-Entwurf

1.2.4 Vergleich der Semantikakquisition im Entwurf und Reverse-Engineering von Datenbanken

In den beiden letzten Abschnitten erfolgte eine kurze Erläuterung, wie sich die Semantikakquisition in den Datenbank-Entwurf und in das Reverse-Engineering von Datenbanken eingliedert. In diesem Abschnitt wird die Semantikakquisition für diese beiden Anwendungsgebiete miteinander verglichen, Gemeinsamkeiten und Unterschiede werden angegeben.

	Entwurf	Reverse-Engineering
zugrunde gelegtes Datenmodell	logisches (relationales) Datenmodell	logisches (relationales) Datenmodell

Sowohl beim Datenbank-Entwurf als auch beim Reverse-Engineering von Datenbanken erfolgt die Semantikakquisition auf dem logischen Datenmodell. Es ist also eine in vielen Punkten identische Unterstützung der Semantikakquisition denkbar.

	Entwurf	Reverse-Engineering
Existenz von Daten	ja - Testdaten (geringer Umfang)	ja - umfangreiche Daten

Für viele Aufgaben im Datenbank-Entwurf werden Testdaten benötigt. Diese Testdaten helfen, die Qualität und Verwendbarkeit der entworfenen Datenbanken frühzeitig zu testen. Man kann also davon ausgehen, daß zu einer entworfenen Datenbank auch Testdaten zur Verfügung stehen. In der Regel enthalten die Testdaten nur wenige Tupel, sie sind nur ein sehr kleiner Ausschnitt realer Datenbanken. Diese Testdaten lassen sich bei der Semantikakquisition auswerten.
Beim Reverse-Engineering von Datenbanken kennt man meist große Datenmengen oder sogar Folgen von Datenbank-Instanzen. Aus diesen sind viele Informationen ableitbar, die bei der Semantikakquisition verwendet werden können. Dabei sind aufgrund der großen Datenmengen effiziente Algorithmen zur Ableitung von Integritätsbedingungen aus Daten erforderlich.

	Entwurf	Reverse-Engineering
Existenz expliziter Integritätsbedingungen	- aus dem konzeptuellen Datenmodell - von Entwerfer angegeben	- explizit bekannt - vom Datenbank-Administrator angegeben

Beim Entwurf von Datenbanken im konzeptuellen Modell und der Übersetzung in ein logisches Datenmodell sind einige explizite Informationen zur Semantik bekannt, die im konzeptuellen Modell strukturell dargestellt werden konnten, im logischen Datenmodell jedoch nicht implizit darstellbar sind. Diese sind explizite Integritätsbedingungen des logischen Datenmodells. Beim Reverse-Engineering von Datenbanken können in den vorliegenden Datenbanken einige Informationen zur Semantik in Form von expliziten Integritätsbedingungen bereits bekannt sein. Außerdem kann es sowohl im Datenbank-Entwurf als auch beim Reverse-Engineering von Datenbanken Entwerfer bzw. Datenbank-Administratoren geben, die in der Lage sind, einige Integritätsbedingungen formal zu spezifizieren.

	Entwurf	Reverse-Engineering
Gewährleistung bestimmter Normalformen	nein	nein

Im Datenbank-Entwurf können nach der Übersetzung einer Datenbank in ein logisches Datenmodell keine Restriktionen gewährleistet werden, man kann z.B. nicht davon ausgehen, daß bestimmte Normalformen in einer relationalen Datenbank gelten.
Im Reverse-Engineering sind Datenbanken im logischen Datenmodell bekannt. Es können auch hier keine Restriktionen gewährleistet werden, man kann ebenfalls nicht davon ausgehen, daß bestimmte Normalformen in den relationalen Datenbanken erfüllt sind. Gerade durch über lange Zeiträume durchgeführte manuelle Erweiterungen und Veränderungen kann man keine Aussagen über die Beschaffenheit der Datenbanken treffen.

	Entwurf	Reverse-Engineering
Existenz von Transaktionen	teilweise vorhanden	meist vorhanden

Es ist im Datenbank-Entwurf möglich, daß Beispieltransaktionen spezifiziert wurden, diese kann man auswerten, um Informationen über geltende Integritätsbedingungen abzuleiten.
Im Reverse-Engineering sind Transaktionen oder Anfragen in irgendeiner Form spezifiziert worden, um mit den Datenbanken arbeiten zu können. Auch aus diesen lassen sich Informationen über die Semantik der Datenbanken ableiten.

	Entwurf	Reverse-Engineering
Anzahl der unbekannten Integritätsbedingungen	in der Regel sehr groß	in Abhängigkeit von den verfügbaren Daten geringer

Da beim Datenbank-Entwurf aus den Testdaten in der Regeln nur wenige Integritätsbedingungen ableitbar sind, existieren sehr viele unbekannte Integritätsbedingungen. Es ist deshalb notwendig, effiziente Vorgehensweisen zur Diskussion der unbekannten Integritätsbedingungen

mit dem Benutzer zu finden.
Beim Reverse-Engineering sind in der Regel weniger unbekannte Integritätsbedingungen vorhanden, da durch die Daten bereits mehr Informationen zur Semantik abgeleitet werden konnten.

	Entwurf	Reverse-Engineering
iterativer Prozeß	ja	nein

Der Datenbank-Entwurf ist ein iterativer Prozeß, in dessen Verlauf die konzeptuelle Datenbank und damit auch die logische Datenbank verändert oder erweitert werden kann. Dabei sollen bereits ermittelte Informationen über die Integritätsbedingungen nach Änderungen soweit möglich übernommen werden. Verfahren dazu müssen bekannt sein.
Beim Reverse-Engineering ist dieses nicht erforderlich, hier ändert sich die logische Datenbank nicht. Es schließt sich an das Reverse-Engineering in der Regel jedoch ein Forward-Engineering-Schritt an, um die analysierte Datenbank zu verändern und an aktuelle Anforderungen anzupassen. Diese Veränderung der Datenbank ist dann ebenfalls ein iterativer Prozeß.

Man sieht durch diese Übersicht, daß die Semantikakquisition im Datenbank-Entwurf und im Reverse-Engineering von Datenbanken sehr viele Gemeinsamkeiten aufweist. Es werden deshalb in dieser Arbeit Methoden zur Semantikakqusition vorgestellt, die in beiden Fällen anwendbar sind. Auf Besonderheiten, die für eines der Anwendungsgebiete auftreten, wird dabei an den gegebenen Stellen hingewiesen.

1.2.5 Verwendung der Integritätsbedingungen

In den vorherigen Abschnitten wurde die Akquisition von Integritätsbedingungen kurz erläutert, in diesem Abschnitt soll dargestellt werden, wie die Integritätsbedingungen für die verschiedenen Entwurfsaufgaben verwendet werden. Einige Anwendungen der Integritätsbedingungen wurden schon aufgezählt, um die Auswahl der Integritätsbedingungen, die dieser Arbeit zugrunde gelegt werden, zu motivieren. In diesem Abschnitt erfolgt eine Vervollständigung der Anwendungen von Integritätsbedingungen in Datenbanken.

Strukturelle Darstellung semantischer Informationen. In konzeptuellen Datenmodellen gibt es Konstrukte, denen eine bestimmte Semantik unterlegt wird. Bildet man Zusammenhänge auf diese konzeptuellen Modelle ab, die diese Semantik nicht haben, so kommt es zu Fehlern bei der Arbeit mit den Datenbanken in diesen Modellen. Die vollständigen und richtigen Informationen über die Semantik der Datenbanken sind also erforderlich, um Datenbanken richtig auf die Datenmodelle abbilden zu können bzw. um zu überprüfen, ob die Abbildung der Datenbanken in den Datenmodellen richtig ist.

Normalisierung. Normalformen spielen in relationalen Datenbanken eine wichtige Rolle, sie werden deshalb in vielen Veröffentlichungen ausführlich behandelt, z.B. in [Cod70], [Lie85], [Heu86], [Sti91], [VRT82], [PBG89], [HeS95]. Normalformen wurden eingeführt, um die Eigenschaften Redundanzfreiheit der Datenbanken und damit Konsistenz der Datenbanken zu sichern.

Normalformen können keinen guten Datenbank-Entwurf garantieren, sie können aber sicherstellen, daß keine Anomalien (Insert-, Update- und Deleteanomalie) in den Datenbanken auftreten, d.h. daß der Zustand der Datenbanken auch nach einer Transaktion konsistent ist.

Eine Übersicht über verschiedene Normalformen und Algorithmen, die überprüfen, ob eine bestimmte Normalform in einem Relationen-Schemata erfüllt ist und über Algorithmen zum Erreichen von Normalformen ist in [Heu86] gegeben. Für diese Algorithmen benötigt man vollständige Informationen über die geltenden Integritätsbedingungen.

Optimierung. Auf Datenbanken sollen effiziente Anfragen möglich sein. Eine Anforderung an den Datenbank-Entwurf ist deshalb das Entwerfen von Datenbank-Strukturen, auf denen bestimmte Transaktionen effizient ausgeführt werden können.

Im vorherigen Abschnitt wurden Normalformen von Datenbanken erläutert. Eine Normalisierung von Datenbanken bewirkt immer eine Dekomposition, damit Redundanzfreiheit gewährleistet werden kann. Diese Dekomposition kann bewirken, daß Anfragen auf einer Datenbank über mehreren Relationen als vor der Normalisierung ausgeführt werden müssen. Die dabei notwendigen Join-Operationen sind jedoch sehr kostenintensiv. Die Kriterien *Redundanzfreiheit* und *Effizienz* können sich also widersprechen.

Diese Aussage wird in [Su85] belegt. Dort werden Kostenfunktionen zur Bewertung von Anfragen auf verschiedenen Ebenen gezeigt. Anhand von Beispielen wird gezeigt, daß eine Datenbank, die nicht in 3. Normalform ist, effizienter handhabbar ist als eine Datenbank, die sich in 3. Normalform befindet.

In [Ste96] wird eine Methode vorgestellt, mit der kritische Datenbankteile durch Abschätzung der *Komplexität der Operationen* ermittelt werden. Aufgrund der Spezifikation der Operationen und der Komplexität der Elementaroperationen sowie der zugehörigen Hilfsoperationen errechnet sich die Komplexität der Operationen. Für diese kritischen Teile können *Vorschläge zur Veränderung* der Datenbank generiert werden, so daß die Datenbank den Anforderungen durch die Operationen besser gerecht wird. Dafür werden konsistenzerhaltende Umformungen durchgeführt. Es kann bestimmt werden, welches Komplexitätsmaß die Datenbanken, die durch die vorgeschlagenen Veränderungen entstehen würden, haben. Aufgrund dessen wird eine mit Prioritäten versehene Liste von Vorschlägen zur Optimierung erstellt. Die kritischen Teile einer Datenbank und die Vorschläge zur Veränderung werden mit dem Entwerfer diskutiert. Aufgrund erwünschter höherer Effizienz der Anfragen möchte man die Datenbanken nicht immer in redundanzfreier Form (in bestimmten Normalformen) haben, teilweise ist es sinnvoll, die Effizienz der Zugriffe durch kontrollierte Redundanzen in den Datenbanken zu erhöhen. Diese beiden Eigenschaften (weitgehende Redundanzfreiheit und Effizienz) werden gegeneinander abgewogen, um Datenbank-Schemata zu finden, die den Anforderungen bestmöglich entsprechen. Man benötigt ebenso wie bei der Normalisierung vollständige und richtige Informationen über die Integritätsbedingungen, um Datenbanken zu finden, die den Kriterien Redundanzfreiheit und Effizienz weitgehend entsprechen. Es ist erforderlich, die Redundanzen, sofern sie aus Effizienzgründen erwünscht sind, bei Ausführung der Transaktionen zu pflegen. Auch dazu sind vollständige Informationen über geltende Integritätsbedingungen erforderlich.

Integritätssicherung von Datenbanken bei Transaktionen. Die Konsistenz von Datenbanken nach Ausführung von Transaktionen muß gesichert werden, dazu werden Integritätsbedingungen spezifiziert. Wie das erfolgt, wird u.a. in [HeS95] erläutert. Dabei können neben den

formalen Integritätsbedingungen weitere Integritätsbedingungen überprüft werden, die Berücksichtigung der Integritätsbedingungen (Schlüssel, funktionale Abhängigkeiten, Inklusions- und Exklusionsabhängigkeiten und Kardinalitäten) ist jedoch unbedingt erforderlich.
Voraussetzung zur Integritätssicherung von Datenbanken nach Transaktionen sind also ebenfalls vollständige Informationen über die Integritätsbedingungen der Datenbank.

Physische Speicherung von Datenbanken. Bei der physischen Speicherung von Datenbanken werden für Schlüssel Indexstrukturen angelegt. Bei abgeleiteten Attributen kann man anstelle der Attribute eine Berechnungsvorschrift, sofern diese effizient ausführbar ist, abspeichern. Kardinalitäten können verwendet werden, um festzulegen, welche Joins für eine Anfrage durchgeführt werden. Wenn mehrere Wege, über die ein Zugriff ausgeführt werden kann, in einer Datenbank möglich sind, kann aufgrund der bekannten Kardinalitäten festgelegt werden, über welchen Pfad die Anfrage erfolgt. Man benötigt also auch für eine effiziente physische Speicherung die Informationen über die Integritätsbedingungen. Fehlende Integritätsbedingungen führen zu falschen logischen Datenbanken, diese bewirken natürlich auch Fehler in den physischen Datenbanken.

View-Integration/Design-by-Units/Wiederverwendung von Datenbanken. Bei den Entwurfsaufgaben View-Integration, dem Entwurf nach der Strategie Design-by-Units ([Tha93], [Tha98]) und bei der Wiederverwendung von bekannten Datenbanken tritt jeweils das gleiche Problem auf. Es muß festgestellt werden, welche Teile einer Datenbank oder welche Teile verschiedener Datenbanken identisch oder ähnlich sind, um über diesen gleichen Teilen die Views, Units oder Teildatenbanken zu vereinigen oder wiederverwendbare Einheiten zu finden.
Zur Suche nach ähnlichen Datenbankteilen können neben den strukturellen Angaben (Bezeichnungen, Wertebereiche) auch die Integritätsbedingungen herangezogen werden, damit genauere Informationen über identische Teile ermittelt werden können.

In [BiC86] wurde eine Methode zur formalen View-Integration gezeigt, die auf Inklusionsabhängigkeiten und Exklusionsabhängigkeiten, die zwischen den einzelnen Views gelten, basieren. Diese werden als Integrations-Constraints bezeichnet. Die vollständigen Informationen über geltende Inklusions- und Exklusionsabhängigkeiten sind Voraussetzung für diesen Zugang zur View-Integration. Integritätsbedingungen sind also auch für die Entwurfsaufgaben View-Integration, Design-by-Units und Wiederverwendung von Datenbanken erforderlich.

Die Verwendung von Integritätsbedingungen zeigt die Bedeutung der vollständigen und richtigen Angaben, da alle diese Aufgaben sonst nicht fehlerfrei durchgeführt werden können. Durch unvollständige oder falsche Angaben über die Integritätsbedingungen der Datenbanken können also falsche oder ineffiziente Datenbanken entstehen.

1.3 Unterstützung der Semantikakquisition

In einer Studie von Küng ([Kün94]) wurde eine empirische Befragung von über 100 Anwendern durchgeführt. Dabei wurde nach der Einschätzung der Fähigkeiten professioneller Endnutzer im konzeptionellen Datenbank-Entwurf gefragt. $\frac{2}{3}$ der befragten Anwender gaben an, daß sie die Fähigkeiten der professionellen Endnutzer als mittelmäßig bis schlecht einstufen, nur $\frac{1}{3}$ gab an,

daß diese Kenntnisse in der Regel gut sind. Auch dieses Ergebnis macht deutlich, daß Endnutzer im Datenbank-Entwurf in geeigneter Weise unterstützt werden müssen.

Nach [Man96b] gibt es zwei Extrema von Entwerfern: Es gibt Entwerfer, die nur mit der Unterstützung von Papier und Bleistift exzellente Datenbanken entwerfen. Anderen Entwerfern hilft auch die beste Unterstützung nicht, gute Datenbanken zu entwerfen. In diesen beiden Extremfällen ist eine Unterstützung des Datenbank-Entwurfes entweder nicht erforderlich oder nicht erfolgreich. Die meisten Entwerfer ordnen sich jedoch nicht in diese Extrema ein, sondern auf einer breiten Skala dazwischen.
Auch gute Entwerfer machen gelegentlich Entwurfsfehler oder vergessen einige Zusammenhänge. In diesem Fall kann ein Tool helfen, diese Fehler oder Unvollständigkeiten zu lokalisieren und zusammen mit dem Entwerfer zu beseitigen. Ungeübte Entwerfer können mit Hilfe guter Tools, die sinnvolle Vorschläge für Erweiterungen und Verbesserungen einer Datenbank machen und ausführlich Entwurfsentscheidungen erklären und erläutern, verwendbare Datenbanken erstellen.
In allen Fällen benötigt man dazu jedoch *qualitativ hochwertige und intelligente Tools*, damit diese den Benutzer in geeigneter Weise unterstützen können und auch von dem Benutzer als Unterstützung akzeptiert werden.

In dieser Arbeit soll kein Tool zu Unterstützung des vollständigen Datenbankentwurfes vorgestellt werden, es sollen Zugänge für ein Tool vorgeschlagen werden, das eine Teilaufgabe des Datenbank-Entwurfes, die *Semantikakquisition*, unterstützt. Die Zugänge können sowohl im Datenbank-Entwurf als auch im Reverse-Engineering von Datenbanken eingesetzt werden.

Bereits 1981 wurde in [SiM81] die folgende Aussage getroffen. *" One of the problems plaguing a data base designer is the inherent difficulty of extracting from a user the complete semantics of the relations utilized to define a conceptual model of a data base."* Dieses Problem ist nach wie vor nicht zufriedenstellend gelöst worden. Möglichkeiten für eine (wenigstens teilweise erfolgreiche) Unterstützung der Benutzer bei dieser Aufgabe sollen in dieser Arbeit beschrieben werden.

1.3.1 Probleme der Spezifikation von Integritätsbedingungen

Die Schwierigkeit der Angabe von Integritätsbedingungen ist auf folgende Probleme zurückzuführen:

- **Formale Angabe von Integritätsbedingungen.** Der traditionelle Zugang zur Erfassung der Semantik basiert auf einer *abstrakten mathematischen Darstellung von Integritätsbedingungen*. Dabei wird vorausgesetzt, daß ein Benutzer in hohem Maße von seiner gewohnten Darstellungsweise abstrahieren kann. Diese Forderung ist nicht immer erfüllbar. Sogar bei kleinen Schemata ist der Abstraktionsgrad von Integritätsbedingungen so hoch, daß auch erfahrene Datenbank-Entwerfer überfordert werden. In der Regel kennt der Benutzer zwar die Bedeutung seiner Daten und kann Zusammenhänge zwischen den Daten beschreiben, kann sie aber nicht formal durch Integritätsbedingungen darstellen.

- **Vollständigkeit von Integritätsbedingungen.** Fast alle Entwurfsmethoden erwarten vom Entwerfer eine *vollständige Angabe der Semantik*. Oft werden aber *einige Integritätsbedingungen vergessen*, obwohl oder auch gerade weil sie *offensichtlich* sind. Es ist auch

möglich, daß Integritätsbedingungen zu *kompliziert* sind, besonders wenn sie über mehreren Attributen definiert sind, diese werden vom Benutzer entweder gar nicht gesehen oder falsch angegeben.

Wünschenswert wäre also eine Unterstützung bei der Semantikakquisition, bei der dem Benutzer sinnvolle Vorschläge für geltende Integritätsbedingungen unterbreitet werden.

1.3.2 Probleme bei der Unterstützung der Semantikspezifikation

Bei der Unterstützung der Semantikakquisition durch Tools möchte man dem Benutzer Vorschläge für geltende Integritätsbedingungen unterbreiten. Man möchte dadurch die in 1.3.2 angegebenen Probleme lösen. Bei einem solchen Vorgehen treten jedoch weitere Probleme auf.

- **Anzahl der zu untersuchenden Integritätsbedingungen.** Es müssen potentiell *exponentiell viele* Integritätsbedingungen untersucht werden, um die vollständige Menge von Integritätsbedingungen zu finden. Die Anzahl möglicher *funktionaler Abhängigkeiten* einer Relation beträgt $2^n * 2^n$, wobei n die Attributanzahl der Relation ist. Für die Suche nach funktionalen Abhängigkeiten kann man sich auf die rechtsminimalen Abhängigkeiten beschränken, die Anzahl dieser beträgt $2^n * n$. Bei einige dieser Abhängigkeiten ist das Attribut der rechten Seite auch auf der linken Seite enthalten, diese funktionalen Abhängigkeiten sind trivialerweise erfüllt, müssen also nicht überprüft werden. Die Anzahl der rechtsminimalen nichttrivialen Abhängigkeiten liegt jedoch auch in der Ordnung $O(2^n)$.
  Die Anzahl möglicher *Schlüssel* einer Relation liegt ebenfalls in der Ordnung $O(2^n)$.
  Auch die Anzahl der konstruierbaren Integritätsbedingungen einer Datenbank ist sehr groß. Es können bereits m^2 *unäre Inklusions- und Exklusionsabhängigkeiten* auftreten, wobei m die Attributanzhal einer Datenbank ist.

- **Komplizierte Integritätsbedingungen.** Einige Integritätsbedingungen, insbesondere solche, die über vielen Attributen definiert sind, sind für viele Benutzer schwer zu verstehen und dadurch auch schwer anzugeben. Es fällt vielen Benutzern auch schwer, solche vorgeschlagenen Integritätsbedingungen zu bestätigen oder abzulehnen.

- **Verschiedene Definition und Interpretation von Integritätsbedingungen.** Integritätsbedingungen (z.B. Kardinalitäten) können in unterschiedlicher Weise definiert sein. Beim Zusammenarbeiten verschiedener Entwerfer muß jedoch eine einheitliche Definition der Integritätsbedingungen zugrunde gelegt werden, um Fehler zu vermeiden. Integritätsbedingungen können auch mißverstanden und dadurch von verschiedenen Entwerfern auf verschiedene Weise interpretiert werden. Auch dieses Problem ist besonders relevant, wenn verschiedene Entwerfer gemeinsam an einer Datenbank arbeiten.
  Es ist aber auch möglich, daß ein Entwerfer im Laufe eines Entwurfes seine Interpretation der Integritätsbedingungen ändert und dadurch falsche Angaben zustande kommen.

Man benötigt also ein Tool zur Unterstützung der Semantikakquisition, das die in 1.3.1 und 1.3.2 angegebenen Probleme berücksichtigt.

1.3.3 Anforderungen an ein Tool zur Semantikakquisition

Aufgrund dieser Probleme bei der Unterstützung der Semantikakquisition muß ein Ansatz gewählt werden, der folgende Forderungen erfüllt:

- Ein **informaler Zugang** muß gewählt werden, um die *Richtigkeit von Integritätsbedingungen* bei ungeübten Benutzern zu gewährleisten. Durch einen informalen Zugang wird auch erreicht, daß Integritätsbedingungen von verschiedenen Benutzern immer *auf die gleiche Weise interpretiert* werden, da die Integritätsbedingungen immer erklärt, umschrieben und dargestellt werden. Dadurch werden auch *komplizierte Zusammenhänge* verständlich erfragt.
- Eine **starke Einschränkung des Suchraumes** und eine *Festlegung einer effizienten Suchreihenfolge* muß durch einen *intelligenten Zugang*, der Heuristiken zur Einschränkung und Beschleunigung der Suche verwendet, erreicht werden. Auf diese Weise wird die Anzahl der zu untersuchenden Integritätsbedingungen eingeschränkt.

Ein Ansatz, der diese Möglichkeiten nutzt, wird in dieser Arbeit ausführlich erläutert.

1.4 Bekannte Methoden zur Semantikakquisition — Literaturüberblick

Ziel dieser Arbeit ist es, Möglichkeiten zur Unterstützung der Semantikakquisition zu zeigen. Es gibt einige Veröffentlichungen, die ein ähnliches Ziel verfolgen, und viele Veröffentlichungen, aus denen bestimmte Aspekte für diese Aufgabe interessant sind. Diese werden im folgenden zusammengefaßt und kurz erläutert. An dieser Stelle erfolgt eine Übersicht über relevante Veröffentlichungen, die das Thema Semantikakquisition betreffen. Es wird anschließend versucht, diese Arbeiten nach den behandelten Aspekten zu sortieren.

In dieser Arbeit gibt es weitere Verweise auf andere Veröffentlichungen. Diese betreffen nur bestimmte Teilaufgaben dieser Arbeit und sind zum besseren Verständnis in die entsprechenden Kapitel eingeordnet.

1.4.1 Axiomatisierung von Integritätsbedingungen, Algorithmen für die Ableitung von Integritätsbedingungen

Aus einer Menge von Integritätsbedingungen lassen sich weitere Integritätsbedingungen ableiten. Dafür gibt es Ableitungsregeln, teilweise sind Algorithmen dazu einsetzbar.

Die Axiomatisierung von *funktionalen Abhängigkeiten* ist aus vielen Veröffentlichungen bekannt ([Mit83a], [Mit83b], [Lie85], [Ull88], [Tha91a], [BCN92], [MaR92] und [HeS95]), sie geht auf [Arm74] zurück. Ein Algorithmus zur Ableitung einer funktionalen Abhängigkeit aus einer Menge von funktionalen Abhängigkeiten wurde ebenfalls in [Ull88], [Tha91a], [BCN92], [MaR92], [HeS95] u.a. angegeben.

Negierte funktionale Abhängigkeiten sind ebenfalls axiomatisierbar, Ableitungsregeln und der Beweis dieser Ableitungsregeln wurden in [Jan89] angegeben.

Inklusionsabhängigkeiten sind nach [CFP84] axiomatisierbar. Die Algorithmen zur Ableitung von *Inklusionsabhängigkeiten* werden von Missaoui/Godin in [MiG90] auf bekannte Graphenalgorithmen zurückgeführt. Die Erstellung eines Graphen, der die Inklusionsabhängigkeiten einer Datenbank repräsentiert, wird beschrieben. Die Ableitung von weiteren Inklusionsabhängigkeiten aus der Menge der bekannten Inklusionsabhängigkeiten erfolgt auf dem Graphen. Die Bildung einer Basis der Inklusionsabhängigkeiten wird ebenfalls auf Graphalgorithmen zurückgeführt. Die Lösung der Algorithmen auf den Graphen ist nur über azyklischen oder unären Inklusionsabhängigkeiten möglich.

In [CFP84] sind Beispiele angegeben, wie aus Mengen von *funktionalen Abhängigkeiten und Inklusionsabhängigkeiten* weitere funktionale Abhängigkeiten und Inklusionsabhängigkeiten ableitbar sind. Eine vollständige Axiomatisierung von funktionalen Abhängigkeiten und Inklusionsabhängigkeiten und der Beweis der Vollständigkeit und Korrektheit der Ableitungsregeln wurde in [Mit83a] und [Mit83b] angegeben. Die Ableitung von funktionalen Abhängigkeiten und Inklusionsabhängigkeiten ist nicht in polynomialer Zeit ausführbar. Es gibt jedoch eingeschränkte Mengen der Integritätsbedingungen, für die das möglich ist. In [MaR92] wurden diese zusammengefaßt. Die Ableitbarkeit ist für funktionale Abhängigkeiten und unäre Inklusionsabhängigkeiten in polynomialer Zeit entscheidbar, der Beweis dazu wurde in [KCV83] geführt. Nach [MaR92] wird die Ableitbarkeit von funktionalen Abhängigkeiten und Inklusionsabhängigkeiten ebenfalls entscheidbar, wenn keine zirkulären Inklusionsabhängigkeiten auftreten.

In [CaV83] wurde ein System von Ableitungsregeln für *Inklusions- und Exklusionsabhängigkeiten* angegeben.

Kardinalitäten sind nicht axiomatisierbar, es gibt jedoch einige Ableitungsregeln für Kardinalitäten. Diese sind in [Tha98] zusammengetragen.

1.4.2 Tools zur Semantikakquisition

Es gibt einige Veröffentlichungen über Entwurfstools, in denen die Semantikakquisition als Teilaufgabe ausführlicher behandelt wird.

Silva/Melkanoff: Semantikerfragung anhand von Daten in Universal-Relationen. In [SiM81] wird ein Zugang beschrieben, in dem *funktionale Abhängigkeiten* anhand von *generierten Universal-Relationen* gesucht werden. In der dort vorgestellten Arbeit wird folgendes Vorgehen gewählt:
Eingabe ist eine Universal-Relation und bekannte funktionale und mehrwertige Abhängigkeiten. Es erfolgt eine Dekomposition der Relation entsprechend der Integritätsbedingungen. Anschließend wird eine Universal-Relation aus den gebildeten atomaren Relationen konstruiert, diese wird zur Illustration der Semantik verwendet. Dabei ist angegeben, welche funktionalen Abhängigkeiten in der Universal-Relation gelten müssen, um 1:1-, 1:n- und n:n- Kardinalitäten abzubilden. Die gebildete Universal-Relation wird dem Entwerfer gezeigt, es wird vorausgesetzt, daß der Entwerfer alle falschen Einträge in der Relation (zwei oder mehrere Tupel, die nicht gleichzeitig existieren können), erkennt. Dieses Vorgehen kann als *iterativer Prozeß zur informalen Semantikakquisition* verwendet werden.

Der Vorteil dieses Ansatzes ist der vollständig informale Zugang. Der Ansatz ist jedoch für größere Datenbanken schlecht anwendbar, da in diesem Fall die gebildeten Universal-Relationen sehr groß werden, sich widersprechende Einträge deshalb nicht immer erkannt werden.

Storey/Goldstein: Viewerstellungs-System mit pseudonatürlichsprachiger Semantikakquisition. In [StG88] wird ein *Viewerstellungs-System* beschrieben, das *Entity-Relationship Diagramme* für die Views des Benutzers entwickelt und diese in die vierte Normalform umwandelt. Dabei werden auch Informationen über Integritätsbedingungen ermittelt. Der Dialog zum Entwurf der Views ist *pseudonatürlichsprachig*, die Steuerung des Dialoges erfolgt über ein *Regelsystem*. Bei der Dialogführung werden Unvollständigkeiten in den Angaben des Benutzers beseitigt, indem gezielte Nachfragen gestellt werden. Aus natürlichsprachigen Eingaben des Entwerfers werden strukturelle Datenbank-Informationen abgeleitet. Durch eine einfache Heuristik, die Attributnamen auswertet, werden auch Schlüsselkandidaten ermittelt, die mit dem Benutzer diskutiert werden. Die Auswertung von Singular- und Pluralformen wird angewendet, um Kardinalitäten abzuleiten. Weitere Integritätsbedingungen werden ebenfalls pseudonatürlichsprachig erfragt, indem eine Umschreibung der Integritätsbedingungen erfolgt. Nach Angaben der Autoren haben ungeübte Entwerfer Schwierigkeiten bei der Arbeit mit dem System, für erfahrene Entwerfer stellt die Dialogführung eine wesentliche Hilfe beim Entwurf von Datenbanken dar.

Bouzeghoub, Gardarin, Metais: Expertensystem zum Datenbank-Entwurf. In [BGM85] und [BoG86] wird ein Expertensystem vorgestellt, das den Datenbank-Entwurf unterstützt. Dieses System arbeitet interaktiv mit dem Entwerfer. Zur Kommunikation wird ein sehr kleiner, formaler Ausschnitt der französischen Sprache gewählt. Informationen über die Datenbank werden aus solchen natürlichsprachigen Beschreibungen abgeleitet und entsprechende relationale Datenbanken erstellt. Diese werden graphisch dargestellt, dabei werden die Relationen und die Beziehungen zwischen den Relationen anschaulich dargestellt. Die Ableitung der semantischen Informationen erfolgt teilweise aus diesen Beschreibungen zur Datenbank. Es wird dabei nach festen Mustern in der natürlichsprachigen Beschreibung gesucht, die funktionale Abhängigkeiten oder bestimmte Kardinalitäten ausdrücken. Weitere Integritätsbedingungen werden über pseudonatürlichsprachige Fragen, in denen die Bedeutung der Integritätsbedingungen umschrieben wird, erfragt.
Es gibt zwei Einschränkungen der Menge der zu erfragenden Integritätsbedingungen: Damit dieser Dialog nicht zu langwierig wird, werden aus Beispieldaten funktionale Abhängigkeiten, die nicht gelten, abgeleitet. Es wird weiterhin zur Einschränkung der Menge der zu erfragenden funktionalen Abhängigkeiten vorgeschlagen, funktionale Abhängigkeiten mit mehr als vier oder fünf Attributen auf der linken Seite nicht zu betrachten.

1.4.3 Algorithmen zur Ableitung von Integritätsbedingungen aus Daten

Einige Integritätsbedingungen können aus vorhandenen Daten abgeleitet werden. Es gibt verschiedene Arbeiten, die sich mit dieser Aufgabe beschäftigen.

Castellanos/Saltor: Effiziente Ableitung von funktionalen Abhängigkeiten und Inklusionsabhängigkeiten aus Daten. In [CaS93] werden *funktionale Abhängigkeiten* und *Inklusionsabhängigkeiten* aus *Daten* abgeleitet. Es werden Algorithmen entwickelt, die diese *Ableitung effektivieren*, diese greifen in dem Fall, daß die Tupelanzahl sehr groß ist. Verschiedene Möglichkeiten zur Effektivierung der Ableitung von Integritätsbedingungen aus Daten

werden vorgestellt, anhand von Beispielen und Benchmarkmessungen werden die Vorteile gezeigt. Zur Absicherung, daß abgeleitete Integritätsbedingungen wirklich gelten, müssen diese direkt vom Benutzer bestätigt werden.

Orlowski: Iterative Ableitung von Schlüsseln und funktionalen Abhängigkeiten aus Daten. In [Orl92] erfolgt die Ableitung von *Schlüsseln* und *funktionalen Abhängigkeiten*, die in einer Relation gelten, nach folgender Idee:
Es wird zunächst davon ausgegangen, daß alle konstruierbaren Schlüssel und funktionalen Abhängigkeiten einer Relation gelten. Alle durch die Daten verletzten Integritätsbedingungen werden aus dieser Menge gelöscht, in der Menge der übrigen Integritätsbedingungen wird die Redundanz beseitigt. Durch die Suche nach *negierten Integritätsbedingungen* (Integritätsbedingungen, die durch die Relation verletzt sind), ist hierbei ein *inkrementelles Vorgehen* möglich. Dieses wird in dem Artikel erläutert. Bei der Ableitung von Integritätsbedingungen aus Daten wird eine Closed-World-Assumption vorausgesetzt, es erfolgt keine Bestätigung der abgeleiteten Integritätsbedingungen durch den Benutzer.

Bouzeghoub, Gardarin, Metais schlagen in [BGM85] und [BoG86] ebenfalls vor, Daten zur Ableitung von nicht geltenden funktionalen Abhängigkeiten zu nutzen. Ein Algorithmus oder eine Vorgehensbeschreibung erfolgt dort nicht, es ist nur ein Beispiel für das Vorgehen angegeben. In der Arbeit werden ebenfalls Kardinalitäten erfragt, die Ableitung von Kardinalitäten aus Daten ist in der gleichen Weise möglich, darauf wird in dem Artikel jedoch nicht verwiesen.

Kivinen/Mannila: Genäherte Ableitung von funktionalen Abhängigkeiten aus Relationen. In [KiM92] wird versucht, die Hülle der geltenden funktionalen Abhängigkeiten einer gegebenen Relation abzuleiten. Die bekannten Algorithmen für dieses Problem haben exponentielle Laufzeit in Bezug auf die Größe der Relation. In dieser Arbeit wird eine Abschätzung für diese funktionalen Abhängigkeiten vorgenommen, die wesentlich effizienter ausführbar ist. Dabei werden für die zu untersuchende Abhängigkeit Maße zwischen 0 (Abhängigkeit gilt) und 1 (Abhängigkeit ist verletzt) eingeführt.

Die Maße werden durch Auswertung folgender Eigenschaften bestimmt:

1. Anzahl der Tupelpaare, die die funktionale Abhängigkeit verletzen
2. Anzahl der Tupel, die in Tupelpaaren auftreten, die die funktionale Abhängigkeit verletzen
3. Anzahl der Tupel, die gelöscht werden müssen, damit die funktionale Abhängigkeit erfüllt ist

Die drei angegebenen Maße sind in polynomialer Zeit berechenbar.

Weiterhin wird untersucht, mit welcher Wahrscheinlichkeit durch den Algorithmus eine verletzte Abhängigkeit erkannt wird, wenn eine zufällig gewählte Teilmenge der Tupel einer gegebenen Relation ausgewählt wird. Es sind Grenzen dafür angegeben worden, z.B. kann eine verletzte funktionale Abhängigkeit einer Relation aus 30 Tupeln mit einer Wahrscheinlichkeit von 95 % gefunden werden.

Mannila/Räihä: Algorithmen zur korrekten Ableitung funktionaler Abhängigkeiten aus Relationen. In [MaR94] werden Algorithmen zur Ableitung von funktionalen Abhängigkeiten aus Relationen angegeben, diese sollen aber nicht wie in [KiM92] die funktionalen Abhängigkeiten abschätzen, sondern diese exakt ermitteln. Dazu wird zunächst ein naiver Algorithmus zur Ableitung funktionaler Abhängigkeiten aus Daten angegeben, die Komplexität dieses Algorithmus wird bestimmt. In der Praxis sind Relationen mit vielen funktionalen Abhängigkeiten in der Hülle nicht wahrscheinlich. Gesucht wird deshalb ein Algorithmus, der für Relationen mit wenigen funktionalen Abhängigkeiten schnell arbeitet, für Relationen mit vielen funktionalen Abhängigkeiten langsam. Dieser Algorithmus hat also ein besseres best-case-Verhalten als ein trivialer Algorithmus. Ein solcher Algorithmus ist angegeben, dieser soll in polynomialer Zeit bzgl. der Attributmengen auf der linken Seite von funktionalen Abhängigkeiten (minimale nichttriviale Attributmenge $X \subseteq R$, sodaß $X \longrightarrow A$ aus der Menge der Abhängigkeiten folgt) arbeiten. Weiterhin wird vorgeschlagen, die Ableitung von funktionalen Abhängigkeiten aus einer Relation auf einer Teilmenge der Relation (Auswahl einiger Tupel der Relation) durchzuführen und die Ergebnisse anschließend auf der gesamten Relation zu validieren.

1.4.4 Informale Diskussion von Integritätsbedingungen

Es sind zwei Möglichkeiten einer nicht formalen erfolgenden Diskussion von Integritätsbedingungen bekannt. Die erste Methode ist die *Darstellung von Integritätsbedingungen anhand von Beispielen* und Erfragung der Gültigkeit dieser Beispiele. Die zweite Methode ist eine *pseudo-natürlichsprachige Erfragung* von Integritätsbedingungen, wobei die Bedeutung der Integritätsbedingungen in der Fragestellung erläutert wird.

Erfragung von Integritätsbedingungen durch Beispiele. Ein mögliches Verfahren zur verständlichen Validierung und Erfragung von Integritätsbedingungen ist die Verwendung von *Armstrong-Datenbanken*. Eine Armstrong-Datenbank von einer Menge von Integritätsbedingungen Σ ist eine Datenbank, aus der sich genau die Integritätsbedingungen σ ableiten lassen, für die gilt: $\Sigma \vdash \sigma$. Weitere Integritätsbedingungen sind aus der Armstrong-Datenbank nicht ableitbar. Armstrong-Relationen für funktionale Abhängigkeiten gehen auf [Arm74] zurück.

Fagin/Vardi untersuchen die Existenz von Armstrong-Datenbanken ([FaV83]). Es wurde bewiesen, daß Armstrong-Datenbanken für Mengen von funktionalen Abhängigkeiten mit nichtleeren linken Seiten und Inklusionsabhängigkeiten existieren.

Mannila/Räihä geben in [MaR92] Algorithmen zur Konstruktion von Armstrong-Relationen für funktionale Abhängigkeiten und von Armstrong-Datenbanken für funktionale und schlüsselbasierte Inklusionsabhängigkeiten an.

Eine Validierung einer Menge von funktionalen Abhängigkeiten durch Diskussion von Armstrong-Universal-Relationen wurde von **Silva/Melkanoff** in [SiM81] angewendet.

Armstrong-Relationen können sehr groß werden, **Beeri/Down/Fagin/Statman** ermitteln in [BDF84] obere und untere Schranken von Armstrong-Relationen für funktionale Abhängigkeiten. Die Anzahl der Tupel in einer minimalen Armstrong-Relation kann exponentiell zur Attributanzahl sein.

Mannila/Räihä. Nach [MaR89c] ist die Größe von minimalen Armstrong-Datenbanken in starkem Maße von der Anzahl der geltenden Schlüssel abhängig. Für einige Anwendungen ist die Verwendung von Armstrong-Datenbanken deshalb möglich. Es wird in dem Artikel empfohlen, Vorschläge zur Normalisierung von Datenbanken mit Armstrong-Relationen zu unterlegen. Es gibt eine weitere Anwendung, in der die Erstellung von Armstrong-Datenbanken in Abhängigkeit von einem Problem reduziert wird. In [MaR89a] wird gezeigt, wie Armstrong-Datenbanken für eine konkrete Anfrage auf einer Datenbank erstellt werden können. Die so konstruierten Armstrong-Datenbanken enthalten meist weniger Tupel als universelle Armstrong-Datenbanken.

Bei der Arbeit mit *Armstrong-Datenbanken* sind generell folgende *Probleme* zu beachten:

- Armstrong-Relationen sind für große Relationen (d.h. Relationen mit vielen Attributen) zur Erfragung von Integritätsbedingungen schlecht geeignet, da die Armstrong-Relationen

für diese Relationen in der Regel sehr viele Tupel enthalten. Bei Armstrong-Relationen mit vielen Tupeln kann es passieren, daß ein Benutzer die Relationen als richtig bestätigt, weil er widersprüchliche Tupelpaare übersehen hat. Fehlende Schlüssel oder funktionale Abhängigkeiten sind aus einem falschen Tupelpaar in der Armstrong-Relation ersichtlich, in einer Relation mit vielen Tupeln kann es übersehen werden. Die Folge ist, daß dadurch falsche Integritätsbedingungen abgeleitet werden können.

- Aus einer Armstrong-Relation sind für einen Benutzer Integritätsbedingungen σ, die in Σ fehlen und ergänzt werden müssen, ersichtlich. Nicht ersichtlich sind jedoch Integritätsbedingungen σ, die in Σ auftreten, jedoch nicht gelten. Das muß bei der Konstruktion und der Erfragung der Gültigkeit von Armstrong-Datenbanken beachtet werden.

Für kleinere Anwendungen sind Armstrong-Relationen ein gut geeignetes Mittel zur informalen Diskussion von Integritätsbedingungen. Für größere Anwendungen sind diese jedoch ungeeignet.

Eine weitere Möglichkeit zur informalen Diskussion von Integritätsbedingungen ist eine natürlichsprachige oder pseudonatürlichsprachige Erfragung.

Pseudonatürlichsprachige Erfragung von Integritätsbedingungen. Zur Erfragung oder Bestätigung von funktionalen Abhängigkeiten und Kardinalitäten wird teilweise eine *pseudonatürlichsprachige Erfragung* verwendet (**Bouzeghoub/Gardarin/Metais** in [BGM85] und [BoG86], **Storey/Goldstein** in [StG88]). Dabei wird in der Fragestellung die Bedeutung der erfragten Integritätsbedingungen erläutert.

1.4.5 Heuristiken zur Semantiksuche

Bei der Unterstützung der Semantikakquisition durch Tools sollen einem Benutzer sinnvolle Vorschläge für geltende Integritätsbedingungen unterbreitet werden. Dazu muß man versuchen, das Hintergrundwissen der Datenbank auszuwerten, um plausible Vorschläge zu generieren. Es gibt verschiedene Arbeiten, in denen Heuristiken zur Ermittlung von Kandidaten für Integritätsbedingungen angegeben werden.

Storey/Goldstein: Heuristiken zur Schlüsselsuche und Kardinalitätenermittlung. In [StG88] werden einfache Heuristiken zur Bestimmung von Schlüsseln aus den Attributbezeichungen und den Attributtypen abgeleitet.

Choobineh/Venkatraman: Heuristiken zur Schlüsselsuche durch Auswertung von Formularen. In [ChV92] wird ein Weg vorgeschlagen, Formulare auf relationale Datenbanken abzubilden, dabei werden auch Schlüssel und funktionale Abhängigkeiten aus den Formularen abgeleitet. Es werden einfache Heuristiken eingesetzt, die Attributnamen auswerten und aus den Daten mögliche funktionale Abhängigkeiten ableiten. Diese Heuristiken werden intuitiv gewichtet und über eine ebenfalls intuitive Berechnung miteinander kombiniert. Diese Vorgehensweise läßt sich von der Formularanalyse auf den Datenbank-Entwurf übertragen.

Chiang/Barron/Storey: Heuristiken zur Fremdschlüsselermittlung. In [CBS94a], [CBS94b], [CBS94c] und [Chi94] werden Heuristiken zur Fremdschlüsselsuche für das Reverse-Engineering von Datenbanken gezeigt. Dabei werden Attributbezeichnungen und Daten ausgewertet.

Spitta: Namensfestlegung in Unternehmen zur Reduktion der Attributanzahl. In diesem Artikel wird für eine zentrale Namensfestlegung plädiert, die die Anzahl der Attributbezeichnungen in Unternehmen verringert. Die Bedeutung der Daten wird leichter erschließbar, wenn geeignete festgelegte Bezeichnungen verwendet werden. In dem Beitrag werden Regeln zur Festlegung von Namen und Abkürzungen angegeben.

Es wird weiterhin eine empirische Untersuchung von 1021 Modulen angegeben und deren Bezeichnungen analysiert. Dabei gibt es einige Bezeichnungen, die sehr häufig auftreten, die folgende Übersicht zeigt diese:

Nr	20.82%
Key	14.99%
Knz	11.65%
Dat	10.29%
Menge	7.68%
Bez/Name	4.96%

Grp	4.21%
Anz	3.84%
Zustand	2.73%
Code	2.6%
User	2.11%

Diese Studie kann man verwenden, um Namensheuristiken zur Abschätzung der Semantik abzuleiten. Sie ist nicht allgemeingültig, da die Bezeichnungen unter bestimmten eingeschränkten Bedingungen entstanden sind, gibt jedoch generelle Hinweise auf häufige Bezeichnungen. Nach diesen kann man mit entsprechenden Heuristikregeln gezielt suchen.

Die in diesem Abschnitt vorgestellten Heuristiken werden in Kapitel 5 dieser Arbeit ausführlich dargestellt und wesentlich erweitert. Weitere Heuristiken zur Ermittlung von Integritätsbedingungen werden dort ebenfalls vorgeschlagen.

1.4.6 Semantikakquisition im Reverse-Engineering von Datenbanken

Beim Reverse-Engineering erfolgt die Transformation von Datenbanken aus einem Datenmodell in ein höheres und semantisch aussagekräftigeres Datenmodell. Dazu sind Informationen über die Semantik der Datenbank erforderlich, damit diese Transformation korrekt durchgeführt werden kann.

In einzelnen Veröffentlichungen zum Reverse-Engineering wird dieses Problem unterschiedlich angegangen, teils werden die Informationen über die Semantik als Voraussetzung angenommen, teils gibt es Ansätze, diese für das Reverse-Engineering notwendigen Integritätsbedingungen zu ermitteln. Arbeiten dazu werden anschließend vorgestellt.

Markowitz/Makowsky: Transformation relationaler Datenbanken in Extended Entity-Relationship Strukturen. In [MaM90] wird ein Algorithmus vorgestellt, in dem erweiterte Entity-Relationship Modelle aus relationalen Datenbanken, Schlüsseln und schlüsselbasierten Inklusionsabhängigkeiten erstellt werden.
Es wird die Übersetzung erweiterter Entity-Relationship Modelle in relationale Datenbanken betrachtet. Die Umkehrung dieses Prozesses wird untersucht, dabei wird davon ausgegangen, daß sich die relationalen Datenbanken in Boyce-Codd Normalform befinden. Es werden nur solche Integritätsbedingungen zugelassen, die bei der Übersetzung der relationalen Datenbanken aus Entity-Relationship Modellen entstanden sind. Weiterhin wird vorausgesetzt, daß alle Inklusionsabhängigkeiten bekannt sind. Kenntnisse über geltende Integritätsbedingungen sind also Voraussetzung für den vorgestellten Algorithmus.

Hainaut/Tonneau/Joris/Chandelon: Semantikakquisition als Teil des Reverse-Engineering-Prozesses. In [HTJ93] wird auf die Notwendigkeit der Semantikakquisition als Teil des Reverse-Engineering von Datenbanken aufmerksam gemacht. Zur Lösung des Problems wird eine Analyse der prozeduralen Teile der Datenbank verlangt.

Chiang/Barron/Storey: Reverse-Engineering von relationalen Datenbanken in Entity-Relationship Modelle. In [CBS94a], [CBS94b], [CBS94c] und [Chi94] wurde ein Ansatz vorgestellt, der keine vollständigen Informationen über die Semantik voraussetzt, sondern Schlüssel und Inklusionsabhängigkeiten durch einfache Heuristiken ermittelt. Als Ausgangspunkt wird eine *strenge Namenssemantik* verlangt, gleiche Attribute müssen auch gleiche Attributnamen haben. Ermittelte Inklusionsabhängigkeiten werden zurückgewiesen, wenn diese in den *Instanzen* nicht erfüllt sind. Die Menge der auf diese Weise ermittelten Inklusionsabhängigkeiten kann jedoch unvollständig sein. Darauf ist in dem Ansatz hingewiesen worden, als Lösung wird eine manuelle Ergänzung der fehlenden Integritätsbedingungen vorgeschlagen.

Petit/Kouloumdjian/Boulicaut/Toumani: Auswertung der Daten durch Verwendung von Anfragen im Reverse-Engineering. Ziel des Artikels [PKB94] ist es, eine Methode vorzustellen, die die Qualität und Effizienz des Reverse-Engineering erhöht. Es soll ein höherer Grad der Automatisierung des Reverse-Engineering erreicht werden.
Dazu wird folgendes Vorgehen gewählt. *Datenbank-Instanzen* und *Ergebnisse von Anfragen* auf diesen Instanzen werden ausgewertet, um Informationen über die Semantik abzuleiten. Dazu werden Beispiele für Anfragen und Ergebnisse dieser angegeben. Weiterhin wird angegeben, welche Auswirkungen die Anfragen auf die zugehörigen konzeptuellen Schemata haben. Der Artikel beinhaltet also die Auswertung von Datenbank-Instanzen, um Semantik über die Datenbanken abzuleiten. Die angegebenen Transaktionen sind dazu ein Hilfsmittel.

Fahrner/Vossen: Transformation von relationalen in objektorientierte Datenbanken. In [FaV95a], [FaV95b] [FaV95c] und [FaV96] wird wie in den Arbeiten von Hainaut ein Vorgehen gewählt, das als erste Phase des Reverse-Engineering eine Vervollständigung und Anreicherung der existierenden Datenbank-Schemata vornimmt. Dazu wird untersucht, wie die relationale Darstellung von objekt-orientierten Schemata aussieht. Es wird ein Vorgehen zur Transformation relationaler Datenbanken in objekt-orientierte Datenbanken abgeleitet, das

aus den Phasen Vervollständigung, Transformation und Redesign besteht. Es werden Eigenschaften definiert, die auf dem vervollständigten relationalen Modell gelten sollen, um für die Transformation ideale Input-Informationen zu schaffen.

Bei der strukturellen und semantischen Vervollständigung von Schemata sind auch die Probleme *Analyse der Schemata* und *Ermittlung von Integritätsbedingungen* als Voraussetzung für Transformationsaufgaben relevant. Dabei tritt auch ein Schritt Ermittlung von Schlüsseln, funktionalen Abhängigkeiten, sowie Inklusions- und Exklusionsabhängigkeiten auf. Hier werden jedoch keine eigenen Ansätze vorgestellt, sondern auf andere Veröffentlichungen (u.a. Mannila/Räihä, Kivinen/Mannila, Johannesson/Kalman, Chiang/Barron/Storey) verwiesen. In diesem Artikel wird also die Bedeutung der Semantikakquisition für das korrekte Reverse-Engineering von objektorientierten Datenbanken aus relationalen Datenbanken unterstrichen.

Tauzovich: Intelligente Unterstützung des Datenbank-Entwurfes. Durch ein Expertensystem wird der Entwurfsprozeß von Datenbanken im erweiterten Entity-Relationship Modell unterstützt. Dabei werden Redundanzen, Inkonsistenzen usw. durch Anwendung von Regeln überprüft. Es sind einige Regeln für semantische Inkonsistenzen enthalten. Entities ohne Verbindung zu anderen Teilen der Datenbank werden gesucht, sowie drei Arten von sich ergebenden Problemen aus der Struktur und den Kardinalitäten werden ermittelt:

- Ein Entity hat 1:n-Beziehungen zu zwei anderen Entities, diese haben keine Verbindung untereinander, eine Verbindung sollte eventuell ergänzt werden.
- Transitive Relationships werden ermittelt, eine Verbindung bei diesen kann redundant sein.
- Die einzige Verbindung zwischen zwei Entities erfolgt über optionale Relationships, so daß für einige Instanzen die Verbindung, die eventuell bestehen sollte, nicht vorhanden ist.

Weiterhin werden Zyklen von Abhängigkeiten gesucht. Das Tool unterbricht nicht den Entwurfsprozeß, kann aber Vorschläge aufgrund dieser überprüften Merkmale generieren.

Die überprüften Regeln zur Inkonsistenzentdeckung sind sicher unvollständig, man könnte diese bestimmt noch um weitere (vage) Regeln erweitern, es ist aber sinnvoll, diese Regeln zur Überprüfung der Semantik anzuwenden.

1.5 Zielstellung der Arbeit

In dieser Arbeit soll gezeigt werden, welche Möglichkeiten es gibt, die geltenden Integritätsbedingungen einer Datenbank in informaler und allgemeinverständlicher, aber trotzdem effizienter Form zu erfragen.

Dazu werden in dieser Arbeit verschiedene Methoden zur Ermittlung der Semantik gezeigt, die sich in die in Abbildung **??** dargestellten drei Gruppen einsortieren lassen.

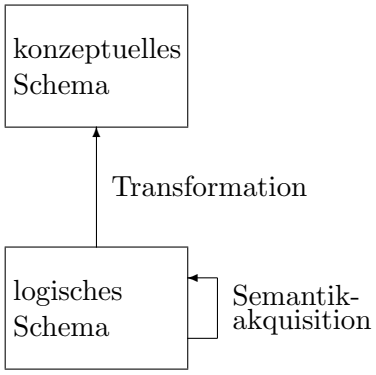


Abbildung 1.4: Semantikakquisition im Reverse-Engineering von Datenbanken

Zu jeder dieser drei Aufgaben werden in der Arbeit verschiedene Vorgehensweisen gezeigt. Die Gliederung der Arbeit ist ausführlich im folgenden Abschnitt erläutert.

1.6 Überblick über den Aufbau der Arbeit

In dieser Arbeit wird die Akquisition von Integritätsbedingungen im *Datenbank-Entwurf* und im *Reverse-Engineering* von Datenbanken dargestellt. Das Vorgehen ist für beide Anwendungsgebiete ähnlich, es wird eine Vorgehensweise zur Semantikakquisition beschrieben, die für beide Gebiete angewendet werden kann.

Im **Kapitel 2 (Theoretische Grundlagen)** erfolgt zunächst die Vorstellung des relationalen Datenmodells und des Entity-Relationship Modells sowie die Definition der Integritätsbedingungen auf dem relationalen Modell. Weiterhin wird eine Axiomatisierung der Integritätsbedingungen und Algorithmen zur Ableitung von Integritätsbedingungen aus anderen Integritätsbedingungen angegeben. Es wird gezeigt, welche Möglichkeit der Speicherung von Integritätsbedingungen für ein Tool zweckmäßig ist. Dabei wird angegeben, welche Basisoperationen in welcher Weise auf der Menge der Integritätsbedingungen ausgeführt werden müssen.

Im folgenden Kapitel sind die Methoden angegeben, mit denen Integritätsbedingungen aus verschiedenen Informationen abgeleitet werden können. Ziel ist dabei immer, für den Benutzer ein informales Vorgehen zu erreichen.

I Ableitung von Integritätsbedingungen

Einige Integritätsbedingungen lassen sich *ohne Bestätigung durch den Benutzer* ableiten. Im **Kapitel 3 (Auswertung von Daten und Datenbank-Statistik)** wird gezeigt, aus welchen Informationen Integritätsbedingungen abgeleitet werden können.
Vorhandene Daten können zur Ableitung von Integritätsbedingungen herangezogen werden. Welche Integritätsbedingungen dabei ableitbar sind, wird im *Abschnitt 3.2* erläutert. Algorithmen zur Ableitung der Integritätsbedingungen und Beispiele für das Vorgehen werden in diesem

Abschnitt angegeben.
Hat man Informationen über die Anzahl der Tupel, die in jeder Relation auftreten werden, und die Anzahl der Werte, die jedes Attribut der Relation annimmt (Datenbank-Statistik), so kann man daraus ebenfalls Informationen über Integritätsbedingungen ableiten. Im *Abschnitt 3.3* wird gezeigt, welche Integritätsbedingungen auf diese Weise ermittelt werden können. Algorithmen und Beispiele werden dazu angegeben.

Auch in einem Zugang, der Benutzer in informaler Weise bei der Semantikangabe unterstützt, muß es möglich sein, formale Integritätsbedingungen zu spezifizieren, die von einem Tool ausgewertet werden. Beim Datenbank-Entwurf gibt es einige formale Integritätsbedingungen, die aus der Darstellung der Datenbanken in konzeptuellen Datenmodellen resultieren. Im Reverse-Engineering können einige formale Integritätsbedingungen bekannt sein, diese möchte man berücksichtigen. Weiterhin kann es sowohl im Datenbank-Entwurf als auch im Reverse-Engineering vorkommen, daß ein Benutzer einige formale Integritätsbedingungen spezifizieren kann. Im **Kapitel 4 (Unterstützung von expliziten Angaben zur Semantik)** wird gezeigt, wie diese Integritätsbedingungen erfragt und überprüft werden können.

Durch die Auswertung von Daten, Datenbank-Statistik-Angaben und durch explizite Eingaben sind einige Informationen über Integritätsbedingungen der Datenbank bekannt. Die Ableitung weiterer Integritätsbedingungen aus vorhandenen Datenbanken ist ohne Bestätigung durch den Benutzer nicht möglich.

II Ableitung von Kandidaten

Die Anzahl der unbekannten Integritätsbedingungen einer Datenbank ist in der Regel zu groß, um der Reihe nach zu überprüfen, ob diese Integritätsbedingungen gültig sind. Deshalb muß vor der Erfragung von Integritätsbedingungen eine Abschätzung und Vorauswahl der Integritätsbedingungen erfolgen, um eine intelligente und effiziente Diskussion von Integritätsbedingungen zu erreichen.

Es lassen sich Kandidaten für Integritätsbedingungen mit Hilfe von Heuristiken ableiten, die Heuristiken werten die Hintergrundinformationen der Datenbank aus. Kandidaten für Integritätsbedingungen können aus den Daten, den strukturellen Angaben, der bereits bekannten Menge von Integritätsbedingungen und aus den bekannten Transaktionen abgeleitet werden. In **Kapitel 5 (Heuristiken zur Abschätzung unbekannter Integritätsbedingungen)** wird gezeigt, über welche Heuristikregeln diese Ableitungen erfolgen können und wie die einzelnen Heuristikregeln miteinander kombiniert und gewichtet werden können.

Eine wichtige Quelle semantischer Informationen einer Datenbank sind auch gleiche oder ähnliche Datenbanken, aus denen bekannte semantische Informationen übernommen werden können. Im **Kapitel 6 (Wiederverwendung von vorhandenen Datenbank-Schemata)** wird gezeigt, wie gleiche oder ähnliche Datenbankteile ermittelt werden können und wie die Übernahme semantischer Informationen aus ähnlichen Datenbanken erfolgen kann. Da die Ermittlung von gleichen und ähnlichen Datenbankteilen sich auf vage Informationen stützt, werden auf diese Weise keine Integritätsbedingungen, aber sinnvolle Kandidaten für Integritätsbedingungen ermittelt. Diese müssen validiert werden.
Die Wiederverwendung von vorhandenen Datenbanken kann nicht nur zur Ableitung von sinnvollen Kandidaten für Integritätsbedingungen verwendet werden. Verschiedene weitere Entwurfsaufgaben können ebenfalls durch dieses Vorgehen unterstützt werden. In Kapitel 6 werden diese aufgezählt und erläutert.

In den Kapiteln 5 und 6 wird erläutert, wie sich durch die Auswertung von Heuristikregeln Kandidaten für Integritätsbedingungen ermitteln lassen. Man kann durch Heuristiken weitere Informationen über die Bedeutung einer Datenbank ableiten - diese werden als *Metainformationen* bezeichnet. Metainformationen sind Informationen über die Semantik der Datenbank, die zur Steuerung und Untergliederung der Semantikakquisition eingesetzt werden können. Durch zahlreiche Heuristikregeln werden *Cluster, unabhängige Einheiten und Kernrelationen* der Datenbank ermittelt. Im **Kapitel 7 (Metainformationen über die Datenbank)** wird angegeben, welche Metainformationen über eine Datenbank abgeleitet werden können und wie sie bei der Semantikakquisition verwendet werden.

III Validierung von Kandidaten

Die Reihenfolge der Diskussion von Integritätsbedingungen hat Einfluß auf die Anzahl der notwendigen Dialogschritte. Der Dialog wird nach verschiedenen Kriterien wie Effizienz, Nachvollziehbarkeit und Suche nach den wichtigsten Integritätsbedingungen gesteuert. Im **Kapitel 8 (Bestimmung der Erfragungsreihenfolge)** wird gezeigt, wie die Dialogsteuerung diesen widersprüchlichen Anforderungen gerecht werden kann. Dabei werden die ermittelten Kandidaten für Integritätsbedingungen (Kapitel 5) und die Metainformationen über die Datenbank (Kapitel 7) einbezogen.

Das **Kapitel 9 (Validierung von Integritätsbedingungen)** erläutert, wie die einzelnen Kandidaten für Integritätsbedingungen validiert werden können. Dazu wird ein Ansatz vorgeschlagen, bei dem Integritätsbedingungen durch Kombination einer Beispieldiskussion und einer pseudonatürlichsprachigen Erfragung vom Benutzer diskutiert werden.

Entwurf und Re-Engineering von Datenbanken sind iterative Prozesse. Die Semantikakquisition ist eine Teilaufgabe dieser Prozesse. Es ist möglich, daß die Semantikakquisition bereits für ein Datenbank-Schema erfolgte, dieses Datenbank-Schema aber anschließend verändert wurde. Man möchte in diesem Fall bereits erkannte Integritätsbedingungen soweit möglich übernehmen. In **Kapitel 10 (Übernahme von Integritätsbedingungen in veränderte Datenbank-Schemata)** wird gezeigt, welche *Integritätsbedingungen* nach Ergänzen, Löschen und Ändern von Attributen und Relationen-Schemata auf welche Weise *übernommen* werden können und welche bereits bekannten Integritätsbedingungen aufgrund der Änderungen erneut valisiert werden müssen und demzufolge nur als *Kandidaten* übernommen werden können.

In den Kapiteln 3 - 10 werden verschiedene Methoden zur Semantikakquisition vorgestellt. In **Kapitel 11 (Gesamtalgorithmus zur Unterstützung der Semantikakquisition)** wird gezeigt, wie diese einzelnen Methoden in einem Tool zur Unterstützung der Semantikakquisition sinnvoll miteinander verbunden werden können. Es wird angegeben, welche Methoden dabei unbedingt erforderlich sind und welche Methoden optional einsetzbar sind. Weiterhin wird gezeigt, wie Ergänzungen vorgenommen werden können, wenn weitere Informationen vorhanden sind, aus denen Integritätsbedingungen abgeleitet werden können. In dem Kapitel wird begründet, welche Methoden vor welchen anderen ausgeführt werden müssen, um eine optimale Unterstützung der Semantikakquisition zu erreichen.

In **Kapitel 12 (Zusammenfassung — Möglichkeiten und Grenzen der informalen Semantikakquisition)** erfolgt eine Zusammenfassung, in der allgemeine Eigenschaften der vor-

gestellten Semantikakquisition angegeben werden. Es wird gezeigt, wie verschiedene Typen von Benutzern bei verschiedenen Aufgaben durch die vorgestellte Semantikakquisition unterstützt werden können. Auftretende Probleme bei der Semantikakquisition werden angegeben. Einige mögliche Erweiterungen der Methode werden ebenfalls kurz beschrieben.

Kapitel 2

Theoretische Grundlagen

In diesem Kapitel werden die dieser Arbeit zugrunde liegenden *Datenmodelle* eingeführt, sowie *Integritätsbedingungen* über diesen Datenmodellen definiert.

In Abschnitt 2.1 erfolgt die formale Definition des *relationalen Datenmodells* als logisches Datenmodell und des *Entity-Relationship Modells* als konzeptionelles Datenmodell. Es werden über dem relationalen Datenmodell *Integritätsbedingungen* definiert, die die Semantik der Datenbanken in formaler Form angeben. Das *allgemeine Vorgehen bei der Semantikakquisition* wird in Abschnitt 2.2 angegeben. In Abschnitt 2.3 erfolgt eine *Axiomatisierung* der eingeführten Integritätsbedingungen, Algorithmen zur Ableitung von Integritätsbedingungen werden beschrieben. Es gibt verschiedene Möglichkeiten, erkannte Informationen über die Semantik zu speichern. Im letzten Abschnitt dieses Kapitels werden diese vorgestellt und es wird betrachtet, welche Vor- und Nachteile die verschiedenen Möglichkeiten bieten. *Basisoperationen* für die Menge der Integritätsbedingungen werden angegeben.

2.1 Datenmodelle

Die in dieser Arbeit vorgestellte Akquisition der Semantikinformationen über Datenbanken soll auf dem relationalen Datenmodell erfolgen. Dieses Datenmodell ist ein logisches Datenmodell, es wird anschließend kurz erläutert. Integritätsbedingungen werden über dem relationalen Datenmodell definiert.
Das Entity-Relationship Modell (ERM) ist ein konzeptuelles Datenmodell, Datenbanken in diesem Modell lassen sich in relationale Datenbanken übersetzen. Das Entity-Relationship Modell wird anschließend kurz in seiner Grundform vorgestellt, es wurde vielfach erweitert. Einige Erweiterungen des Entity-Relationship Modells werden ebenfalls vorgestellt. Eine einfache Übersetzung von Datenbanken des Entity-Relationship Modells in relationale Datenbanken wird ebenfalls gezeigt.

2.1.1 Relationale Datenbanken

Relationale Datenbanken sind nach [HeS95] gegenwärtig das am weitesten verbreitete logische Datenmodell. Eine Ursache ist wahrscheinlich die Einfachheit des Datenmodells und die

anschauliche tabellarische Darstellung der Relationen. Nach [KüL94] verwenden 94 % der befragten Betriebe in der Schweiz, die Datenbanken einsetzen, unter anderem auch relationale Datenbanken.

Relationale Datenbanken gehen auf Codd zurück und werden aufgrund ihrer Verbreitung in sehr vielen Veröffentlichungen und Lehrbüchern ausführlich erklärt und definiert. An dieser Stelle soll eine formale Definition der relationalen Datenbanken erfolgen, da die in dieser Arbeit beschriebene Semantikakquisition auf der Basis relationaler Datenbanken erfolgt. Für eine anschaulichere Erklärung der relationalen Datenbanken, die auch mit vielen Erläuterungen und Beispielen erfolgte, sei u.a. auf [Ull88], [PDG89] und [HeS95] verwiesen.

Vor der Definition des Datenmodells erfolgt hier eine sehr kurze und informale Erläuterung, die Lesern, die dieses Datenmodell nicht kennen, helfen soll, die nachfolgenden Definitionen zu verstehen. Diese ist in [MaR92] in wesentlich ausführlicherer Form zu finden.

Im relationalen Datenmodell werden die Daten in Tabellen gespeichert. Die Spalten der Tabellen sind mit Namen versehen, diese Namen werden als *Attribute* bezeichnet. Jedes Attribut ist mit einem *Wertebereich* verbunden, der angibt, welche Einträge in der betreffenden Spalte der Tabelle auftreten können. Die Zeilen der Tabelle werden als *Tupel* bezeichnet, eine Menge von Tupeln wird *Relation* genannt.
Die Attribute definieren die Struktur der gespeicherten Daten, sie werden als *Relationen-Schemata* bezeichnet. Relationale *Datenbanken* bestehen aus mehreren Tabellen, also aus einer Menge von Relationen. Ein *Datenbank-Schema* besteht aus einer Menge von Relationen-Schemata.

Im nächsten Abschnitt werden diese bereits kurz eingeführten Begriffe des relationalen Datenmodells formal definiert. Die im folgenden dargestellten Definitionen erfolgen in Anlehnung an [PDG89].

Definition 2.1 *Ein* **einfaches Relationen-Schema** *ist ein Dreitupel*

$$ERS = (U, \underline{D}, dom)$$

wobei

- U *eine endliche Menge von Attributen ist,*
- $\underline{D}$ *ist eine endliche Menge von Wertebereichen und*
- $dom : U \longrightarrow \underline{D}$ *ist eine Funktion, die jedes Attribut mit seinem Wertebereich verbindet.*

Die Definition der einfachen Relationen-Schemata kann um Integritätsbedingungen erweitert werden. Das führt zur Definition von Relationen-Schemata.

Definition 2.2 *Ein* **Relationen-Schema** *ist ein Zweitupel*

$$R = (ERS, I_l)$$

wobei

- ERS *ein einfaches Relationen-Schema ist und*
- I_l *ist eine Menge von lokalen Integritätsbedingungen, die Bedeutung dieser wird in Definition 2.4 erklärt.*

Relationen-Schemata geben die Struktur der Relationen an. Die aktuelle Information, die in Relationen gespeichert wird, wird durch Relationen-Instanzen beschrieben. Diese werden wie folgt formal definiert:

Definition 2.3 *Sei* $ERS = (U, \underline{D}, dom)$ *ein einfaches Relationen-Schema.*

- *Ein* **Tupel** *über ERS ist eine Funktion* $t, t : U \longrightarrow \bigcup_{d \in \underline{D}} \underline{D}$*, so daß für jedes Attribut* A *gilt:* $t(A) \in dom(A)$*.*
- *Eine* **mögliche Relationen-Instanz** *eines einfachen Relationen-Schemas ERS ist eine Menge von Tupeln über ERS.*

In einer möglichen Relationen-Instanz treten beliebige Tupel auf. Um diese Menge auf eine sinnvolle Menge von Tupeln zu beschränken, werden die lokalen Integritätsbedingungen ausgewertet.

Definition 2.4 *Sei* $R = (ERS, I_l)$ *ein Relationen-Schema.*

- *Eine lokale Integritätsbedingung des Relationen-Schemas wird durch eine Boolesche Funktion repräsentiert, die jeder möglichen Relationen-Instanz des einfachen Relationen-Schemas ERS den Wert true oder false zuordnet.*
- *Wenn die Funktion den Wert true einer möglichen Relationen-Instanz zuordnet, dann erfüllt die Relationen-Instanz die lokale Integritätsbedingung.*
- *Eine* **Relationen-Instanz oder Relation** *r des Relationen-Schemas R ist eine mögliche Relationen-Instanz von ERS, die alle lokalen Integritätsbedingungen* I_l *erfüllt.*

Mehrere Relationen-Schemata können zu einem Datenbank-Schema zusammengefaßt werden.

Definition 2.5 *Ein* **einfaches Datenbank-Schema** *EDBS ist eine endliche Menge von Relationen-Schemata* $EDBS = \{ R_i = (U_i, \underline{D_i}, dom_i, I_{li} | i \in I) \}$*, sodaß* $\forall i, j \in I$ *und* $A \in U, A \in U_i \cap U_j$ *gilt:* $dom_i(A) = dom_j(A)$*. Die Funktionen* dom_i *können also zu einer Funktion* $dom_i : \bigcup U_i \longrightarrow \underline{D_i}$ *zusammengefaßt werden.*

Die einfachen Datenbank-Schemata können um globale Integritätsbedingungen erweitert werden, die über mehreren Relationen der Datenbank definiert werden. Das führt zu folgender Definition.

Definition 2.6 *Ein* **Datenbank-Schema** *D ist ein Zweitupel*

$$D = (EDBS, I_g)$$

wobei

- *EDBS ein einfaches Datenbank-Schema ist und*
- I_g *ist eine Menge globaler Integritätsbedingungen der Datenbank, die Bedeutung dieser wird in Definition 2.8 angegeben.*

Die aktuelle Information, die in Datenbanken gespeichert wird, wird durch Datenbank-Instanzen beschrieben. Diese werden im folgenden definiert:

Definition 2.7 *Sei EDBS ein einfaches Datenbank-Schema. Eine* **mögliche Datenbank-Instanz** *des Datenbank-Schemas ist eine Menge von Relationen $(r_1..r_n)$, so daß genau eine Relation r_i für jedes Relationen-Schema R_i in EDBS existiert.*

Auch diese möglichen Datenbank-Instanzen sollen durch Auswertung der globalen Integritätsbedingungen auf eine sinnvolle Menge von Datenbank-Instanzen eingeschränkt werden:

Definition 2.8 *Sei $D = (EDBS, I_g)$ ein Datenbank-Schema.*

- *Eine globale Integritätsbedingung des Datenbank-Schemas wird durch eine Boolesche Funktion repräsentiert, die jeder möglichen Datenbank-Instanz des einfachen Datenbank-Schemas den Wert true oder false zuordnet.*
- *Wenn die Funktion den Wert true einer möglichen Datenbank-Instanz zuordnet, dann erfüllt die Datenbank-Instanz die globale Integritätsbedingung.*
- *Eine* **Datenbank oder Datenbank-Instanz** *d des Datenbank-Schemas D ist eine mögliche Datenbank-Instanz von EDBS, die alle globalen Integritätsbedingungen I_g erfüllt.*

Über den relationalen Datenbanken können *Operationen* definiert werden. Diese sind in ausführlicher Darstellung in [MaR92], [PDG89], [HeS95], u.a. zu finden.

Eine einstellige Operation auf relationalen Datenbanken ist die *Projektion*. Die Projektion wird mit dem Ziel durchgeführt, eine Tabelle (Relation) auf die interessierenden Spalten (Attribute) zu reduzieren. Die *Projektion eines Tupels t* auf die Attributmenge X (geschrieben $t[X]$), ist eine Funktion t', die eine Zuordnung $X \rightarrow dom(X)$ bestimmt, sodaß $t'(A) = t(A)$ für jedes $A \in X$. Die *Projektion einer Relation r* auf die Attributmenge X (geschrieben $r[X]$) wird definiert als $r[X] = \{t[X] | t \in r\}$

Eine weitere einstellige Operation auf relationalen Datenbanken ist die *Selektion*. Das Ziel der Selektion ist die gezielte Auswahl einer Teilmenge von Tupeln einer Relation. Das Auswahlkriterium ψ ist dabei eine Formel, die aus Konstanten, Attributnamen, Vergleichsopratoren und logischen Operatoren zusammengesetzt wird. Die Selektionsoperation basiert auf diesem Auswahlkriterium. Sei ψ gegeben, dann ist die Selektion einer Relation definiert als: $\sigma_{\psi(r)} = \{t \in r | t$ erfüllt $\psi\}$.

Diese Operationen werden benötigt, um die Integritätsbedingungen der Relationen-Schemata bzw. Datenbank-Schemata zu definieren.

Die *lokalen und globalen Integritätsbedingungen* werden im weiteren auch kurz nur als *Integritätsbedingungen* der Relationen-Schemata bzw. Datenbank-Schemata bezeichnet. Diese werden in den folgenden Abschnitten im einzelnen definiert.

2.1.2 Integritätsbedingungen der relationalen Datenbanken

In der Definition der relationalen Datenbanken treten lokale und globale Integritätsbedingungen auf, die die Korrektheit der Datenbanken sichern. Diese wurden bisher nur als allgemeine Boolesche Funktionen eingeführt. In diesem Abschnitt werden die Integritätsbedingungen definiert.

Lokale Integritätsbedingungen der Relationen-Schemata

Innerhalb eines Relationen-Schemas können Schlüssel und funktionale Abhängigkeiten als lokale Integritätsbedingungen definiert werden. Funktionale Abhängigkeiten und Schlüssel sind allgemein bekannt, viele Lehrbücher geben ausführliche Motivationen und zahlreiche Beispiele für diese an (z.B. [Ull88], [MaR92] und [HeS95]). Mit Hilfe von funktionalen Abhängigkeiten werden Zusammenhänge zwischen den Attributen eines Relationen-Schemas dargestellt, diese werden zur Sicherung der Konsistenz von Datenbanken benötigt. Deshalb sind funktionale Abhängigkeiten sehr wichtige Integritätsbedingungen in relationalen Datenbanken.

Funktionale Abhängigkeiten. Eine funktionale Abhängigkeit zwischen den Attributmengen X und Y gibt an, daß die Werte der Attributmenge X die Werte der Attributmenge Y in einer Relation bestimmen. Funktionale Abhängigkeiten werden wie folgt definiert.

Definition 2.9 *Sei $ERS = (U, \underline{D}, dom)$ ein einfaches Relationen-Schema und $X, Y \subseteq U$. Eine* **funktionale Abhängigkeit (FD)** *$X \longrightarrow Y$ über ERS ist eine Integritätsbedingung, die durch eine mögliche Relationen-Instanz r erfüllt wird, genau dann wenn folgende Bedingung gilt: $\forall\, t_1, t_2 \in r$: $t_1[X] = t_2[X] \longrightarrow t_1[Y] = t_2[Y]$.*

Schlüssel. Ein Schlüssel eines Relationen-Schemas ist die Attributkombination, die ein Tupel in jeder Relation eines Relationen-Schemas identifizierbar macht. Schlüssel in Relationen-Schemata werden wie folgt definiert.

Definition 2.10 *Sei $ERS = (U, \underline{D}, dom)$ ein einfaches Relationen-Schema und $X \subseteq U$. Ein* **Schlüssel (KEY)** *X über ERS ist eine Integritätsbedingung, die durch eine mögliche Relationen-Instanz r erfüllt wird, genau dann wenn sich die Projektion aller Tupel der Relation auf die Attributmenge X voneinander unterscheidet: $\forall\, t, t' \in r$ mit $t \neq t'$ gilt: $t[X] \neq t'[X]$.*

Definition 2.11 *Sei $ERS = (U, \underline{D}, dom)$ ein einfaches Relationen-Schema und $X \subseteq U$. Ein* **minimaler Schlüssel** *X über ERS ist eine Integritätsbedingung, die durch eine mögliche Relationen-Instanz r erfüllt wird, genau dann wenn folgende Bedingung gilt:*
X ist Schlüssel der Relationen-Instanz r und es gibt keine echte Teilmenge von X, die ebenfalls Schlüssel von r ist.

Globale Integritätsbedingungen der Datenbank-Schemata

Inklusionsabhängigkeiten. Inklusionsabhängigkeiten sind in relationalen Datenbanken notwendig, um Verbindungen zwischen den einzelnen Relationen-Schemata herzustellen. Inklusionsabhängigkeiten geben an, daß Werte von einer Attributmenge einer Relation auch in einer anderen Attributmenge einer anderen Relation auftreten müssen. Die folgende Definition der Inklusionsabhängigkeiten ist u.a. in [Mit83a], [Mit83b], [MaR92] und [PBG89] zu finden.

Definition 2.12 *Sei* $EDBS = \{.., R, .., S, ..\}$ *ein einfaches Datenbank-Schema und* $R = (U_1, \underline{D_1}, dom_1, I_{l1})$ *und* $S = (U_2, \underline{D_2}, dom_2, I_{l2})$ *seien Relationen-Schemata mit* $X = \langle X_1..X_n \rangle$ *Attribute aus* U_1 *und* $Y = \langle Y_1..Y_n \rangle$ *Attribute aus* U_2.
Eine **Inklusionsabhängigkeit (ID)** $X \subseteq Y$ *ist durch mögliche Relationen-Instanzen* r *von* R *und* s *von* S *erfüllt, genau dann wenn es* $\forall\ t \in r$ *ein Tupel* $t' \in s$ *gibt, für das* $\forall\ 1 \leq i \leq n$ $t[X_i] = t'[Y_i]$ *gilt.*

Exklusionsabhängigkeiten. Exklusionsabhängigkeiten geben an, daß in zwei Attributsequenzen zweier Relationen keine gleichen Werte auftreten dürfen.

Exklusionsabhängigkeiten können in vielen Datenmodellen, z.B. im Entity-Relationship Modell nicht strukturell dargestellt werden. Die meisten Ansätze zur Akquisition von Integritätsbedingungen in Datenbanken bauen daher nur auf Schlüsseln, funktionalen Abhängigkeiten und Inklusionabhängigkeiten auf. Exklusionsabhängigkeiten sind jedoch für Konsistenzprüfungen erforderlich. Exklusionsabhängigkeiten werden bei der Transformation von Datenbanken in objektorientierte Modelle besonders wichtig, hier sind sie im Datenmodell darstellbar. Sie geben in diesem Fall disjunkte Klassenzerlegungen an. In verteilten Datenbanken geben Informationen über Exklusionsabhängigkeiten Hinweise zur sinnvollen Verteilung. Deshalb ist es für verschiedene Anwendungen nützlich, die Exklusionsabhängigkeiten zu ermitteln.

Definition 2.13 *Sei* $EDBS = \{.., R, .., S, ..\}$ *ein einfaches Datenbank-Schema und* $R = (U_1, \underline{D_1}, dom_1, I_{l1})$ *und* $S = (U_2, \underline{D_2}, dom_2, I_{l2})$ *Relationen-Schemata mit* $X = \langle X_1..X_n \rangle$ *Attribute von* U_1 *und* $Y = \langle Y_1..Y_n \rangle$ *Attribute von* U_2.
Eine **Exklusionsabhängigkeit (ED)** $X \,||\, Y$ *ist durch mögliche Relationen-Instanzen* r *von* R *und* s *von* S *erfüllt, genau dann wenn es kein* $t \in r$ *und* $t' \in s$ *gibt, für die*
$t[X_i] = t'[Y_i]$, $\forall\ 1 \leq i \leq n$.

Kardinalitäten. Kardinalitäten sind Integritätsbedingungen, die für das Entity-Relationship Modell eingeführt wurden. Sie spiegeln sich in analoger Form im relationalen Datenmodell wider. Werden Datenbanken im Entity-Relationship Modell in relationale Datenbanken übersetzt, so findet man die entsprechenden Kardinalitäten im relationalen Modell wieder. Durch Kardinalitäten können funktionale Abhängigkeiten und Inklusionsabhängigkeiten ausgedrückt werden. Die Definition von Kardinalitäten im relationalen Datenmodell ist trotzdem unüblich. Ich möchte diesen Weg gehen, da in der Arbeit Methoden zur Semantikakquisition vorgestellt werden, bei denen mit Daten gearbeitet wird. Deshalb werden diese Methoden im relationalen Datenmodell durchgeführt. Die Semantikakquisition auf dem relationalen Modell kann das Ziel haben, Angaben zur Semantik, die in einem zugehörigen Entity-Relationship Modell strukturell dargestellt wurden, zu überprüfen. Es kann auch möglich sein, daß man die relationale Datenbank in ein konzeptuelles Datenmodell (Entity-Relationship Modell) transformieren möchte. In beiden Fällen benötigt man Informationen zu den Kardinalitäten, die im logischen Datenmodell gesucht und mit dem Benutzer anhand von Beispielen diskutiert werden sollen. Deshalb erfolgt an dieser Stelle die Definition von Kardinalitäten auf dem relationalen Datenmodell.

Kardinalitäten können in Form von Minimum-Maximum-Kardinalitäten angegeben werden. In einer relationalen Datenbank gibt die Kardinalität zwischen zwei Relationen an, wie oft die Werte einer Relation mindestens und wie oft sie maximal in einer anderen Relation auftreten.

Definition 2.14 *Seien* $R = (U_1, \underline{D_1}, dom_1, I_{l1})$ *und* $S = (U_2, \underline{D_2}, dom_2, I_{l2})$ *zwei Relationen-Schemata eines Datenbank-Schemas* D *mit* $X = \langle X_1..X_n \rangle$ *Attribute aus* U_1, X *- Schlüssel in* R_1 *und* $Y = \langle Y_1..Y_n \rangle$ *Attribute aus* U_2.
Die **Kardinalität** $\mathsf{card}(S, R) = (m, n)$ *ist durch Relationen-Instanzen* r *von* R *und* s *von* S *erfüllt, genau dann wenn für jedes Tupel* t *von* r *mindestens* m *und maximal* n *Tupel* t' *von* s *existieren, für die gilt:* $\forall\ 1 \leq i \leq n\ t[X_i] = t'[Y_i]$.

Da es bei einer Semantikakquisition, die durch eine Beispieldiskussion erfolgt, zu aufwendig wäre, alle möglichen Kardinalitäten zu untersuchen, wird die Menge der Kardinalitäten zunächst auf die Minimumwerte 0 und 1 und die Maximumwerte 1 und n beschränkt. Daraus ergeben sich die Kardinalitäten (0,1), (1,1), (0,n), (1,n), die zunächst unterschieden werden. Exaktere Werte für die Kardinalitäten werden durch gezielte Nachfragen ermittelt.

2.1.3 Das Entity-Relationship Modell

Das Entity-Relationship Modell (ERM) wird als anschauliches Mittel zur Darstellung von konzeptuellen Datenbanken angesehen. Es geht auf Chen [Che76] zurück. Für das Entity-Relationship Modell gibt eine einfache und anschauliche graphische Darstellung. Diese beinhaltet die Darstellung von Entity-Typen, Relationship-Typen und Attributen.

Ein *Entity* ist ein existierendes Objekt der realen Welt, eine Ganzheit oder Einheit. Beziehungen zwischen den Entities werden als Relationships bezeichnet. Gleichartige Entities werden zu *Entity-Typen* zusammengefaßt. Gleichartige Relationships werden zu *Relationship-Typen* zusammengefaßt. Die Eigenschaften oder Merkmale eines Entity-Typen oder eines Relationship-Typen heißen *Attribute*. Abbildung 2.1 zeigt die graphische Darstellung dieser Komponenten.

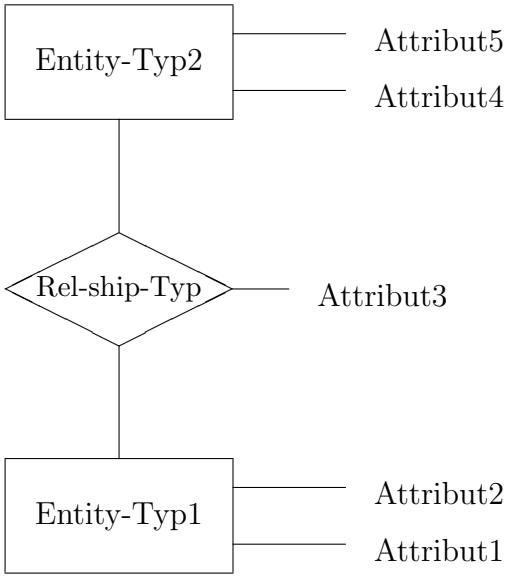


Abbildung 2.1: Graphische Darstellung des Entity-Relationship Modells

Für eine ausführlichere Darstellung dieser Begriffe sei auf zahlreiche Datenbanklehrbücher wie [Ull88], [Yao85], [BCN92], [MaR92], [VRT82] und [HeS95] verwiesen.

Erweiterungen des Entity-Relationship Modells

Es gibt verschiedene Erweiterungen des klassischen Entity-Relationship Modells von Chen. Diese haben das Ziel, universellere Darstellungsmöglichkeiten zu schaffen.

Es sollen hier keine erweiterten Entity-Relationship Modelle vollständig vorgestellt werden, sondern einige Erweiterungen, die in verschiedenen Modellen vorkommen, kurz erwähnt werden. Diese Zusammenfassung wurde teilweise aus [HeS95] übernommen, sie erhebt keinen Anspruch auf Vollständigkeit:

- **Erweiterungen bei Attributen.**
  Attribute können nicht nur atomare Attribute sein, es können Mengen und Tupel von Attributen als Attribute auftreten [ElN94].
  Das Higher-Order Entity-Relationship Model ([BOT90], [Tha91b], [Tha91c] und [Tha98]) ist eine Erweiterung des Entity-Relationship Modells, bei der als Attribute neben atomaren Attributen auch Mengen, Tupel und Listen von Attributen auftreten können.
  In einer Erweiterung des Entity-Relationship Modells im System ESCHER ([PTW94]) werden mit Mengen, Tupel, Listen, Multimengen, Links, Varianten, usw. weitere komplexe Attribute zugelassen.

- **Höhere Relationship-Typen.**
  Das von Chen eingeführte Entity-Relationship Modell beinhaltete nur binäre Relationship-Typen. Es gibt zahlreiche Erweiterungen des Entity-Relationship Modells, in denen auch ternäre und höhere Relationship-Typen verwendet werden. Diese werden über drei oder mehreren Entity-Typen definiert.

- **Beziehungen höheren Types.**
  Im HERM-Modell ([BOT90], [Tha91b], [Tha91c] und [Tha98]) sind Higher-Order Relationship-Typen möglich. Relationship-Typen können sowohl über Entity-Typen als auch über anderen Relationship-Typen definiert sein. Entity-Typen haben dabei immer die Ordnung 0. Relationship-Typen, die nur über Entity-Typen definiert sind, haben die Ordnung 1. Relationship-Typen, die nur über Entity-Typen und Relationship-Typen der Ordnung 1 definiert sind, haben die Ordnung 2, usw.

- **Spezialisierung, Generalisierung, Partitionierung.**
  Bei diesen Erweiterungen werden spezielle Relationship-Typen eingeführt, die mit der Semantik der Spezialisierung, Generalisierung oder Partitionierung unterlegt sind.

Eine vollständigere Darstellung von verschiedenen Erweiterungen des Entity-Relationship Modells befindet sich in [HeS95].

Für alle *erweiterten Entity-Relationship Modelle* ist die *Semantikakquisition* mit der in dieser Arbeit vorgestellten Methode problemlos möglich, wenn eine Übersetzung dieser Datenmodelle in eine relationale Datenbank erfolgen kann. Die Semantik, die in den ergänzten Konstrukten implizit enthalten ist, muß dabei in lokale und globale Integritätsbedingungen übertragen werden, um bei der Semantikakquisition auswertbar zu sein, sonst gehen diese Informationen verloren.
Es wird an einigen Stellen der Arbeit darauf hingewiesen, wie bestimmte Erweiterungen des Entity-Relationship Modells bei der Semantikakquisition behandelt werden können.

Übersetzung von Datenbanken im Entity-Relationship Modell in relationale Datenbank-Schemata

Das Entity-Relationship Modell ist ein konzeptionelles Datenmodell. Will man bei Datenbanken im Entity-Relationship Modell mit Daten arbeiten, so müssen diese Datenbanken in ein logisches Datenmodell übersetzt werden. Diese Übersetzung ist in einfacher Weise möglich. Auch bei der Semantikakquisition, die als Grundlage Daten verwendet, muß eine Übersetzung von Datenbanken im Entity-Relationship Modell in das relationale Modell erfolgen.

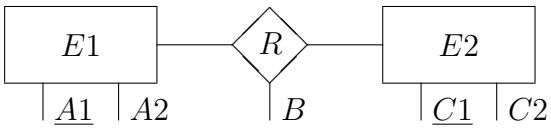
Bei dieser Übersetzung wird jeder Entity-Typ in ein Relationen-Schema übersetzt. Die Schlüsselattribute des Entity-Typen werden dabei als Schlüssel in das Relationen-Schema übernommen. Relationship-Typen werden ebenfalls in eigene Relationen-Schemata übersetzt, in diese werden auch die Schlüsselattribute der zugehörigen Entity-Typen aufgenommen. Die Schlüsselattribute der zugehörigen Entities sind Schlüssel der Relationen-Schemata, die aus den Relationship-Typen entstanden sind. Diese Attribute müssen keine minimalen Schlüssel sein. In [HeS95] ist eine Übersicht zusammengestellt, wie aus diesen Schlüsseln in Abhängigkeit von den Kardinalitäten minimale Schlüssel gebildet werden können.

Diese einfache Übersetzung läßt sich optimieren. Bei der Optimierung können einige der so entstandenen Relationen-Schemata zusammengefaßt werden.

Aus den strukturellen Merkmalen der Datenbanken im Entity-Relationship Modells lassen sich einige lokale und globale Integritätsbedingungen des zugehörigen relationalen Datenbank-Schemas ableiten. Schlüssel der Datenbank im Entity-Relationship Modell werden wie oben bereits beschrieben als lokale Integritätsbedingungen in das relationale Datenbank-Schema übernommen. Bei der Übersetzung entstehen Fremdschlüssel, diese bedingen Inklusionsabhängigkeiten. Damit man diese Informationen nicht verliert, müssen sie in Form von globalen Integritätsbedingungen in dem relationalen Datenbank-Schema ergänzt werden. Das folgende abstrakte Beispiel zeigt, wie das erfolgen kann.

Beispiel 2.1:

Datenbank im Entity-Relationship Modell:



Bei der beschriebenen Übersetzung entstehendes relationales Datenbank-Schema:

$E1$:

$A1$	$A2$

R:

$E1.A1$	$E2.C1$	B

$E2$:

$C1$	$C2$

Dabei werden folgende lokale und globale Integritätsbedingungen in das relationale Datenbank-Schema übernommen:

Lokale Integritätsbedingungen der Relationen-Schemata:
- A_1 - Schlüssel in E_1
- C_1 - Schlüssel in E_2

- $E1.A1$ $E2.C1$ - Schlüssel in R

Globale Integritätsbedingungen der Datenbank-Schemata:
- $R[E_1.A_1] \subseteq E_1[A_1]$
- $R[E_2.C_1] \subseteq E_2[C_1]$

Weiterhin weiß man, daß die Werte der Kardinalitäten $\mathsf{card}(R, E_1)$ und $\mathsf{card}(R, E_2)$ im relationalen Datenmodell bestimmt werden müssen.

Voraussetzung für diese Übersetzung von Datenbanken im Entity-Relationship Modell in relationale Datenbank-Schemata sind *Schlüsselinformationen* der Entity-Typen. Kennt man diese nicht, so kann das Verfahren *schrittweise* durchgeführt werden.

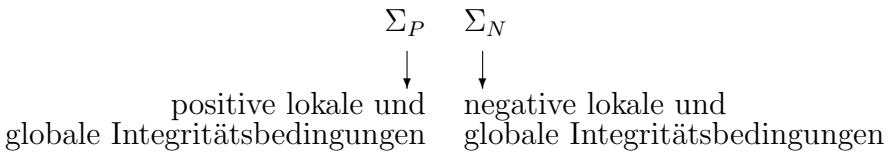
1. Es werden zuerst alle Entity-Typen sowie die Relationship-Typen, zu denen man die Schlüssel der zugehörigen Entity-Typen kennt, übersetzt. Daten zu diesen Teilen der Datenbank werden eingelesen. Es erfolgt dann die Semantiksuche mit der in dieser Arbeit beschriebenen Methode, in deren Ergebnis auch Schlüssel bekannt sind.
2. Ist für jeden Entity-Typen ein Schüssel bekannt, so erfolgt die Übersetzung der übrigen Teile der Datenbank unter Verwendung der Schlüsselinformationen und das Einlesen der Daten für diese. Anschließend wird die Semantikakquisition für diese Teile durchgeführt.

2.2 Vorgehen bei der Semantikakquisition

Während der Semantikakquisition sind die Integritätsbedingungen der Relationen-Schemata bzw. Datenbank-Schemata noch nicht oder nicht vollständig bekannt. Man hat Datenbank-Schemata gegeben, in denen bestimmte Abhängigkeiten gelten müssen. Diese Abhängigkeiten müssen aufgrund der Darstellung der Daten in den Datenbanken bzw. aufgrund von Zusammenhängen in der realen Welt, die auf die Datenbank abgebildet werden, erfüllt sein. Man *sucht* bei der Semantikakquisition diese Abhängigkeiten, also die *Integritätsbedingungen der Relationen-Schemata und Datenbank-Schemata.*

Dabei werden alle verfügbaren Informationen über die Integritätsbedingungen gesucht. Es können dabei *nicht nur positive Informationen* (geltende Schlüssel, geltende funktionale Abhängigkeiten, geltende Inklusionsabhängigkeiten, geltende Exklusionsabhängigkeiten, geltende Kardinalitäten) gefunden werden. Teilweise ist zu ermitteln, daß bestimmte Integritätsbedingungen nicht gelten. Die *Ermittlung von nicht geltenden Integritätsbedingungen* (nicht geltende Schlüssel, nicht geltende funktionale Abhängigkeiten, nicht geltende Inklusionsabhängigkeiten, nicht geltende Exklusionsabhängigkeiten, nicht geltende Kardinalitäten) ist oft sogar einfacher als die Ableitung geltender Integritätsbedingungen.

Während der Phase der Semantikakquisition ermittelt man also zusätzlich zu den geltenden lokalen und globalen Integritätsbedingungen der Relationen-Schemata bzw. Datenbank-Schemata auch die nicht geltenden lokalen und globalen Integritätsbedingungen.



Die *geltenden (positiven) Integritätsbedingungen* Σ_P sind die bekannten oder ableitbaren lokalen Integritätsbedingungen der Relationen-Schemata (I_l) und die bekannten oder ableitbaren globalen Integritätsbedingungen des Datenbank-Schemas (I_g). Diese wurden bereits in Abschnitt 2.1.2 definiert. Die Definition der nicht geltenden Integritätsbedingungen Σ_N erfolgt als *Negation* der Definition der geltenden Integritätsbedingungen. Diese Integritätsbedingungen werden im nächsten Abschnitt definiert.

2.2.1 Definition negierter Integritätsbedingungen in Datenbanken

Negierte funktionale Abhängigkeiten. Geltende funktionale Abhängigkeiten eines Relationen-Schemas sagen aus, daß für alle Instanzen des Relationen-Schemas die entsprechenden funktionalen Abhängigkeiten gelten müssen. Will man diese funktionalen Abhängigkeiten ermitteln, so müßte man alle Instanzen untersuchen. Leichter läßt sich die Information über nicht geltende funktionale Abhängigkeiten ableiten, da diese aus einer Relation abgeleitet werden können. Die Negation der funktionalen Abhängigkeiten soll als eigener Begriff eingeführt werden.

Definition 2.15 *Sei R ein Relationen-Schema, U die Attributmenge von R und $X, Y \subseteq U$. Eine* **negierte funktionale Abhängigkeit (NFD)** *$X \not\rightarrow Y$ gilt in R, wenn es eine Relation r von R gibt, mit: $\exists\, t, t' \in r$ mit $(t[X] = t'[X]) \wedge (t[Y] \neq t'[Y])$.*

Anmerkung 1: Die negierten funktionalen Abhängigkeiten wurden in [Jan89] eingeführt, dort werden sie als funktionale Unabhängigkeiten bezeichnet.
Anmerkung 2: Die Definition der negierten funktionalen Abhängigkeiten unterscheidet sich von der Definition der afunktionalen Abhängigkeiten, die bei der horizontalen Dekomposition verwendet wird. Eine afunktionale Abhängigkeit $X \not\rightarrow Y$ ist in einer Relation erfüllt, wenn es zu jedem Tupel t der Relation ein Tupel t' gibt, das folgende Eigenschaft erfüllt: $(t[X] = t'[X]) \wedge (t[Y] \neq t'[Y])$ (vergleiche auch [Bis96] und [PBG89]).

Negierte Schlüssel. Ebenso wie bei funktionalen Abhängigkeiten ist es auch bei der Schlüsselsuche sinnvoll, Informationen über nicht geltende Schlüssel zu ermitteln und auszuwerten. Diese werden wie folgt definiert:

Definition 2.16 *Sei R ein Relationen-Schema, U die Attributmenge von R und $X \subseteq U$. Ein* **negierter Schlüssel (NKEY)** *X gilt in R, wenn es eine Relation r von R gibt, für die gilt: $\exists\, t, t' \in r, t \neq t'$ mit $t[X] = t'[X]$.*

Negierte Inklusionsabhängigkeiten. Auch bei globalen Integritätsbedingungen ist es leichter, die Negation dieser zu finden als die geltenden Integritätsbedingungen zu ermitteln. In analoger Weise wie bei den funktionalen Abhängigkeiten werden deshalb auch die negierten Inklusionsabhängigkeiten als eigener Begriff eingeführt:

Definition 2.17 *Sei* $EDBS = \{.., R, .., S, ..\}$ *ein einfaches Datenbank-Schema und seien* $R = (U_1, \underline{D_1}, dom_1, I_{l1})$ *und* $S = (U_2, \underline{D_2}, dom_2, I_{l2})$ *zwei Relationen-Schemata mit* $X = \langle X_1..X_n \rangle$ *- Attribute von* U_1 *und* $Y = \langle Y_1..Y_n \rangle$ *- Attribute von* U_2*. Eine* **negierte Inklusionsabhängigkeit (NID)** $R.X \not\subseteq S.Y$ *gilt, wenn es zwei Relationen* r *von* R *und* s *von* S *gibt, für die gilt:* $\exists\, t \in r$ *für das kein Tupel* $t' \in s$ *existiert mit* $t[X_i] = t'[Y_i]$, $\forall\, 1 \leq i \leq n$.

Negierte Exklusionsabhängigkeiten. Auch bei Exklusionsabhängigkeiten wird die Negation der Integritätsbedingungen eingeführt, da diese für die Semantikakquisition benötigt wird.

Definition 2.18 *Sei* $EDBS = \{.., R, .., S, ..\}$ *ein einfaches Datenbank-Schema und seien* $R = (U_1, \underline{D_1}, dom_1, I_{l1})$ *und* $S = (U_2, \underline{D_2}, dom_2, I_{l2})$ *zwei einfache Relationen-Schemata mit* $X = \langle X_1..X_n \rangle$ *- Attribute aus* U_1 *und* $Y = \langle Y_1..Y_n \rangle$ *- Attribute aus* U_2*. Eine* **negierte Exklusionsabhängigkeit (NED)** $R.X \nparallel S.Y$ *ist gültig, wenn es zwei Relationen* r *von* R *und* s *von* S *gibt, für die gilt:* $\exists\, t \in r$ *und* $t' \in s$ *mit* $t[X_i] = t'[Y_i]$ $\forall\, 1 \leq i \leq n$.

Aus den Definitionen der Inklusions- und Exklusionsabhängigkeiten ist zu ersehen, daß zwischen zwei Attributsequenzen nicht gleichzeitig geltende Inklusions- und Exklusionsabhängigkeiten existieren können.
Zwischen zwei Attributsequenzen $R.X$ und $S.Y$ können gleichzeitig folgende Abhängigkeiten für nichtleere Relationen gelten. Aufgrund der Symmetrie der Exklusionsabhängigkeiten sind einige der in der nachfolgenden Aufzählung auftretenden Fälle redundant:

- ($R.X \subseteq S.Y$) und ($R.X \nparallel S.Y$)
- ($R.X \subseteq S.Y$) und ($S.Y \nparallel R.X$)
- ($R.X \mid\mid S.Y$) und ($R.X \not\subseteq S.Y$)
- ($R.X \mid\mid S.Y$) und ($S.Y \not\subseteq R.X$)
- ($R.X \nparallel S.Y$) und ($R.X \not\subseteq S.Y$)
- ($R.X \nparallel S.Y$) und ($S.Y \not\subseteq R.X$)

Gilt zwischen zwei Attributsequenzen eine Inklusionsabhängigkeit, so gilt zwischen diesen Sequenzen für nichtleere Relationen auch immer eine negierte Exklusionsabhängigkeit.

- if ($R.X \subseteq S.Y$) then ($R.X \nparallel S.Y$)

Eine Exklusionsabhängigkeit beinhaltet gleichzeitig auch negierte Inklusionsabhängigkeiten.

- if ($R.X \mid\mid S.Y$) then ($R.X \not\subseteq S.Y$) and ($S.Y \not\subseteq R.X$)

Negierte Kardinalitäten. Ebenso wie man bei der Semantikakquisition geltende Kardinalitäten ermitteln kann, kann man auch feststellen, welche Kardinalitäten nicht gelten.

Definition 2.19 *Seien* $R = (U_1, \underline{D_1}, dom_1, I_{l1})$ *und* $S = (U_2, \underline{D_2}, dom_2), I_{l2}$ *zwei Relationen-Schemata eines Datenbank-Schemas* D *mit* $X = \langle X_1..X_n \rangle \subseteq U_1$, X *ist Schlüssel in* R_1 *und* $Y = \langle Y_1..Y_n \rangle \subseteq U_2$.
Die **Kardinalität** $\mathsf{card}(S, R) = (m, n)$ *ist durch Relationen-Instanzen* r *von* R *und* s *von* S *nicht erfüllt, genau dann wenn es mindestens ein Tupel* t *von* r *gibt, für das nicht mindestens* m *oder maximal* n *Tupel* t' *von* s *existieren, für die gilt:* $\forall\ 1 \leq i \leq n\ t[X_i] = t'[Y_i]$.

Damit die Anzahl der ableitbaren negierten Kardinalitäten nicht unendlich groß wird, ist auch hier eine Einschränkung auf die vier Kardinalitäten $(0, 1)$, $(1, 1)$, $(0, n)$ und $(1, n)$ sinnvoll.

2.2.2 Vorteile der Verwendung geltender und negierter Integritätsbedingungen

Bei der Einführung der negierten Integritätsbedingungen wurde bereits ein Vorteil, der bei der Verwendung dieser auftritt, genannt. Diese Informationen sind oft leichter zu finden als Informationen über geltende Integritätsbedingungen. Es ist auch aus weiteren Gründen nützlich, Negativinformationen über die Semantik einzubeziehen. An dieser Stelle sollen diese Vorteile zusammengefaßt und erläutert werden.

Ableitbarkeit aus Daten. Die negierten Integritätsbedingungen sind durch Auswertung von Daten leichter zu finden, als geltende Integritätsbedingungen. Um negierte Integritätsbedingungen zu finden, muß man in bekannten Daten nur ein Gegenbeispiel der geltende Integritätsbedingungen finden. Es genügt dabei, eine Relation bzw. zwei Relationen einer Datenbank auszuwerten, um nicht geltende Integritätsbedingungen zu finden.

Abbildung 2.2 zeigt, aus welchen Relationen welche Arten von Integritätsbedingungen abgeleitet werden können.

Testdaten im Datenbankentwurf sind in der Regel nur Ausschnitte realer Datenbanken. Die Übersicht zeigt, daß sich negierte Schlüssel, negierte funktionale Abhängigkeiten, negierte Exklusionsabhängigkeiten und Kardinalitäten der Form $\mathsf{card}(R_1, R_2) = (\_, n)$ bereits aus diesen ableiten lassen. Die abgeleiteten Integritätsbedingungen sind *invariant gegenüber Tupelergänzungen*. Sie können also durch das Ergänzen von Tupeln nicht ungültig werden, wenn sie vorher in den Relationen erfüllt waren. Deshalb ist ihre Ableitung bereits aus Ausschnitten von Relationen möglich.

Kennt man *reale Datenbanken, die in sich abgeschlossen sind*, d.h. alle geltenden Inklusionsabhängigkeiten sind auch in der Datenbank erfüllt, so ist hier zusätzlich die Ableitung weiterer Integritätsbedingungen aus den Daten möglich. Man kann keine geltenden Inklusionsabhängigkeiten ableiten, da diese nur zufällig in den Daten erfüllt sein können, man kann aber negierte Inklusionsabhängigkeiten und Kardinalitäten der Form $\mathsf{card}(R_1, R_2)$=(0,_) ableiten. Solche in sich abgeschlossenen Datenbanken sind häufig im Reverse-Engineering vorhanden.

r_1^2	$\Rightarrow$	negierte funktionale Abhängigkeiten	Ableitung aus Ausschnitten von Datenbanken
r_1^2	$\Rightarrow$	negierte Schlüssel	
r_1^1, r_2^1	$\Rightarrow$	negierte Exklusionsabhängigkeiten	
r_1^1, r_2^2	$\Rightarrow$	card(R_1, R_2)=(_,n)	
r_1^1, r_2^m	$\Rightarrow$	negierte Inklusionsabhängigkeiten	Ableitung aus in sich abgeschlossenen Datenbanken
r_1^1, r_2^m	$\Rightarrow$	card(R_1, R_2)=(0,_)	
alle r_1^n	$\Rightarrow$	Schlüssel	keine Ableitung aus Daten möglich, da unendlich viele Relationen ausgewertet werden müßten
alle r_1^n	$\Rightarrow$	funktionale Abhängigkeiten	
alle r_1^n, alle r_2^m	$\Rightarrow$	Inklusionsabhängigkeiten	
alle r_1^n, alle r_2^m	$\Rightarrow$	Exklusionsabhängigkeiten	
alle r_1^n, alle r_2^m	$\Rightarrow$	card(R_1, R_2)=(1,_)	
alle r_1^n, alle r_2^m	$\Rightarrow$	card(R_1, R_2)=(_,1)	

r_1^y, gibt an, daß die Relation r_1 y Tupel enthält, wobei r_1^n eine in sich abgeschlossenen Relation ist,
r_2^z, gibt an, daß die Relation r_2 z Tupel enthält, wobei r_2^m eine in sich abgeschlossenen Relation ist.

Abbildung 2.2: Möglichkeit der Ableitung von Integritätsbedingungen aus Daten

Suchraumeinschränkung. Da negierte Integritätsbedingungen oft leichter ableitbar sind als geltende, erreicht man durch diese eine Einschränkung des Suchraumes zur Ermittlung geltender Integritätsbedingungen. Durch Informationen über negierte Integritätsbedingungen schränkt man die Anzahl der unbekannten und noch zu untersuchenden Integritätsbedingungen zum Teil erheblich ein.

Fehlende Closed-World-Assumption. Die *geltenden und negierten Integritätsbedingungen einer Relation* oder *Datenbank* sind aus der Relation oder Datenbank ersichtlich. Durch die Betrachtung negierter Integritätsbedingungen als Negation der geltenden Integritätsbedingungen gilt hier eine *Closed-World-Assumption.*

Die *lokalen und globalen Integritätsbedingungen der Relationen-Schemata* bzw. *Datenbank-Schemata* sind nicht so einfach ableitbar, da hier alle Relationen der Relationen-Schemata bzw. Datenbanken der Datenbank-Schemata ausgewertet werden müssen. Man kann diese Informationen aber schrittweise ermitteln. Dabei kann man sowohl positive als auch negative Informationen über die Integritätsbedingungen ableiten. Eine *Closed-World Assumption ist während der Semantikakquisition nicht sinnvoll*, da man über keine vollständigen Informationen über die Semantik verfügt. Man möchte eine *dreiwertige Logik* verwenden, um geltende Integritätsbedingungen, negierte Integritätsbedingungen und unbekannte Integritätsbedingungen unterscheiden zu können. Abbildung 2.3 veranschaulicht diese drei Mengen von Integritätsbedingungen.

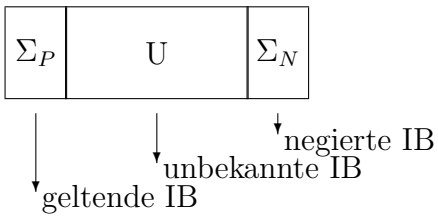


Abbildung 2.3: Integritätsbedingungen eines Relationen- bzw. Datenbank-Schemas

Durch die getrennte Ermittlung geltender und negierten Integritätsbedingungen erreicht man diese Unterscheidung, man kann dadurch vier verschiedene Fälle für jede Integritätsbedingungen σ unterscheiden:

	$\Sigma_N \nvdash \neg\sigma$	$\Sigma_N \vdash \neg\sigma$
$\Sigma_P \nvdash \sigma$	σ ist unbekannt	$\neg\sigma$
$\Sigma_P \vdash \sigma$	σ	Konfliktfall

Eine *Integritätsbedingung gilt* (σ), wenn diese Integritätsbedingung aus der Menge der positiven Integritätsbedingungen und nicht aus der Menge der negativen Integritätsbedingungen ableitbar ist $(\Sigma_P \vdash \sigma) \wedge (\Sigma_N \nvdash \neg\sigma)$.

Eine *Integritätsbedingung gilt nicht* ($\neg\sigma$), wenn diese Integritätsbedingung aus der Menge der negativen Integritätsbedingungen ableitbar ist und nicht aus der Menge der positiven Integritätsbedingungen ableitbar ist $(\Sigma_P \nvdash \sigma) \wedge (\Sigma_N \vdash \neg\sigma)$.

Eine Integritätsbedingung ist *unbekannt*, wenn sie weder aus der Menge der geltenden Integritätsbedingungen noch aus der Menge der nicht geltenden Integritätsbedingungen ableitbar ist $(\Sigma_P \nvdash \sigma) \wedge (\Sigma_N \nvdash \neg\sigma)$.

Bei der Speicherung der geltenden und negierten Integritätsbedingungen zum Erreichen einer dreiwertigen Logik kann auch noch ein vierter Fall, der Konfliktfall, auftreten. Ein *Konflikt* besteht, wenn eine geltende und entsprechende negierte Integritätsbedingung gleichzeitig ableitbar ist $(\Sigma_P \vdash \sigma \wedge \Sigma_N \vdash \sigma)$. Dieser Fall ist für die Semantikakquisition besonders bedeutsam, er wird im folgenden Punkt beschrieben.

Konflikterkennung. Es kann durch die gezeigte getrennte Speicherung positiver und negativer Informationen über die Semantik auch die Möglichkeit bestehen, daß ein *Konflikt* in der Menge der Integritätsbedingungen auftritt, dieser ist durch widersprüchliche Angaben zu den Integritätsbedingungen bedingt. Die *Konfliktfälle* müssen bei der Semantikakquisition *erkannt* und *geklärt* werden. Die durch die getrennte Ermittlung von positiven und negativen Informationen über die Semantik betrachteten Konfliktfälle sind eine zusätzliche *Kontrolle der erkannten Integritätsbedingungen.*

Überprüfung der Vollständigkeit der Integritätsbedingungen. Man kann durch die Auswertung der negierten Integritätsbedingungen neben den geltenden Integritätsbedingungen erkennen, ob noch unbekannte Integritätsbedingungen existieren. Nur wenn das nicht der Fall

ist, kann die *Vollständigkeit der geltenden Integritätsbedingungen* gewährleistet werden.
Die Ermittlung und Auswertung der negierten Integritätsbedingungen bewirkt also ein *Abbruchkriterium* für die Semantikakquisition.

2.2.3 Allgemeines Vorgehen bei der Semantikakquisition

Ziel eines Semantikakquisitionstools ist die Unterstützung des Benutzers bei der Angabe der geltenden Integritätsbedingungen I_l und I_g von Datenbanken. Dazu werden während der Akquisitionsphase geltende und negierte Integritätsbedingungen ermittelt. Es wird schrittweise die Menge der noch unbekannten Integritätsbedingungen untersucht. Abbildung 2.4 veranschaulicht das Vorgehen bei der Semantikakquisition.

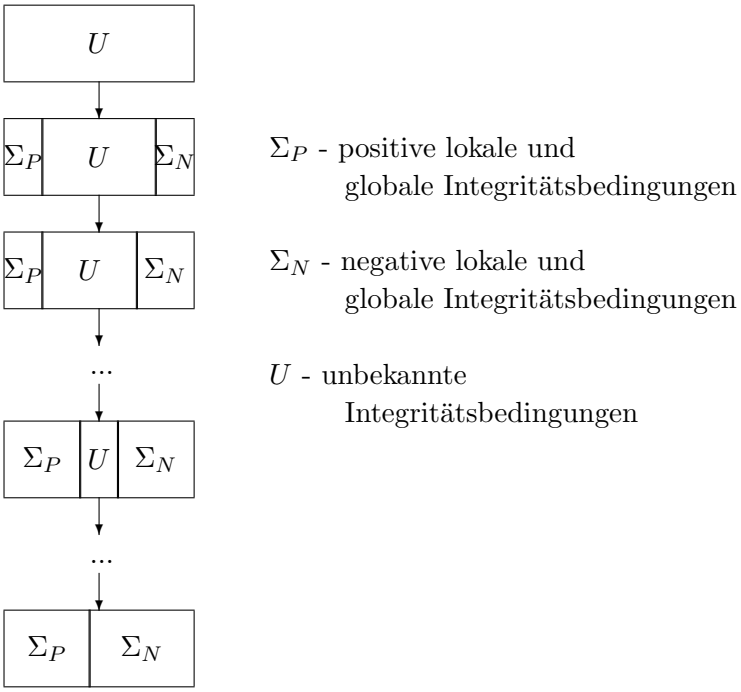


Abbildung 2.4: Positive und negative Informationen über Integritätsbedingungen während der Semantikakquisition

Begonnen wird bei der Semantikakquisition mit Datenbank-Schemata, bei denen keine Integritätsbedingungen bekannt sind. Aus der Menge der unbekannten Integritätsbedingungen werden nacheinander Kandidaten ausgewählt, für die anschließend durch verschiedene Methoden untersucht wird, ob sie geltende Integritätsbedingungen (Σ_P) oder negierte Integritätsbedingungen (Σ_N) sind. Diese Elemente werden in die Mengen (Σ_P) bzw. (Σ_N) ergänzt, sie sind dadurch nicht mehr in der Menge der unbekannten Integritätsbedingungen U enthalten.

Die Mengen Σ_P und Σ_N sind bei diesem Vorgehen *monoton wachsend*. *Ausnahme* ist die explizite Rücknahme von Integritätsbedingungen, diese erfolgt, wenn einmal durch den Benutzer gemachte Angaben von diesem zurückgenommen werden (z.B. eingegebene Beispieldaten wer-

den korrigiert, nach erkannten Konflikten durch widersprüchliche Angaben werden Integritätsbedingungen vom Benutzer gelöscht).

Die Semantikakquisition kann durchgeführt werden, bis es keine unbekannten Integritätsbedingungen mehr gibt ($\nexists\, \sigma : (\Sigma_P \not\vdash \sigma) \wedge (\Sigma_N \not\vdash \neg\sigma)$). Ist das gegeben, so ist die *Semantikakquisition vollständig*. In diesem Fall kann man gewährleisten, daß *alle geltenden Integritätsbedingungen* gefunden wurden.
In sehr großen Datenbanken ist die Anzahl der zu untersuchenden unbekannten Integritätsbedingungen sehr groß. Es kann sein, daß eine *vollständige Semantikakquisition nicht möglich* ist, da dazu zu viele Schritte zur Diskussion unbekannter Integritätsbedingungen ausgeführt werden müssen. In diesem Fall kann nur ein Teil der unbekannten Integritätsbedingungen untersucht und geklärt werden. Man kann deshalb auch *nicht davon ausgehen, daß alle geltenden Integritätsbedingungen gefunden* werden.

Die gefundenen *geltenden Integritätsbedingungen* Σ_P können als lokale und globale Integritätsbedingungen in die Relationen-Schemata bzw. Datenbank-Schemata übernommen werden und für alle Entwurfsaufgaben ausgewertet werden, die in Abschnitt 1.2.5 angegeben wurden.

2.3 Axiomatisierung und Algorithmen zur Ableitung von Integritätsbedingungen

Im vorigen Abschnitt wurde gezeigt, daß geltende und negierte Integritätsbedingungen bei der Semantikakquisition getrennt gespeichert werden. Zur Behandlung dieser Integritätsbedingungen sind Algorithmen zur Ableitung von Integritätsbedingungen aus einer Menge von Integritätsbedingungen notwendig. Mit diesen läßt sich feststellen, ob eine Integritätsbedingung erfüllt ist, die Negation dieser gilt, die Integritätsbedingung noch unbekannt ist oder ein Konflikt besteht. Die Realisierung dieser Algorithmen zur Ableitung von Integritätsbedingungen folgt in diesem Abschnitt.

Funktionale und negierte funktionale Abhängigkeiten. Die Axiomatisierung funktionaler Abhängigkeiten ist allgemein bekannt (u.a. in [Arm74], [Mit83a], [Mit83b], [Ull88], [PDG89], [Tha91a], [VRT82], [BCN92] und [MaR92] dargestellt).

Sei FD eine Menge von funktionalen Abhängigkeiten, f eine funktionale Abhängigkeit, dann gilt $FD \vdash f$, wenn f in FD ist oder durch endliche Anwendung folgender Ableitungsregeln, die auch als *Armstrong-Axiome* bezeichnet werden, auf die Elemente in FD entsteht.

Sei U die Menge der Attribute eines Relationen-Schemas R und $XYZ \subseteq U$.

(F1)	Reflexivität:	if $(Y \subseteq X)$	then	$(X \longrightarrow Y)$
(F2)	Augmentation:	if $(X \longrightarrow Y)$	then	$(XZ \longrightarrow YZ)$
(F3)	Transitivität:	if $((X \longrightarrow Y) \wedge (Y \longrightarrow Z))$	then	$(X \longrightarrow Z)$

Definition 2.20 *: Sei FD eine Menge von funktionalen Abhängigkeiten, dann heißt*

$$FD+ := \{f | FD \vdash f\}$$

die **Hülle der funktionalen Abhängigkeiten**.

Zur Ableitung einer funktionalen Abhängigkeit aus einer Menge von Abhängigkeiten kann auch folgender Algorithmus verwendet werden (ebenfalls angegeben in [Ull88], [PDG89], [Tha91a], [BCN92], [MaR92], in ähnlicher Form in [HeS95] u.a.):

INPUT: (FD) - Menge von funktionalen Abhängigkeiten
$X \longrightarrow C$ - zu überprüfende funktionale Abhängigkeit

OUTPUT: true oder false

$X_0 := \{X\}$;
$i := 1$;
REPEAT
    $X_i := \{A | \exists Y : Y \longrightarrow A \in FD, Y \subseteq X_{i-1}\}$;
    $i := i + 1$;
UNTIL $(C \subseteq X_n)$ OR $(X_n = X_{n+1})$;
IF $(C \subseteq X_n)$ THEN
    { die funktionale Abhängigkeit $X \longrightarrow C$ ist ableitbar }
ELSE
    { $X \longrightarrow C$ ist nicht ableitbar }
END;

Algorithmus 2.1: Ableitung von funktionalen Abhängigkeiten

Negierte funktionale Abhängigkeiten sind ebenfalls axiomatisierbar [Jan89]. Dabei können aus einer Menge von funktionalen und negierten funktionalen Abhängigkeiten alle ableitbaren negierten funktionalen Abhängigkeiten bestimmt werden.

Folgende Axiome werden verwendet:

Sei FD eine Menge von funktionalen Abhängigkeiten, NFD eine Menge von negierten funktionalen Abhängigkeiten, f eine negierte funktionale Abhängigkeit, $(FD \cup NFD) \vdash f$, wenn f in NFD ist oder durch endliche Anwendung folgender Ableitungsregeln auf die Elemente in FD und NFD entsteht.

Sei U die Menge der Attribute eines Relationen-Schemas R und $XYZ \subseteq U$.

(NFD1)	if $((X \longrightarrow Y) \wedge (X \not\longrightarrow Z))$	then	$(Y \not\longrightarrow Z)$
(NFD2)	if $(X \not\longrightarrow Y)$	then	$(X \not\longrightarrow YZ)$
(NFD3)	if $(XZ \not\longrightarrow YZ)$	then	$(XZ \not\longrightarrow Y)$
(NFD4)	if $((X \longrightarrow Z) \wedge (X \not\longrightarrow YZ))$	then	$(X \not\longrightarrow Y)$
(NFD5)	if $((Y \longrightarrow Z) \wedge (X \not\longrightarrow Z))$	then	$(X \not\longrightarrow Y)$

In [Jan89] sind die Ableitungsregeln **(NFD1) - (NFD3)** angegeben, in [Bel95] und [WGS97] sind die anderen Regeln ergänzt worden.

Definition 2.21 *Sei FD eine Menge von funktionalen Abhängigkeiten, NFDeine Menge von negierten funktionalen Abhängigkeiten, dann heißt*

$$(FD \cup NFD)+ := \{ f | (FD \vdash f) \vee ((FD \cup NFD) \vdash f)\}$$

die **Hülle der funktionalen und negierten funktionalen Abhängigkeiten**.

Die Ableitung negierter funktionaler Abhängigkeiten ist auch nach folgendem Axiom, das in [Jan89] angegeben wurde, möglich.
Die negierte funktionale Abhängigkeit $X \not\rightarrow Y$ gilt genau dann, wenn es eine negierte funktionale Abhängigkeit $V \not\rightarrow W$ gibt, für die gilt: $X \subseteq (V+)$ und $W - (V+) \subseteq Y$.

Inklusionsabhängigkeiten. Inklusionsabhängigkeiten sind nach [CFP84] axiomatisierbar. Dazu werden folgende Ableitungsregeln verwendet:

Seien RST Relationen-Schemata eines Datenbank-Schemas.

(I1) Reflexivität
$R[A_1, .., A_n] \subseteq R[A_1, .., A_n]$

(I2) Permutation, Projektion und Redundanz
if $R[A_1, .., A_n] \subseteq S[B_1, .., B_n]$ then $R[A_{i_1}, .., A_{i_k}] \subseteq S[B_{i_1}, .., B_{i_k}]$
für jede Sequenz $i_1..i_k$ von unterschiedlichen Werten {1..k}

(I3) Transitivität
if $(R[A_1, .., A_n] \subseteq S[B_1, .., B_n]) \wedge (S[B_1, .., B_n] \subseteq T[C_1, .., C_n])$ then
$(R[A_1, .., A_n] \subseteq T[C_1, .., C_n])$

Läßt man weiterhin zu, daß in einer Inklusionsabhängigkeit Attributsequenzen auftreten können, bei denen ein Attribut mehrfach auftritt, so gibt es eine weitere Ableitungsregel:

(I4) wenn$R[AB] \subseteq S[CC]$ gilt und eine Inklusionsabhängigkeit σ existiert, die ebenfalls gilt und in der A in Form von $R[.., A, ..]$ auftritt, dann gilt auch σ', wobei σ' aus σ durch Austauschen von einem oder mehreren Vorkommen von A gegen B entsteht.

Es gibt einen Algorithmus zur Ableitung von Inklusionsabhängigkeiten aus einer Menge von Inklusionsabhängigkeiten ([AHV95]), dieser ist jedoch nicht in polynomialer Zeit lösbar. In [KCV83] wird das Ergebnis der Arbeit bereits im Titel genannt. Für unäre Inklusionsabhängigkeiten gibt es einen Algorithmus zur Ableitung dieser in polynomialer Zeit. In [MaR92] wird eine Übersicht gegeben, welche Komplexität das Implikationsproblem für die Inklusionsabhängigkeiten hat. Die Implikation von unären Inklusionsabhängigkeiten ist in linearer Zeit entscheidbar, k-stellige Inklusionsabhängigkeiten ohne wiederholte Attribute sind in polynomialer Zeit entscheidbar. Die Implikation von nichtzirkulären Inklusionsabhängigkeiten ist NP-vollständig.

Funktionale Abhängigkeiten und Inklusionsabhängigkeiten. In [CFP84] sind Beispiele angegeben, wie aus Mengen von funktionalen und Inklusionsabhängigkeiten weitere funktionale Abhängigkeiten und Inklusionsabhängigkeiten ableitbar sind.

Eine vollständige Axiomatisierung von funktionalen Abhängigkeiten und Inklusionsabhängigkeiten ist in [Mit83a], [Mit83b] und [MaR92] angegeben. Diese verwendet die bereits angegebenen Ableitungsregeln **(F1) - (F3)**, sowie **(I1) - (I4)** und folgende weitere Ableitungsregeln:

(F4) Seien X und X', sowie Y und Y' Attributmengen, die jeweils die gleichen Attribute enthalten. Die Darstellung $X \longrightarrow Y$ bezeichnet auch die funktionale Abhängigkeit $X' \longrightarrow Y'$.

(FI1) if $(R[UV] \subseteq S[XY], |X| = |U|) \wedge (S : X \longrightarrow Y)$ then $(R : U \longrightarrow V)$

(FI2) if $(R[UV] \subseteq S[XY]) \wedge (R[UW] \subseteq S[XZ]) \wedge (S : X \longrightarrow Y)$ then $(R[UVW] \subseteq S[XYZ])$

(FI3) if $(R[U] \subseteq S[V]) \wedge (S : V \longrightarrow A)$ then $(R'[UB] \subseteq S[VA])$, wobei B ein neues Attribut ist und $R' = R \cup B$

Der Beweis der Vollständigkeit und Korrektheit der Ableitungsregeln wurde in [Mit83a] und [Mit83b] geführt. Die Ableitbarkeit von funktionalen Abhängigkeiten und Inklusionsabhängigkeiten ist nur entscheidbar, wenn Einschränkungen auf der Menge der Integitätsbedingungen gemacht werden. Dabei gibt es zwei Möglichkeiten.
1. Man kann nur azyklische Inklusionsabhängigkeiten zulassen ([AHV95], [MaR92]). Die Lösung dieser Implikation ist jedoch nur in exponentieller Zeit zu finden.
2. Die Inklusionsabhängigkeiten können auf die Mengen der unären Inklusionsabhängigkeiten beschränkt werden, um die Ableitbarkeit lösen zu können. In [KCV83] wurde gezeigt, daß die Implikation von funktionalen Abhängigkeiten und unären Inklusionsabhängigkeiten in polynomialer Zeit lösbar ist.

Inklusions- und Exklusionsabhängigkeiten. In [CaV83] wurde ein System von Ableitungsregeln für Inklusions- und Exklusionsabhängigkeiten angegeben. Dieses verwendet neben den bereits in diesem Abschnitt angegebenen Ableitungsregeln **(I1)-(I3)** folgende weitere Regeln:

(E1) if $(R[X] \,||\, S[Y])$ then $(S[Y] \,||\, R[X])$

(E2) if $R(A_1..A_n) \,||\, S(B_1..B_n)$ then $R(A_{i_1}..A_{i_k}) \,||\, S(B_{i_1}..B_{i_k})$ für jede Permutation von A und B

(E3) if $R(X) \,||\, R(X)$ then $R(X) \,||\, S(Y)$

(IE1) if $(R(X) \,||\, R(X))$ then $(R(X) \subseteq S(Y))$

(IE2) if $(R[X] \subseteq S[Y] \wedge T[W] \subseteq U[Z] \wedge S[Y] \,||\, U[Z])$ then $(R[X] \,||\, T[W])$

Kardinalitäten. Kardinalitäten sind nicht vollständig axiomatisierbar. Ein System von Kardinalitäten in einer Datenbank ist erfüllbar, wenn es eine Belegung der Datenbank gibt, die jede Kardinalität erfüllt und bei der keine leere Relation existiert. Gibt es unbekannte Kardinalitäten, für die nur ein Wert möglich ist, damit ein erfüllbares System existiert, so kann man die Werte dieser unbekannten Kardinalitäten aus den anderen bekannten Kardinalitäten ableiten. Auf diese Weise lassen sich einige sehr eingeschränkte Ableitungsregeln für Kardinalitäten ableiten. Diese sind in [Tha98] zusammengefaßt.

2.4 Speicherung von Integritätsbedingungen und Basisoperationen zur Semantikakquisition

In diesem Abschnitt wird zunächst erläutert, welche verschiedenen Möglichkeiten es zur Auswahl der zu speichernden Integritätsbedingungen gibt. Die Vor- und Nachteile dieser Möglichkeiten werden bewertet. Anschließend werden Basisoperationen auf der Menge der gespeicherten Integritätsbedingungen erläutert.

2.4.1 Speicherung von Integritätsbedingungen

Es gibt Ableitungsregeln, mit denen Integritätsbedingungen aus anderen Integritätsbedingungen abgeleitet werden können. Es gibt *Algorithmen zur Hüllenbildung* von Integritätsbedingungen und Bildung einer *Basis der Menge der Integritätsbedingungen*, d.h. einer nichtredundanten Menge der Integritätsbedingungen.
Daraus ergibt sich, daß es *verschiedene Möglichkeiten* gibt, welche Integritätsbedingungen während der Akquisitionsphase gespeichert werden. Diese sollen im folgenden erläutert werden, dazu werden Vor- und Nachteile der einzelnen Möglichkeiten aufgezählt.

I Speicherung aller ableitbaren Integritätsbedingungen (Speicherung der Hülle)

Es ist möglich, alle Integritätsbedingungen σ zu speichern, die aus der Menge der ermittelten Integritätsbedingungen ableitbar sind ($\Sigma \vdash \sigma$). Dazu wird nach jeder Änderung der Menge der Integritätsbedingungen die Hülle dieser Menge gebildet und gespeichert.

Vorteile: Die Überprüfung, ob eine Integritätsbedingung ableitbar ist (*Ableitbarkeitsprüfung*) ist sehr einfach, es muß nur untersucht werden, ob die zu überprüfende Integritätsbedingung bereits in der Menge der gespeicherten Integritätsbedingungen vorhanden ist.
Ebenso einfach ist die Konfliktprüfung für eine Abhängigkeit. Es muß dazu nur untersucht werden, ob die Negation dieser Abhängigkeit bereits in der Hülle gespeichert ist.

Nachteile: Die *Anzahl der zu speichernden Integritätsbedingungen* ist sehr groß.
Die (sehr zeitaufwendige) Bildung der Hülle der Integritätsbedingungen muß nach jeder Änderung der Menge der Integritätsbedingungen erfolgen, das heißt nach jeder *Ergänzung* und jedem *Löschen von Integritätsbedingungen.*
Das *Löschen* einer Integritätsbedingung ist sehr aufwendig, da aus der zu löschenden Integritätsbedingung bei der Hüllenbildung weitere Integritätsbedingungen abgeleitet worden sein können. Diese müssen gefunden und ebenfalls gelöscht werden.

II Speicherung einer nichtredundanten Menge von Integritätsbedingungen (Basis der Integritätsbedingungen)

Man kann in einem Tool nur eine Basis der Integritätsbedingungen, also eine nichtredundante Menge von Integritätsbedingungen, speichern. Dazu werden alle Integritätsbedingungen σ, mit der Eigenschaft $(\Sigma - \sigma) \nvdash \sigma$ gespeichert.

Vorteil: Die *Anzahl der zu speichernden Integritätsbedingungen* ist minimal.

Nachteile: Bei der *Ergänzung einer Integritätsbedingung* müssen zeitaufwendige Algorithmen zur Bildung einer Basis durchgeführt werden. Dabei muß festgestellt werden, ob die ergänzte Integritätsbedingung bereits ableitbar ist oder ob andere bereits gespeicherte Integritätsbedingungen jetzt redundant sind, diese müssen dann gelöscht werden.
Die Ableitbarkeits- und Konfliktprüfungen sind ebenfalls aufwendig.
Im Falle des *Löschens einer Integritätsbedingung* können Informationen verloren gehen. Das soll an einem Beispiel veranschaulicht werden.

Beispiel: Es erfolgt die Speicherung der Integritätsbedingungen $A \longrightarrow B, A \longrightarrow C$ und $B \longrightarrow C$, dabei ist die Abhängigkeit $A \longrightarrow C$ redundant und wird deshalb nicht gespeichert. Wird $B \longrightarrow C$ gelöscht, so ist auch $A \longrightarrow C$ nicht mehr ableitbar, diese Abhängigkeit geht also verloren.

Um das zu vermeiden, muß bei dieser Methode neben der Speicherung der Integritätsbedingungen eine umfangreiche Historiespeicherung erfolgen.

III Speicherung der aus Interaktionen ableitbaren Integritätsbedingungen

Es ist möglich, alle Integritätsbedingungen, die bei der Semantikakquisition durch verschiedene Methoden ermittelt und mit dem Benutzer diskutiert werden, in der Form, in der sie abgeleitet wurden, zu speichern.

Vorteile: Die *Anzahl der zu speichernden Integritätsbedingungen* ist relativ gering.
Bei dem *Löschen von Integritätsbedingungen* gehen keine Informationen, die bereits bekannt waren, verloren. Es ist keine Historiekomponente erforderlich.
Beim *Ergänzen einer Integritätsbedingung* sind keine Algorithmen zur Hüllenbildung auszuführen.

Nachteile: Die *Ableitbarkeitsprüfung* und die *Konfliktprüfung* von Integritätsbedingungen sind nicht einfach auszuführen, die Algorithmen zur Bestimmung ableitbarer Integritätsbedingungen müssen dazu ausgeführt werden.

Bei dieser Variante werden bewußt Redundanzen in der Menge der Integritätsbedingungen gehalten, sofern diese durch wiederholte Eingabe gleicher Informationen zur Semantik (aus gleichen oder verschiedenen Quellen) abgeleitet wurde. Diese redundanten Informationen sind notwendig, um durch das Löschen von Integritätsbedingungen keine bereits ermittelten Angaben zur Semantik zu verlieren.

IV Auswahl der Speicherungsmöglichkeit für Integritätsbedingungen

Aufgrund dieser Vor- und Nachteile ist für die Semantikakquisition die dritte Möglichkeit der Speicherung von Integritätsbedingungen zu bevorzugen. Diese ist die *"natürlichste Variante"*. Es werden alle Informationen zur Semantik in der Form gespeichert, in der sie aus Interaktionen mit dem Benutzer oder aus anderen Quellen ableitbar sind. Damit ist der *Verlauf der Semantikakquisition nachvollziehbar*. Die Menge der bekannten Integritätsbedingungen muß nach Änderungen *nicht wiederhergestellt* werden.

V Notwendige zusätzliche Informationen bei der Speicherung von Integritätsbedingungen

Bei der Semantikakquisition werden die Integritätsbedingungen der Relationen-Schemata und Datenbank-Schemata gesucht. Es müssen dazu alle gefundenen Integritätsbedingungen gespeichert werden. Dabei muß nachvollziehbar sein, wie diese Integritätsbedingungen abgeleitet wurden. In dem Fall, daß widersprüchliche Informationen abgeleitet wurden, muß bekannt sein, woher diese Integritätsbedingungen stammen, damit diese mit dem Benutzer diskutiert werden können.

Folgende Informationen sind deshalb für die Semantikakquisition erforderlich:

- Schlüssel und negierte Schlüssel
- funktionale und negierte funktionale Abhängigkeiten
- Inklusions- und negierte Inklusionsabhängigkeiten
- Exklusions- und negierte Exklusionsabhängigkeiten
- Kardinalitäten und negierte Kardinalitäten

Zu diesen Integritätsbedingungen muß jeweils gespeichert sein, aus welchen Informationen diese abgeleitet wurden, z.B. aus den Daten, der Darstellung der Datenbanken in einem konzeptuellen Datenmodell, dem Dialog mit dem Benutzer, usw, damit bei der Rücknahme von Integritätsbedingungen durch den Benutzer auch die Ursache dieser Integritätsbedingungen geändert werden kann. Soll z.B. eine aus den Daten abgeleitete negierte funktionale Abhängigkeit gelöscht werden, so müssen auch die Daten, aus denen die Abhängigkeit abgeleitet wurde, verändert werden.

2.4.2 Basisoperationen

Es gibt bei der Semantikakquisition bestimmte Operationen, die sehr häufig auf der Menge der Integritätsbedingungen ausgeführt werden müssen. Diese werden im folgenden als Basisoperationen bezeichnet. In diesem Abschnitt erfolgt die Aufzählung von Basisoperationen, dabei wird erläutert, was bei diesen Operationen zu beachten ist.

I Ableitbarkeitsprüfung

Um die noch unbekannten Integritätsbedingungen zu bestimmen, müssen Ableitbarkeitsprüfungen durchgeführt werden. Dabei wird überprüft, ob eine Integritätsbedingung σ oder $\neg\sigma$ aus der Menge der bereits bekannten Integritätsbedingungen ableitbar ist.

Zur Bestimmung der Ableitbarkeit von *funktionalen Abhängigkeiten* wird der in Abschnitt 2.3 angegebene Algorithmus verwendet. Die Ableitbarkeit von *negierten funktionalen Abhängigkeiten* erfolgt nach dem in Abschnitt 2.3 angegebenen Axiom und dem Algorithmus zur Ableitung von funktionalen Abhängigkeiten.

Zur Bestimmung der Ableitbarkeit von *Schlüsseln* werden alle aus einem Schlüssel resultierenden funktionalen Abhängigkeiten gebildet, für diese erfolgt die Untersuchung der Ableitbarkeit. Die Schlüsseleigenschaft geht über die funktionalen Abhängigkeiten hinaus, deshalb kann die Ableitung von Schlüsseln nur aus anderen Schlüsseln, die Teilmenge des zu überprüfenden

Schlüssels sind, erfolgen. Bei *negierten Schlüsseln* wird untersucht, ob eine negierte funktionale Abhängigkeit ableitbar ist, die die Attribute des negierten Schlüssels auf der linken Seite enthält.

Für Inklusionsabhängigkeiten, negierte Inklusionsabhängigkeiten, Exklusionsabhängigkeiten, negierte Exklusionsabhängigkeiten und Kardinalitäten ist eine Ableitbarkeitsprüfung nicht vollständig möglich. Dort kann nur die Ableitbarkeit der Integritätsbedingungen soweit möglich und durch die Ableitungsregeln in Abschnitt 2.3 beschrieben untersucht werden.

II Konfliktprüfung

Durch die Sammlung von geltenden und negierten Integritätsbedingungen kann es passieren, daß *widersprüchliche Angaben* zu den Integritätsbedingungen vorhanden sind. Deshalb muß überwacht werden, daß die Menge der gesammelten Integritätsbedingungen in *konsistentem Zustand* gehalten wird. Es wird dazu für alle erkannten oder eingegebenen Integritätsbedingungen geprüft, ob diese konfliktfrei zur Menge der bereits bekannten Integritätsbedingungen sind. Ein *Konflikt* besteht, wenn sowohl eine Integritätsbedingung σ als auch die Negation der Integritätsbedingung $\neg\sigma$ in der Hülle der Abhängigkeiten $(FD \cup FI)+$ enthalten ist. Tritt ein Konflikt auf, so muß eine der Integritätsbedingungen, die zur Ableitung von σ oder eine der Integritätsbedingung, die zur Ableitung von $\neg\sigma$ führten, falsch sein.

Um Konflikte zu erkennen, wird deshalb vor der Abspeicherung einer Integritätsbedingung untersucht, ob durch diese ein Konflikt entsteht. Ist das der Fall, so existiert ein Konflikt zwischen der Menge der bereits bekannten Integritätsbedingungen und der zu ergänzenden Integritätsbedingung, der gelöst werden muß.

Die *automatische Konfliktlösung* ist aus folgenden Gründen *nicht empfehlenswert*:

1. Bei der automatischen Lösung von Konflikten müßte man *Prioritäten* festlegen, welche Angaben zur Semantik mit größerer Wahrscheinlichkeit richtig sind als andere Angaben, z.B. Beispieldaten können eher Fehler beinhalten als explizite Angaben zu den Integritätsbedingungen. Für einige Anwendungen sind diese Angaben leicht zu machen. Allgemeine Aussagen dazu zu treffen, ist jedoch kompliziert und kann zu falschen Integritätsbedingungen führen, da durch falsche Prioritäten richtige Integritätsbedingungen durch falsche Angaben korrigiert werden können.

2. Bei einer automatischen Konfliktlösung müßte man entscheiden, wie man *Konflikte zwischen Integritätsbedingungen*, die aus der *gleichen Quelle* abgeleitet wurden, z.B. explizite Angaben des Benutzers, löst. Man könnte in diesem Fall die Entscheidung treffen, daß ältere Angaben eher fehlerbehaftet sind als aktuelle Angaben, eine automatische Auflösung solcher Konflikte führt aber nicht in jedem Fall zum gewünschten Ergebnis.

3. *Konflikte* in der Menge der Integritätsbedingungen *resultieren aus Fehlern*. Durch die Konflikterkennung besteht die Möglichkeit, Fehler zu erkennen. Diese Möglichkeit würde man bei einer automatischen Konfliktlösung verlieren. Durch die automatische Lösung von Konflikten können die richtigen Angaben zurückgenommen werden und die fehlerhaften erhalten bleiben.

4. Die automatische Lösung von Konflikten kann *Akzeptanzprobleme* mit sich bringen, es ist in einem Tool nicht vertretbar, daß von einem Benutzer gemachte Angaben automatisch zurückgenommen werden und der Benutzer keinen Einfluß darauf hat.

Aus diesem Grund ist eine Konfliktlösung *nur im Dialog mit dem Benutzer* sinnvoll. Ein *Konflikt* ist ein *Ausnahmefall* bei der Semantikakquisition, dieser weist darauf hin, daß fehlerhafte Eingaben vorliegen. Der erkannte Konflikt ist Voraussetzung für die *Beseitigung dieser Fehler*.

Zur Beseitigung eines Konfliktes werden dem Benutzer die Integritätsbedingungen aufgelistet, zwischen denen ein Konflikt besteht und es wird angegeben, woher diese abgeleitet wurden, z.B. aus den Daten, aus dem Dialog, aus Schlüsseln usw. Der Benutzer muß entscheiden, welche der Angaben falsch ist. Diese wird dann gelöscht, sodaß die Menge der Integritätsbedingungen immer konsistent gehalten wird.

III Ergänzen von Integritätsbedingungen

Es werden nur Integritätsbedingungen gespeichert, die keinen Konflikt hervorrufen, dadurch kann die Menge der gesammelten Integritätsbedingungen immer in konsistentem Zustand gehalten werden.

Vor einer Speicherung einer Integritätsbedingung wird deshalb eine Konfliktprüfung durchgeführt. Existiert kein Konflikt zwischen der Menge der bereits ermittelten Integritätsbedingungen und der zu ergänzenden Integritätsbedingung, so kann die neue Integritätsbedingung in die Menge der bekannten Integritätsbedingungen aufgenommen werden.

IV Löschen von Integritätsbedingungen

Eine Rücknahme von Integritätsbedingungen kann erfolgen, wenn

- Integritätsbedingungen explizit vom Benutzer gelöscht werden oder
- Integritätsbedingungen nach einem Konflikt vom Benutzer als falsch ausgewiesen werden und deshalb gelöscht werden.

Beim Löschen von Integritätsbedingungen ist zu beachten, daß die Integritätsbedingungen nicht nur aus der Menge der bekannten Integritätsbedingungen entfernt werden, sondern in einigen Fällen auch weitere Aktionen folgen müssen. Diese werden in den anschließenden Punkten aufgezählt:

- Die zu löschende Integritätsbedingung wurde aus Daten abgeleitet:
  In diesem Fall weiß man durch die Rücknahme der Integritätsbedingung, daß die Daten falsche Angaben enthielten. Der Benutzer muß zur Änderung der Daten aufgefordert werden, anschließend müssen aus den Daten erneut die ableitbaren Integritätsbedingungen ermittelt werden.
- Die zu löschende Integritätsbedingung wurde aus Datenbank-Statistik-Angaben abgeleitet:
  In diesem Fall weiß man, daß die Angaben zur Tupelanzahl und der Anzahl der verschiedenen Werte der Attribute falsch waren. Der Benutzer muß zur Änderung der Datenbank-Statistik aufgefordert werden, anschließend müssen aus den Angaben zur Datenbank-Statistik wieder die ableitbaren Integritätsbedingungen ermittelt werden.

- Die gelöschte Integritätsbedingung ist eine funktionale Abhängigkeit, die aus einem Schlüssel abgeleitet wurde:
  In diesem Fall muß der gesamte Schlüssel und alle aus diesem resultierenden funktionalen Abhängigkeiten gelöscht werden.
  Beispiel: Für ein Relationen-Schema $R = (A, B, C)$ ist ein Schlüssel A bekannt. Aus diesem werden die funktionalen Abhängigkeiten $A \longrightarrow B$ und $A \longrightarrow C$ abgeleitet. Aufgrund eines Konfliktes soll die Abhängigkeit $A \longrightarrow B$ gelöscht werden. Es muß deshalb auch der Schlüssel A und die funktionale Abhängigkeit $A \longrightarrow C$ zurückgenommen werden.

- Die gelöschte Integritätsbedingung resultiert aus einer expliziten Eingabe des Benutzers:
  In diesem Fall muß die gesamte explizite Eingabe und alle daraus abgeleiteten Integritätsbedingungen gelöscht werden.
  Beispiel: Für ein Relationen-Schema $R = (A, B, C, D)$ ist eine funktionale Abhängigkeit $AB \longrightarrow CD$ eingegeben worden. Aufgrund eines Konfliktes soll die daraus abgeleitete Abhängigkeit $AB \longrightarrow D$ gelöscht werden. Aufgrund dessen muß auch die explizit angegebene Integritätsbedingung $AB \longrightarrow CD$ und die ebenfalls daraus abgeleitete funktionale Abhängigkeit $AB \longrightarrow C$ zurückgenommen werden.

Die hier vorgestellten Basisoperationen werden bei der Semantikakquisition, die in den folgenden Kapiteln beschrieben wird, verwendet.

Kapitel 3

Auswertung von Daten und Datenbank-Statistik

Nachdem in Kapitel 2 die theoretischen Grundlagen der Arbeit erläutert wurden und die Integritätsbedingungen definiert und axiomatisiert wurden, sollen in den folgenden Kapiteln Methoden zur Ableitung von Integritätsbedingungen vorgestellt werden.

Inhalt dieses Kapitels ist die Ableitung von Integritätsbedingungen aus *Daten* und *Datenbank-Statistik-Angaben* ohne Bestätigung durch den Benutzer. Setzt man keine Closed-World-Assumption voraus, so sind aus *Daten und Datenbank-Statistik* nur *negierte Integritätsbedingungen* Σ_N ableitbar.

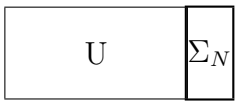


Abbildung 3.1: Menge der durch Auswertung von Daten und Datenbank-Statistik ableitbaren Integritätsbedingungen

Abbildung 3.1 zeigt die Menge der unbekannten Integritätsbedingungen eines Relationen-Schemas bzw. Datenbank-Schemas, sowie die negierten Integritätsbedingungen, die durch die in diesem Kapitel vorgestellten Verfahren ermittelt werden.

3.1 Literaturüberblick

Es gibt Veröffentlichungen, in denen die Ableitung von Informationen zur Semantik durch Auswertung von Daten beschrieben wird.

Die Ableitung *nicht geltender funktionaler Abhängigkeiten* aus *Daten* einer Relation wurde von Bouzeghoub, Gardarin, Metais in [BGM85] und [BoG86] beschrieben.
Castellonos, Saltor entwickelten in [CaS93] effizientere Algorithmen zur Ableitung von nicht geltenden funktionalen Abhängigkeiten aus Daten.

Ein inkrementelles Vorgehen zur Ableitung nicht geltender funktionaler Abhängigkeiten und nicht geltender Schlüssel aus Daten wurde von Orlowski in [Orl92] gezeigt.
In den Arbeiten von Orlowski [Orl92] und Castellonos, Saltor [CaS93] wurden Heuristiken zur Ableitung von funktionalen Abhängigkeiten aus Relationen entwickelt. Dabei wird eine Closed-World-Assumption angenommen, alle Abhängigkeiten, die nicht durch die Einträge der Relation widerlegt sind, gelten in der Relation. Die verwendeten Algorithmen sind trotz des Einsatzes von Heuristiken sehr aufwendig ($O(2^n)$), da die Verfahren für alle konstruierbaren funktionalen Abhängigkeiten überprüfen, ob diese in den bekannten Daten erfüllt sind.

Die Ableitung funktionaler Abhängigkeiten aus Daten wird hier aufgegriffen, aufgrund der eingeführten Definition der negierten funktionalen Abhängigkeiten kann die Methode vereinfacht werden. Ebenso sind negierte Schlüssel in einfacher Weise aus Daten ableitbar. Auch dieses wird im folgenden vorgestellt. Die Ableitung von *negierten Exklusionsabhängigkeiten* und *Kardinalitäten* aus Daten, sowie unter bestimmten Umständen auch die Ableitung *negierter Inklusionsabhängigkeiten* ist in ähnlicher Form möglich. Diese Methoden sind aus der Literatur nicht bekannt, da sich die meisten Veröffentlichungen nicht mit diesen Arten von Integritätsbedingungen beschäftigen. In diesem Kapitel wird die Ableitung dieser Integritätsbedingungen aus vorhandenen Daten erläutert, Algorithmen und Beispiele werden dazu angegeben.

Weiterhin wird in diesem Kapitel die Möglichkeit, *Datenbank-Statistik* zur Ableitung negierter Integritätsbedingungen zu nutzen, gezeigt. Hainaut zeigte in [Hai96], wie aus diesen Angaben und Integritätsbedingungen weitere Angaben zur Datenbank-Statistik ableitbar sind. Man kann jedoch auch aus der Datenbank-Statistik Informationen über Integritätsbedingungen ableiten. Hierfür werden in Abschnitt 3.3 ebenfalls Algorithmen und Beispiele angegeben.

3.2 Auswertung von Daten

Aus vorhandenen Daten lassen sich bestimmte semantische Zusammenhänge ableiten. Man sucht dabei nach Merkmalen in den Relationen bzw. Datenbanken, die Rückschlüsse auf die Integritätsbedingungen der zugehörigen Relationen-Schemata bzw. Datenbank-Schemata zulassen.

Bereits aus Testdaten, die in der Regel nur ein sehr kleiner Ausschnitt realer Datenbanken sind, lassen sich Integritätsbedingungen der Relationen- bzw. Datenbank-Schemata ableiten. Im nächsten Abschnitt wird gezeigt, welche Integritätsbedingungen sich dabei aus welchen Daten ableiten lassen.

3.2.1 Ableitung von Integritätsbedingungen aus unvollständigen Daten

Aus vorhandenen *unvollständigen Datenbanken* lassen sich negierte Schlüssel, negierte funktionale Abhängigkeiten, negierte Exklusionsabhängigkeiten und Kardinalitäten der Form $\mathsf{card}(\_,\_)=(\_,n)$ ableiten.

I Negierte Schlüssel

Die Ableitung von negierten Schlüsseln aus vorhandenen Daten einer Relation ist möglich, sofern es in der Relation Attribute gibt, bei denen mehrfach gleiche Werte auftreten. Wenn in einer Relation r in zwei Tupeln der gleiche Wert bei einem oder mehreren Attributen vorkommt, so kann man ableiten, daß diese Attribute kein Schlüssel des Relationen-Schemas R sein können, da sie bereits in dieser einen bekannten Relation r nicht alle Tupel identifizieren können. Diese Ableitung negierter Schlüssel aus Daten soll zunächst an einem Beispiel erläutert werden.

Beispiel 3.1: Gegeben sei die folgende Relation:

Personen:

Personennummer	Nachname	Vorname	Wohnort	PLZ	Straße	Nummer
271263	Meier	Klaus	Rostock	18119	Kirchenstr.	20
218263	Meier	Ilona	Berlin	10249	Mollstr.	149
948547	Mueller	Karl	Berlin	12621	Mittelweg	281
323983	Schmidt	Klaus	Rostock	18055	Gerberbruch	30
239283	Weber	Peter	Rostock	18055	Gerberbruch	15
043984	Lehmann	Iris	Rostock	18055	Grubenstr	2

Aus dieser Relation läßt sich ableiten, daß folgende Attributmengen keine Schlüssel darstellen:

- {Nachname}
- {Vorname, Wohnort}
- {Wohnort, PLZ, Straße }

Weiterhin können Teilmengen dieser angegebenen drei Mengen (z.B. {Vorname}, {Wohnort, PLZ}, usw.) keine Schlüssel sein, diese negierten Schlüssel lassen sich aus den oberen ableiten.

Der folgende Algorithmus kann zur Ableitung der negierten Schlüssel verwendet werden. Dieser Algorithmus setzt für eine einfachere Darstellung voraus, daß die Menge der Tupel einer Relation durchnumeriert wird.

INPUT: Relation r
OUTPUT: Menge der negierten Schlüssel von r

$M := \emptyset$;
FOR i:=1 TO Tupelanz_r - 1 DO
    FOR j:= i TO Tupelanz_r DO
        (* In der Menge m werden die Attribute gesammelt,
           die gleiche Einträge in t_i und t_j haben *)
        m:= $\emptyset$;
        FOR k:= 1 TO Attributanz_r DO
            IF ($t_i[A_k] = t_j[A_k]$) THEN
                $m := m \cup A_k$;
            END;
        END;
        (* In der Menge M werden die Attributmengen gesammelt,
           bei denen gleiche Einträge in der Relation auftreten *)
        IF ($m \neq \emptyset$) AND ($\nexists\, m' \in M$ mit $m = m'$) THEN

```
                M := M ∪ m;
            END;
        END;
    END;
    FOR ALL m ∈ M DO
        IF ∄ m' ∈ M mit m ⊂ m' THEN
            ⟨ Abspeicherung der Attributmenge in m als negierter Schlüssel ⟩
        END;
    END;
```

Algorithmus 3.1: Ableitung negierter Schlüssel aus einer Relation

Aus einer gegebenen Relation wird durch diesen Algorithmus die Menge der negierten Schlüssel dieser Relation abgeleitet. Durch den letzten Teil des Algorithmus wird gesichert, daß diese Menge redundanzfrei ist.

In dem Algorithmus müssen $\frac{m^2-m}{2} * n$ Vergleichsoperationen durchgeführt werden, wobei n die Anzahl der Attribute ist und m die Anzahl der Tupel der Relation ist. Die Anzahl der notwendigen Vergleichsoperationen des Verfahrens ist unabhängig von den Daten der Relation und den Schlüsseln des zugehörigen Relationen-Schemas.

Inkrementelles Vorgehen

Die *Eingabe und Auswertung von Daten* kann ein *iterativer Prozeß* sein. Es ist deshalb sinnvoll, den Algorithmus daraufhin anzupassen, sodaß nur neu hinzugekommene Tupel einer Relation ausgewertet werden. Werden zu bereits ausgewerteten Relationen weitere Tupel hinzugefügt, so muß die Ableitung negierter Schlüssel nur über diesen ergänzten Tupeln erfolgen. Der beschriebene Algorithmus kann dazu entsprechend angepaßt werden.

```
INPUT: Relation r mit ergänztem Tupel t_neu
       Menge der ableitbaren negierten Schlüssel aus r ohne t_neu
OUTPUT: Menge der negierten Schlüssel von r

M:= {Menge der ableitbaren Schlüssel aus r ohne t_neu}
FOR i:=1 TO Tupelanz_r DO
    (* In der Menge m werden die Attribute gesammelt,
       die gleiche Einträge in t und t_neu haben *)
    m:= ∅;
    FOR k:= 1 TO Attributanz_r DO
        IF (t_i[A_k] = t_neu[A_k]) THEN
            m := m ∪ A_k;
        END;
    END;
    IF (m ≠ ∅) AND (∄ m' ∈ M mit m = m') THEN
        M := M ∪ m;
    END;
```

END;
FOR ALL $m \in M$ DO
    IF $\nexists\, m' \in M$ mit $m \subset m'$ THEN
        $\langle$ Abspeicherung der Attributmenge in m als negierter Schlüssel $\rangle$
    END;
END;

Algorithmus 3.2: Ableitung negierter Schlüssel aus einer Relation und einem Tupel

Bei Ausführung des Algorithmus sind $m * n$ Vergleichsoperationen durchzuführen, wobei m die Tupelanzahl der Relation (vor der Ergänzung des Tupels) ist und n die Anzahl der Attribute.

II Negierte funktionale Abhängigkeiten

Die Ableitung von negierten funktionalen Abhängigkeiten aus einer Relation ergibt sich direkt aus der Definition der negierten funktionalen Abhängigkeiten. Tritt in einer Relation r der Fall auf, daß in einer Attributsequenz bei zwei Tupeln gleiche Einträge stehen, so läßt sich ableiten, daß zu allen Attributen, bei denen in diesen Tupeln zwei verschiedene Einträge vorhanden sind, negierte funktionale Abhängigkeiten gelten. Die negierten funktionalen Abhängigkeiten einer Relation sagen aus, daß in dem Relationen-Schema die entsprechende funktionale Abhängigkeit nicht gelten kann.
Für das Beispiel 3.1 läßt sich folgende nichtredundante Menge von negierten funktionalen Abhängigkeiten ableiten:

- Nachname $\not\rightarrow$ Personennummer
- Nachname $\not\rightarrow$ Vorname
- Nachname $\not\rightarrow$ Wohnort
- Nachname $\not\rightarrow$ PLZ
- Nachname $\not\rightarrow$ Straße
- Nachname $\not\rightarrow$ Nummer
- Vorname Wohnort $\not\rightarrow$ Personennummer
- Vorname Wohnort $\not\rightarrow$ Nachname
- Vorname Wohnort $\not\rightarrow$ PLZ
- Vorname Wohnort $\not\rightarrow$ Straße
- Vorname Wohnort $\not\rightarrow$ Nummer
- Wohnort PLZ Straße $\not\rightarrow$ Personennummer
- Wohnort PLZ Straße $\not\rightarrow$ Nachname
- Wohnort PLZ Straße $\not\rightarrow$ Vorname
- Wohnort PLZ Straße $\not\rightarrow$ Nummer
- Wohnort PLZ $\not\rightarrow$ Straße

Ein Algorithmus kann eingesetzt werden, um diese negierten funktionalen Abhängigkeiten zu ermitteln. Dabei werden paarweise alle Tupel der Relation betrachtet und gemeinsame Einträge in den Tupeln ermittelt. Treten in zwei Tupeln gleiche Einträge bei einer Attributsequenz auf, so lassen sich negierte funktionale Abhängigkeiten zu allen Attributen ableiten, die verschiedene Einträge in diesen Tupeln haben.

INPUT: Relation r
OUTPUT: Menge der negierten funktionalen Abhängigkeiten von r

$M := \emptyset$;
FOR i:=1 TO Tupelanz_r -1 DO
    FOR j:= i TO Tupelanz_r DO
        (* In der Menge m werden die Attribute gesammelt,
           die gleiche Einträge in t_i und t_j haben *)
        m:= $\emptyset$;

```
            FOR k := 1 TO Attributanz_r DO
                IF (t_i[A_k] = t_j[A_k]) THEN
                    m := m ∪ A_k ;
                END;
            END;
            IF (m ≠ ∅) AND (∄ m' ∈ M mit m = m') THEN
                M := M ∪ m;
            END;
        END;
    END;
    FOR ALL m ∈ M DO
        FOR k := 1 TO Attributanz_r DO
            IF (A_k ∉ m) AND (∄ m' ∈ M mit m ⊂ m' und A_k ∉ m') THEN
                ⟨ Bilden von rechtsminimalen negierten funktionalen Abhängigkeiten
                  mit m auf der linken Seite und A_k auf der rechten Seite
                  und Abspeicherung dieser ⟩
                END;
            END;
        END;
    END;
```

Algorithmus 3.3: Ableitung negierter funktionaler Abhängigkeiten aus Daten einer Relation

In dem Algorithmus müssen $\frac{m^2-m}{2}$ Vergleichsoperationen durchgeführt werden, wobei n die Anzahl der Attribute ist und m die Anzahl der Tupel der Relation ist. Die Anzahl der notwendigen Vergleichsoperationen des Verfahrens ist unabhängig von den Daten der Relation und den abgeleiteten negierten funktionalen Abhängigkeiten des Relationen-Schemas.

Inkrementelles Vorgehen

Bei einer Ergänzung der Relationen um weitere Tupel ist es für negierte funktionale Abhängigkeiten ebenfalls möglich, die neu hinzugekommenen negierten funktionalen Abhängigkeiten nur über den ergänzten Teilen abzuleiten. Das Vorgehen ist analog der inkrementellen Ableitung negierter Schlüssel, die aus Seite 57 beschrieben wurde.

Der beschriebene Algorithmus kann dazu entsprechend angepaßt werden. Es werden nur gleiche Werte zwischen den Tupeln der Relation und einem ergänzten Tupel gesucht.

In dem Algorithmus werden $m*n$ Vergleichsoperationen durchgeführt, wobei m die Tupelanzahl der bereits ausgewerteten Relation ist und n die Anzahl der Attribute.

Kombination der Algorithmen zur Ableitung negierter Schlüssel und negierter funktionaler Abhängigkeiten

Die *Algorithmen* zur *Ermittlung negierter Schlüssel* und *negierter funktionaler Abhängigkeiten* können sehr einfach miteinander *kombiniert* werden, da in beiden Fällen die gleichen Vergleichsoperationen durchgeführt werden müssen.

III Negierte Exklusionsabhängigkeiten

Treten in zwei Relationen einer Datenbank Attribute oder Attributsequenzen auf, in denen gleiche Einträge stehen, so hat man nach Definition 2.18 negierte Exklusionsabhängigkeiten zwischen diesen Attributsequenzen in den Relationen gefunden. Die Ableitung negierter Exklusionsabhängigkeiten zwischen zwei Relationen r_1 und r_2 kann nach folgendem Algorithmus erfolgen:

```
INPUT: Relation r1 und Relation r2
OUTPUT: Menge der negierten Exklusionsabhängigkeiten über r1 und r2

FOR i:= 1 TO Tupelanz_r1 DO
    FOR j:= 1 TO Tupelanz_r2 DO
        ⟨ negierte Exklusionsabhängigkeit NED
          mit leerer linker und rechter Seite initialisieren ⟩
        FOR k:= 1 TO Attributanz_r1 DO
            FOR l:= 1 TO Attributanz_r2 DO
                IF (t_i[A_l] = t_j[A_k]) THEN
                    ⟨ A_l in die linke und A_k in die rechte Seite
                      von NED aufnehmen ⟩
                END;
            END;
        END;
        IF ⟨ NED nicht leer ⟩ THEN
            ⟨ Abspeicherung von NED,
              sofern diese noch nicht vorhanden ⟩
        END;
    END;
END;
```

Algorithmus 3.4: Ableitung negierter Exklusionsabhängigkeiten aus zwei Relationen

Nach Ausführung des Algorithmus erfolgt die Beseitigung von Redundanzen, es muß überprüft werden, ob abgeleitete negierte Exklusionsabhängigkeiten auch aus anderen negierten Exklusionsabhängigkeiten, die aus den Daten ermittelt wurden, abgeleitet werden können. Ist das der Fall, so werden diese gelöscht.

Für den vorgestellten Algorithmus ist folgende Anzahl von Vergleichsoperationen erforderlich:
(Tupelanz_$r1$ * Tupelanz_$r2$ * Attributanz_$r1$ * Attributanz_$r2$).
Dieses *Verfahren* läßt sich noch *effektivieren*, indem die Suche von negierten Exklusionsabhängigkeiten nur auf *Attributpaaren mit gleichen Typen und gleichen oder ähnlichen Längen* erfolgt. Diese werden zu Beginn des Vorgehens ermittelt, nur über diesen Attributpaaren erfolgt der Vergleich der Werte.

Inkrementelles Vorgehen

Auch dieser Algorithmus ist für ergänzte Tupel ausführbar, dabei soll ohne Beschränkung der Allgemeinheit in der Relation r_2 ein Tupel ergänzt worden sein. Der Algorithmus wird dann

entsprechend angepaßt. Es werden gleiche Einträge über den Tupeln der Relation r_1 und dem ergänzten Tupel der Relation r_2 ermittelt. Können solche gefunden werden, so erfolgt die Abspeicherung der negierten Exklusionsabhängigkeiten in analoger Form.

IV Kardinalitäten card(_,n)

Für Relationen-Schemata R_1 und R_2 kann aus bestimmten Beispielrelationen r_1 von R_1 und r_2 von R_2 abgeleitet werden, daß die Kardinalität $\mathsf{card}(R_2, R_1) \neq (\_, 1)$ ist. Voraussetzung dazu ist, daß in R_2 ein *Fremdschlüssel* aus R_1 existiert. Findet man in einer Relation r_1 bei dem Schlüssel einen Eintrag, der in r_2 zwei- oder mehrmals als Fremdschlüssel auftritt, so kann man ableiten, daß die Kardinalität $\mathsf{card}(R_2, R_1) \neq (\_,\mathsf{1})$ gilt. Aufgrund der Einschränkung der Kardinalitäten auf die Maximum-Werte 1 und n bedeutet diese Negation, daß die Kardinalität $\mathsf{card}(R_2, R_1) = (\_, n)$ erfüllt ist.

Beispiel 3.2:

Student:

Matrikelnr	Name	Vorname	Hauptfach	Fachsemester
12273	Schulz	Peter	Mathematik	1
37623	Meier	Iris	Medizin	5
12884	Schmidt	Maria	Biologie	7

Vorlesungsbelegung:

Student_Matrikelnr	Fach	Semester
...	...	...
37623	Anatomie	WS96
37623	Biologie	WS96
...	...	...

Aus diesem Beispiel folgt, daß die Kardinalität $\mathsf{card}(Vorlesungsbelegung, Student) = (\_, 1)$ nicht auftreten kann.

Der Algorithmus ist dabei ähnlich dem Algorithmus zur Ableitung der anderen negierten Integritätsbedingungen. Folgende Algorithmusidee liegt dem zugrunde:
Wird die Kardinalität $\mathsf{card}(R_2, R_1)$ durch Auswertung von Relationen r_1 von R_1 und r_2 von R_2 untersucht, so wird für jedes Tupel in r_1 untersucht, ob der Wert des Schlüssels in r_2 als Fremdschlüssel mehrfach auftritt. Kann so ein Fall gefunden werden, so weiß man, daß $\mathsf{card}(R_2, R_1) = (\_,\mathsf{n})$ gilt.

Im Datenbank-Entwurf, in dem nur einige Testdaten bekannt sind, lassen sich diese angegebenen Integritätsbedingungen der Relationen- bzw. Datenbank-Schemata (negierte Schlüssel, negierte funktionale Abhängigkeiten, negierte Exklusionabhängigkeiten und Kardinalitäten der Form $\mathsf{card}(\_, \_) = (\_, n)$) aus den Daten ableiten.

Die Integritätsbedingungen, für die die Ableitung aus Daten gezeigt wurde, sind invariant gegenüber Tupelergänzungen, d.h. Integritätsbedingungen, die aus einer Relation oder Datenbank abgeleitet werden könnten, sind auch nach Ergänzung von Tupeln weiterhin aus der Relation bzw. Datenbank ableitbar. Daher sind diese Arten von Integritätsbedingungen aus Ausschnitten realer Datenbanken ableitbar.

3.2.2 Ableitung von Integritätsbedingungen aus abgeschlossenen Datenbanken

Es gibt weitere negierte Integritätsbedingungen, die jedoch nicht invariant gegenüber Tupelergänzungen sind. Diese Integritätsbedingungen sind ebenfalls aus Daten ableitbar, wenn man über vollständige Datenbanken verfügt (in sich abgeschlossene oder reale Datenbanken).
Datenbanken sind in sich abgeschlossen, wenn für alle Attributeinträge, für die Inklusionsabhängigkeiten gelten, diese auch in den Daten erfüllt sind. Diese Bedingung ist bei realen Datenbanken, die im Reverse-Engineering oft zur Verfügung stehen, erfüllt, bei Testdaten kann man jedoch nicht davon ausgehen.

Kennt man *in sich abgeschlossene Datenbanken*, wie das meist im Reverse-Engineering der Fall ist, so sind aus diesen zusätzlich zu den in Abschnitt 3.2.1 beschriebenen Integritätsbedingungen auch negierte Inklusionsabhängigkeiten und Kardinalitäten $\mathsf{card}(\_,\_) = (0,\_)$ ableitbar.

V Negierte Inklusionsabhängigkeiten

Über allen Attributpaaren mit gleichen Typen und gleichen oder ähnlichen Längen kann man versuchen, negierte Inklusionsabhängigkeiten abzuleiten. Negierte Inklusionsabhängigkeiten gelten nach Definition 2.17 zwischen zwei Attributen, wenn Werte eines Attributes nicht vollständig in der Menge der Werte des anderen Attributes auftreten.

Folgender Algorithmus wird zur Ableitung negierter Inklusionsabhängigkeiten zwischen den Relationen r_1 und r_2 verwendet:

```
INPUT: Relation r1 und Relation r2
OUTPUT: negierte Inklusionsabhängigkeiten zwischen den Relationen r1 und r2

FOR k:= 1 TO Attributanz_r1 DO
    FOR l:= 1 TO Attributanz_r2 DO
        i:= 1;
        gespeichert:=FALSE;
        WHILE NOT gespeichert AND (i < Tupelanz_r1) DO
                gefunden:=FALSE;
                FOR j:= 1 TO Tupelanz_r2 DO
                    IF (t_i[A_k] = t_j[A_l]) THEN
                        gefunden:=TRUE;
                    END;
                END;
                IF NOT gefunden THEN
                    ⟨ Abspeicherung von NID A_k ⊈ A_l, sofern nicht schon vorhanden ⟩
                END;
                gespeichert:= TRUE;
            END;
        END;
    END;
END;
```

Algorithmus 3.5: Ableitung negierter Inklusionsabhängigkeiten aus zwei Relationen

Anschließend wird für alle abgeleiteten negierten Inklusionsabhängigkeiten $X \subseteq Y$ untersucht, ob auch eine weitere negierte Inklusionsabhängigkeit $X' \subseteq Y'$ ermittelt wurde, sodaß $X \subseteq X'$ und $Y \subseteq Y'$. Ist das der Fall, dann muß $X \subseteq Y$ gelöscht werden, da diese Information redundant ist. Auf diese Weise wird gewährleistet, daß eine *redundanzfreie Mengen von negierten Inklusionsabhängigkeiten* aus einer gegebenen Datenbank abgeleitet wird.

Die maximale Anzahl der Vergleichsoperationen, die bei diesem Verfahren notwendig ist, beträgt
(Tupelanz_$r1$ * Tupelanz_$r2$ * Attributanz_$r1$ * Attributanz_$r2$).

VI Kardinalitäten card(0,_)

Existieren zwei Relationen-Schemata R_1 und R_2, in denen der Schlüssel von R_1 als Fremdschlüssel in R_2 auftaucht, so muß die Kardinalität $\mathsf{card}(R_2, R_1)$ bestimmt werden. Dieses kann durch Auswertung gegebener Datenbanken r_1 von R_1 und r_2 von R_2 erfolgen. Treten in einer Relation r_1 Tupel auf, die nicht als Fremdschlüssel in r_2 vorkommen, so kann man bei der zu ermittelnden Kardinalität die 1 im Minimum ausschließen. Es gilt aufgrund der Einschränkung der Minimum-Werte der Kardinalitäten auf die Werte 0 und 1, daß dann die Kardinalität $\mathsf{card}(R_2, R_1) = (0, \_)$ gilt.

Für die Ermittlung dieser Kandidaten wird ein Algorithmus nach folgender Idee verwendet:
Für alle Tupel von r_1 erfolgt die Suche, ob der Wert des Schlüssels als Fremdschlüssel in r_2 vorkommt. Gibt es ein Tupel, für das dieses nicht auftritt, so weiß man, daß die Kardinalität $\mathsf{card}(R_2, R_1) = (0, \_)$ gilt.

Beispiel 3.3:

Student:

Matrikelnr	Name	Vorname	Hauptfach	Fachsemester
12273	Schulz	Peter	Mathematik	1
37623	Meier	Iris	Medizin	5
12884	Schmidt	Maria	E-Technik	7

Vorlesungsbelegung:

Student_Matrikelnr	Fach	Semester
37623	Anatomie	WS96
37623	Biologie	WS96
12884	Mathematik	WS96
12884	Informatik	WS96
12884	Statik	WS96

Im Beispiel 3.3 ist dieser Fall dargestellt, es läßt sich aus diesem Beispiel die Kardinalität $\mathsf{card}(Vorlesungsbelegung, Student) = (0, \_)$ ableiten.

In Datenbanken, die in NF^2-Form sind, können in analoger Weise aus Daten Integritätsbedingungen abgeleitet werden. Die Ableitung der Integritätsbedingungen aus Daten basiert auf dem Vergleich von gleichen und unterschiedlichen Werten. Bei Datenbanken in NF^2-Form muß nicht nur der Vergleich atomarer Attribute, sondern auch der Vergleich von Tupeln, Listen und Mengen eingesetzt werden.

Durch Auswertung von Daten werden nur Informationen über negierte Integritätsbedingungen Σ_N aus den Daten abgeleitet. Um diese zu erkennen, benötigt man nur eine konkrete Relation bzw. Datenbank. Geltende Integritätsbedingungen Σ_P können nicht aus Daten abgeleitet werden, da man nicht weiß, ob diese nur zufällig in der Relation oder Datenbank erfüllt sind.

Eine weitere Möglichkeit zur Ableitung von Integritätsbedingungen der Relationen- bzw. Datenbank-Schemata ist die Auswertung von Datenbankstatistik-Angaben.

3.3 Auswertung der Datenbank-Statistik (active domains)

Teilweise lassen sich bereits aus den *Wertebereichen der Attribute* Angaben über Integritätsbedingungen der Relationen- bzw. Datenbank-Schemata ableiten. So kann z.B. ein binäres Attribut, das nur zwei verschiedene Werte annehmen kann, nicht Schlüssel einer Relation mit mehr als zwei Tupeln sein. Oft ist jedoch aus der Definition der Wertebereiche nicht ersichtlich, wieviele verschiedene Werte jedes Attribute tatsächlich annimmt. Eine konkretere Angabe als die Wertebereiche (domains) ist deshalb die Information, wieviele verschiedene Werte des Wertebereiches eines Attributes in einer Relation tatsächlich auftreten (active domain) und wieviele Tupel in einer Relation vorhanden sind. Diese Informationen werden in [Hai96] als *Datenbank-Statistik* bezeichnet.
Die Informationen über die Anzahl der Tupel jeder Relation und die Anzahl der verschiedenen Werte, die jedes Attribut der Relationen in einem konkreten Datenbankzustand annimmt, können für verschiedene Aufgaben verwendet werden. Diese werden in [Hai96] wie folgt aufgezählt:

- Berechnungen des aktuellen und zukünftigen Umfanges von Datenbanken
- Abschätzen der Kosten für Queries und Anwendungen
- Optimierung von Anfragen
- logischer/physischer Entwurf, Wahl einer optimalen Datenstrukturierung

Die Datenbank-Statistik kann auch verwendet werden, um einige negierte *Integritätsbedingungen* zu ermitteln. Das führt besonders dann zu neuen Informationen über die Semantik der Datenbanken, wenn nur wenige Daten auswertbar sind. Diese können in einem Tool zur Semantikakquisition bestimmt werden.

Man kann die Informationen über die Datenbank-Statistik vom Benutzer erfragen. Man kann erfragen, wieviele Tupel in jeder Relation auftreten werden und wie viele verschiedene Werte jedes Attribut der Relation annimmt. Dadurch bekommt man viel Hintergrundwissen über das Datenbank-Schema und damit auch über die lokalen und globalen Integritätsbedingungen des Datenbank-Schemas. In diesem Abschnitt wird aufgezählt, welche negierten Integritätsbedingungen sich aus diesen Informationen ableiten lassen.

I Negierte Schlüssel

Alle Attribute und Attributmengen, die weniger verschiedene Werte annehmen als Tupel in der Relation auftreten werden, können keine Schlüssel sein, da diese nicht alle Tupel identifizieren

können.
Dieses soll an einem Beispiel erläutert werden. Für eine Personendatei werden folgende Angaben zur Anzahl der verschiedenen Werte gemacht.

Beispiel 3.4:

Personen:

Tupelanzahl	2000
Personennummer	2000
Name	1400
Vorname	400
Wohnort	20
PLZ	40
Straße	1600
Nummer	300

Aus dem Beispiel sind folgende negierte Schlüssel ableitbar:

- Name
- Vorname
- Wohnort PLZ
- Straße
- Nummer

Aus der Datenbank-Statistik können also unter Umständen sehr viele Informationen über Integritätsbedingungen abgeleitet werden. Oft werden durch diese Methode gerade solche Integritätsbedingungen ausgeschlossen, die einem Benutzer trivialerweise nicht erfüllt scheinen.

II Negierte funktionale Abhängigkeiten

Zwischen zwei Attributmengen X und Y besteht eine negierte funktionale Abhängigkeit $X \not\rightarrow Y$, wenn X weniger verschiedene Werte in einer Relation annimmt als Y. Die Begründung ist einfach: In dem Fall, daß X weniger verschiedene Werte annimmt als Y, muß der Fall auftreten, daß 2 Tupel in X den gleichen Wert haben, in Y aber einen unterschiedlichen. Dann liegt nach Definition eine negierte funktionale Abhängigkeit vor.

Für das Beispiel 3.4 lassen sich folgende negierte funktionale Abhängigkeiten ableiten:

- Name $\not\rightarrow$ Personennummer
- Name $\not\rightarrow$ Straße
- Vorname $\not\rightarrow$ Personennummer
- Vorname $\not\rightarrow$ Name
- Vorname $\not\rightarrow$ Straße
- PLZ Wohnort $\not\rightarrow$ Personennummer
- PLZ Wohnort $\not\rightarrow$ Name
- PLZ Wohnort $\not\rightarrow$ Straße
- Wohnort $\not\rightarrow$ Vorname
- Wohnort $\not\rightarrow$ PLZ
- Wohnort $\not\rightarrow$ Nummer
- PLZ $\not\rightarrow$ Vorname
- PLZ $\not\rightarrow$ Nummer
- Straße $\not\rightarrow$ Personennummer
- Nummer $\not\rightarrow$ Personennummer
- Nummer $\not\rightarrow$ Name
- Nummer $\not\rightarrow$ Vorname
- Nummer $\not\rightarrow$ Straße

III Negierte Inklusionsabhängigkeiten

Zwischen zwei Attributen $R.X$ und $S.Y$ eines Datenbank-Schemas können negierte Inklusionsabhängigkeiten abgeleitet werden, wenn $R.X$ mehr verschiedene Werte annimmt als $S.Y$. Diese Information ist jedoch nur über solchen Attributen sinnvoll, die gleiche Typen und gleiche oder ähnliche Längen haben.

IV Kardinalitäten card(0,_) und card(_,n)

Aus der Anzahl der Tupel der einzelnen Relationen können Kardinalitäten abgeleitet werden. Voraussetzung ist die Existenz von Fremdschlüsselbeziehungen zwischen den Relationen-Schemata.
Gibt es zwei Relationen r_1 und r_2 einer Datenbank und r_2 enthält mehr Tupel als r_1, dann kann man ableiten, daß $\mathsf{card}(R_2,R_1)=(\_,n)$ gilt.
Enthält eine Relation r_1 weniger Tupel als eine Relation r_2, dann ist ableitbar, daß $\mathsf{card}(R_1,R_2)=(0,\_)$ gilt.
Diese Ableitung ist auch in [BCN92] erläutert. Dort wird aufgrund der Tupelanzahl der Mittelwert der Kardinalitäten berechnet, der für Effizienz-Abschätzungen verwendet wird.

Mit dieser einfachen Methode der Auswertung von Datenbank-Statistik können sehr viele negierte Integritätsbedingungen Σ_N gefunden werden. Das gilt insbesondere, wenn in einer Datenbank viele Attribute vorhanden sind, deren Wertebereich nur wenige Werte umfaßt, z.B. binäre Werte oder Aufzählungstypen. Diese abgeleiteten nicht geltenden Integritätsbedingungen gelten für die zugehörigen Relationen- bzw. Datenbank-Schemata.

Die Angaben über die Datenbank-Statistik sind *für den Benutzer* in der Regel *leichter anzugeben* als die Spezifikation formaler Integritätsbedingungen. Auch wenn *diese Angaben* nur *unvollständig* gemacht werden können, lassen sich daraus einige negierte Integritätsbedingungen ableiten.

Durch diese Methode können insbesondere *solche negierten Integritätsbedingungen* ermittelt werden, die dem Benutzer *offensichtlich erscheinen*, z.B. ein binäres Attribut kann als Schlüssel ausgeschlossen werden. Zusätzlich zur Einschränkung der Menge der noch unbekannten Integritätsbedingungen hat das den Vorteil, daß bei der Diskussion der Integritätsbedingungen mit dem Benutzer die *Akzeptanz erhöht* werden kann, da viele einem Benutzer nicht sinnvoll erscheinende Integritätsbedingungen durch die Auswertung von Datenbank-Statistik bereits bekannt sind und deshalb von der Diskussion ausgeschlossen werden.

Eine *Erweiterung dieser Methode* wäre möglich, indem detailliertere Angaben zu den Werten der Datenbank (über die Verteilung der Werte der Attribute) erfragt werden. Daraus ließen sich weitere Angaben zu den nicht geltenden Integritätsbedingungen ableiten. Diese Angaben sind aber für Benutzer nur schwer zu machen, deshalb ist die Anwendung in einem Tool, das die Forderung der allgemeinen Verständlichkeit erfüllen muß, nicht geeignet. Sind diese Angaben bekannt, z.B. zu erwartende Verteilungen bei Meßwerten, so kann man diese Angaben zur Semantikermittlung auswerten. Detaillierte Arbeiten dazu gibt es auf dem Gebiet *Data Mining* ([PiF91], [FPM91],[KRT95]).

Kapitel 4

Unterstützung von expliziten Angaben zur Semantik

In dieser Arbeit werden Methoden zur Semantikakquisition vorgestellt, durch die für einen Benutzer ein einfach verständliches und nicht formales Vorgehen erreicht wird. Trotzdem müssen explizite formale Angaben zur Semantik berücksichtigt werden, um alle bereits vorhandenen Informationen auswerten zu können. Verfügt man über formal spezifizierte Integritätsbedingungen, so wird das in Abbildung 2.4 vorgestellte Verfahren in modifizierter Weise durchgeführt (Abbildung 4.1).

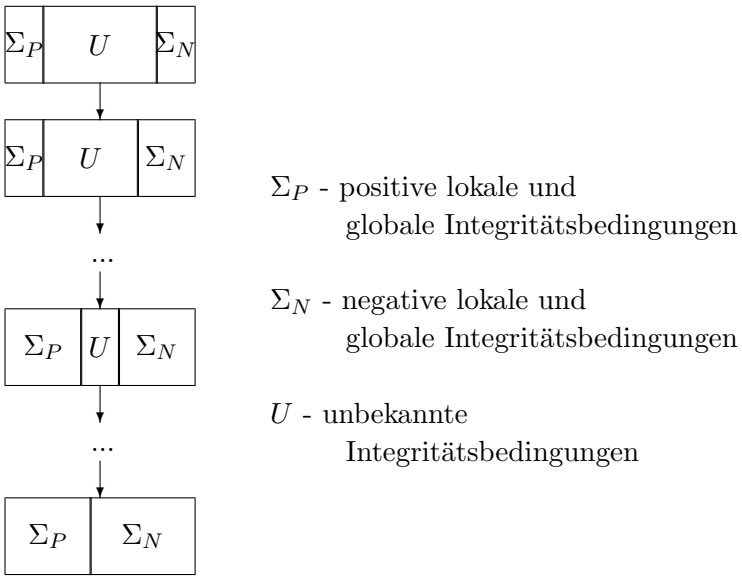


Abbildung 4.1: Semantikakquisition auf der Basis von expliziten Angaben

Auf der Basis der bekannten geltenden und negierten Integritätsbedingungen wird dann die Semantikakquisition durchgeführt.

Es können sowohl beim Entwurf und als auch beim Reverse-Engineering von Datenbanken Informationen über geltende und negierte Integritätsbedingungen bekannt sein. Woher diese

Informationen über die Semantik stammen können, wird in Abschnitt 4.1 angegeben. In Abschnitt 1.3.2 wurde beschrieben, welche Probleme viele Benutzer bei der formalen Angabe von Integritätsbedingungen haben. Aufgrund dessen muß gesichert werden, daß die explizit angegebenen Informationen über die Semantik, die aus verschiedenen Quellen stammen können, richtig sind. Die dazu durchgeführten Konsistenzprüfungen werden in Abschnitt 4.2 beschrieben. In Abschnitt 4.3 werden noch einmal die Vorteile, die durch die Einbeziehung formaler Angaben zur Semantik entstehen, zusammengefaßt.

4.1 Quellen formal angegebener Integritätsbedingungen

Es gibt verschiedene Möglichkeiten, warum formal spezifizierte Integritätsbedingungen in relationalen Datenbanken bekannt sein können.

- **Übersetzung aus einem konzeptuellen Datenmodell**
  Im *Datenbank-Entwurf* werden Datenbanken in einem konzeptuellen Modell entworfen und in ein logisches Modell übersetzt. Dadurch sind einige Integritätsbedingungen, die im konzeptuellen Modell strukturell dargestellt werden konnten, auch im logischen Datenmodell explizit bekannt. Z.B. sind bei Datenbanken, die im Entity-Relationship Modell entworfen und in das relationale Datenmodell übersetzt wurden, Schlüssel und Fremdschlüsselbeziehungen und damit auch Inklusionsabhängigkeiten bekannt. Diese Informationen sollen im relationalen Datenmodell nicht verloren gehen, da sie nicht strukturell darstellbar sind, müssen sie als explizite Integritätsbedingungen gespeichert werden.

- **Explizite Angaben beim Reverse-Engineering**
  Im *Reverse-Engineering* können in bestehenden Datenbanken einige Integritätsbedingungen explizit bekannt sein. Diese sollen im Tool zur Semantikakquisition ebenfalls ausgewertet werden, um zu verhindern, daß diese Informationen erneut erfragt werden.

- **Explizite Angaben durch den Benutzer**
  Beim *Datenbank-Entwurf* können viele Datenbank-Entwerfer einige formale Integritätsbedingungen, z.B. Schlüssel, spezifizieren. Ebenso können viele Datenbank-Administratoren im *Reverse-Engineering* solche expliziten Angaben machen. Es ist sinnvoll, solche Informationen direkt einzulesen und auszuwerten und nicht in einem langwierigen Dialog zu diskutieren, da mit diesem Vorgehen keine Akzeptanz beim Benutzer erreicht werden kann. Benutzer wollen die Möglichkeit haben, ihr Wissen über die Semantik der Datenbanken direkt angeben zu können. Darauf aufbauend erfolgt die Suche nach weiteren Integritätsbedingungen.

Es ist also im *Datenbank-Entwurf* und beim *Reverse-Engineering* von Datenbanken erforderlich, das explizite Eingeben von Informationen zu unterstützen.

4.2 Notwendige Konsistenzprüfungen

Die eingegebenen Integritätsbedingungen sollen (soweit möglich) überprüft werden. Das ist erforderlich, um zu vermeiden, daß falsche Integritätsbedingungen spezifiziert werden, weil ein

Benutzer die Definition der Integritätsbedingungen falsch verstanden hat oder in bestehenden Datenbanken eine andere Definition der Integritätsbedingungen zugrunde gelegt wurde. In Abschnitt 1.3.1 wurde dargestellt, welche Probleme bei der expliziten Eingabe der Semantik auftreten können.

4.2.1 Konsistenzprüfung im Dialog mit dem Benutzer

Eine Überprüfung der explizit angegebenen Integritätsbedingungen ist anhand von *Armstrong-Datenbanken* möglich. Angegebene Integritätsbedingungen werden dabei anhand von generierten Datenbanken dargestellt und die Gültigkeit dieser Datenbanken wird erfragt. Armstrong-Datenbanken werden für reale Anwendungen sehr umfangreich, deshalb sind sie nicht in jedem Fall ein geeignetes Mittel zur Überprüfung der Integritätsbedingungen.

Man kann auch für jede explizit angegebene Integritätsbedingung eine Beispieldatenbank generieren und diese validieren. Das Vorgehen dazu wird in Kapitel 9 gezeigt. Diese Validierung führt zu einem langwierigen Dialog mit dem Benutzer, in dem noch einmal alle expliziten Angaben in einer veränderten Form diskutiert werden.

Es ist auch möglich, die eingegebenen Integritätsbedingungen durch eine pseudonatürlichsprachige Erklärung zu umschreiben und noch einmal zu erfragen.

Eine Diskussion der expliziten Angaben sichert die Richtigkeit der gemachten Angaben. Ein solches Vorgehen ist aber nicht immer erwünscht, da die angegebenen Informationen noch einmal mit dem Benutzer diskutiert werden. Es existiert jedoch auch die Möglichkeit, explizite Angaben *ohne Interaktion mit dem Benutzer* zu überprüfen. Dabei kann jedoch keine vollständige Überprüfung der angegebenen Integritätsbedingungen erreicht werden. Möglichkeiten dazu werden im nächsten Abschnitt erläutert.

4.2.2 Konsistenzprüfung ohne Benutzer-Interaktion

Konsistenzprüfungen ohne Benutzerinteraktion sollten immer durchgeführt werden, da es möglich sein kann, daß Fehler in den angegebenen Integritätsbedingungen dadurch gefunden werden, die dann mit dem Benutzer diskutiert werden müssen. Es muß bei der Konsistenzprüfung getestet werden, ob die explizit eingegebenen Integritätsbedingungen nicht in *Konflikt* zur Menge der bereits bekannten Integritätsbedingungen stehen.
Da auch aus den Daten bekannter Relationen oder Datenbanken formale Integritätsbedingungen abgeleitet werden können, erfolgt mit der Konfliktprüfung auch die Überprüfung, ob die explizit eingegebenen formalen Integritätsbedingungen auch *in den bekannten Daten des Datenbank-Schemas gelten*. Ebenso wird überprüft, ob die Angaben zu den Integritätsbedingungen konform zu der Datenbank-Statistik sind.

Im Falle eines gefundenen Konfliktes müssen die sich widersprechenden Integritätsbedingungen dem Benutzer gezeigt werden, mindestens eine dieser Integritätsbedingungen muß falsch sein.

Beispiel: Für einen Benutzer kann die Konfliktprüfung folgendermaßen aussehen:

Zwischen folgenden Integritätsbedingungen besteht ein Konflikt:

explizit eingegeben:	Name	$\longrightarrow$	Adresse
aus den Daten:	Name Vorname	$\not\longrightarrow$	Adresse

Welche dieser Angaben ist falsch ?

Die Lösung eines Konfliktes kann nicht automatisch geschehen. Das Vorgehen bei der Konfliktprüfung wurde ausführlich in Kapitel 2.4.2 beschrieben.

Tritt kein Konflikt auf, so handelt es sich bei dieser Überprüfung um eine Methode, die ohne Interaktion mit dem Benutzer erfolgt. Bei dieser Methode ist nur im Falle eines Konfliktes eine Diskussion der eingegebenen Integritätsbedingungen mit dem Benutzer erforderlich.

4.3 Vorteile der Einbeziehung expliziter Angaben in ein Tool

Durch eine Unterstützung expliziter Angaben wird erreicht, daß die Methode zur Semantikakquisition *universell einsetzbar* wird, da sie *unterschiedliche Ausgangsinformationen* auswerten kann und auf diesen aufbaut. Die Kombination der vorgestellten Methoden zur Semantikakquisition mit anderen Verfahren ist dadurch möglich. Die *redundanten Eingabe* identischer Informationen wird auf diese Weise *vermieden*.

Man erreicht durch die Einbeziehung expliziter Angaben zur Semantik auch eine einfache *Benutzer-Adaption*, ein Benutzer kann alle Informationen zur Semantik angeben, die er formal spezifizieren kann. Die Semantikakquisition wird dann aufbauend auf diesen Angaben durchgeführt. Dadurch können Benutzer mit unterschiedlichen Fähigkeiten bei der Spezifikation von Integritätsbedingungen mit den gleichen Verfahren unterstützt werden.

Da die bekannten Integritätsbedingungen meist unvollständig sind und auch ein Benutzer, der Integritätsbedingungen spezifiziert hat, dabei einige geltende Integritätsbedingungen vergessen kann, oder generell nicht in der Lage ist, alle Arten Integritätsbedingungen vollständig anzugeben, kann die explizite Eingabe in jedem Fall nur ein *erster Schritt* der Semantikakquisition sein. Aufbauend auf den expliziten Angaben des Benutzers muß anschließend nach weiteren geltenden Integritätsbedingungen gesucht werden.

Kapitel 5

Heuristiken zur Abschätzung der unbekannten Integritätsbedingungen

In den vorherigen Kapiteln wurde gezeigt, daß durch die Auswertung von Daten, Datenbankstatistik-Angaben und von expliziten Eingaben einige geltende und negierte Integritätsbedingungen des Relationen- bzw. Datenbank-Schemas ermitteln werden können. Man kann jedoch nicht davon ausgehen, daß die so abgeleiteten Integritätsbedingungen vollständig sind. Es muß deshalb überprüft werden, ob weitere Integritätsbedingungen gelten. Die Ermittlung von weiteren geltenden oder negierten Integritätsbedingungen ist ohne Bestätigung durch den Benutzer nicht möglich, es ist also notwendig, eine *Diskussion der unbekannten Integritätsbedingungen mit dem Benutzer* durchzuführen. Die *Anzahl der unbekannten Integritätsbedingungen* eines Relationen- oder Datenbank-Schemas kann *sehr groß* sein, deshalb ist eine Diskussion der Integritätsbedingungen der Reihe nach nicht möglich. Man sucht also nach einer Möglichkeit, den Dialog mit dem Benutzer zu beschleunigen. Auch möchte man mit dem Benutzer einen Dialog führen, der diesem sinnvoll erscheint.

Deshalb ist eine *Abschätzung der unbekannten Integritätsbedingungen vor ihrer Erfragung* erforderlich. Dabei wird dafür bestimmt, mit welcher Plausibilität die noch unbekannten Integritätsbedingungen geltende Integritätsbedingungen sind. Dazu werden *Heuristikregeln* eingesetzt, durch die *in der Menge der unbekannten Integritätsbedingungen plausible Kandidaten* für Integritätsbedingungen und Kandidaten für negierte Integritätsbedingungen ermittelt. Abbildung 5.1 zeigt die Einordnung dieser Kandidaten in die Menge der Integritätsbedingungen.

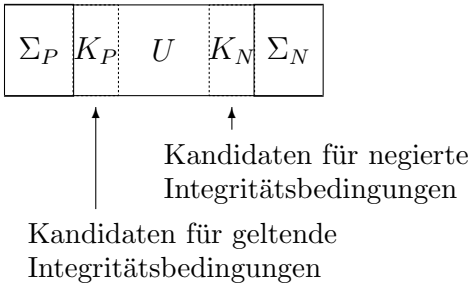


Abbildung 5.1: Einordnung der Kandidaten für Integritätsbedingungen und negierte Integritätsbedingungen

5.1 Einordnung der Kandidatenermittlung in die Diskussion unbekannter Integritätsbedingungen

Abbildung 5.2 zeigt, wie sich die Abschätzung von Kandidaten in die Diskussion von unbekannten Integritätsbedingungen einordnet.

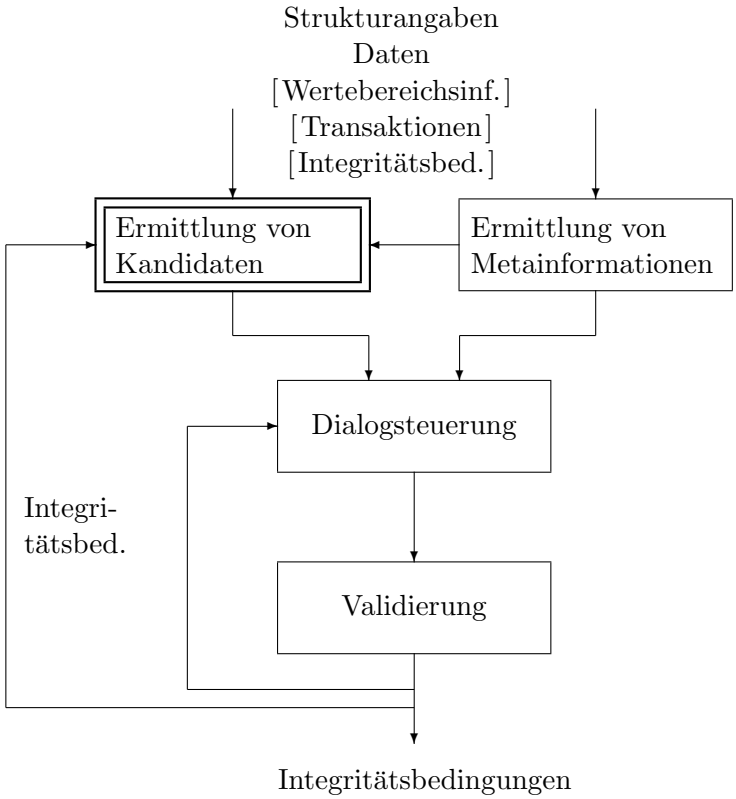


Abbildung 5.2: Gezielte Diskussion von Integritätsbedingungen

In diesem Kapitel werden Heuristiken zur Suche nach plausiblen Kandidaten für Schlüssel, funktionale Abhängigkeiten, Inklusions- und Exklusionsabhängigkeiten und Kardinalitäten, sowie für die entsprechenden negierten Integritätsbedingungen angegeben. Weitere Methoden zur Ermittlung von Kandidaten für Integritätsbedingungen folgen in Kapitel 6.

Neben Kandidaten für Integritätsbedingungen lassen sich durch Anwendung von Heuristikregeln weitere Informationen über die Semantik eines Datenbank-Schemas ableiten, diese werden als *Metainformationen* bezeichnet. In *Kapitel 7* wird erläutert, um welche Informationen es sich dabei handelt und wie diese zur Semantikakquisition verwendet werden können. Die ermittelten Kandidaten für Integritätsbedingungen und die Metainformationen sind Voraussetzung für die gezielte Erfragung unbekannter Integritätsbedingungen. Die *Dialogsteuerung* (Kapitel 8) legt dabei die Reihenfolge der Erfragung der unbekannten Integritätsbedingungen nach den Kriterien Effizienz und Nachvollziehbarkeit fest. Die *Validierung* der einzelnen durch die Dialogsteuerung ausgewählten Integritätsbedingungen wird in *Kapitel 9* vorgestellt.

5.2 Ziele, Möglichkeiten und Grenzen der Heuristikregeln

Ein Benutzer, der über das notwendige *Weltwissen* und *Kenntnisse über das Anwendungsgebiet* einer Datenbank verfügt, kann aus einer Datenbank viele Informationen über die Bedeutung der Daten ersehen, da die Bezeichnungen der Datenbank in der Regel viel Hintergrundwissen über die Datenbank widerspiegeln. Aufgrund dessen kann ein geübter Datenbank-Entwerfer auch einige semantische Zusammenhänge (und damit auch *formale Integritätsbedingungen*) direkt aus der Datenbank erkennen. Man möchte erreichen, daß ein Tool zur Semantikakquisition solche Zusammenhänge durch Auswertung verschiedener Datenbankcharakteristika teilweise ebenfalls erkennt. Es ist problematisch, diesen Prozeß zu automatisieren, da kein umfangreiches Allgemeinwissen und Wissen für jedes Anwendungsgebiet vorhanden ist, das maschinell auswertbar wäre. Dennoch ist es möglich, einige allgemeine *Heuristikregeln* zur Abschätzung der unbekannten Integritätsbedingungen zu entwickeln.

Die hier vorgestellten Heuristiken gehen von der *Annahme* aus, daß Informationen über die *Semantik* der Datenbank sich in den folgenden *Merkmalen der Datenbank widerspiegeln*:

- Bezeichnungen
- Wertebereiche
- Daten
- bereits bekannte Integritätsbedingungen
- Transaktionen

Es wird in den Heuristiken versucht, das Hintergrundwissen des Anwendungsgebietes, das sich in diesen Datenbankmerkmalen widerspiegelt, auszunutzen und Kandidaten für geltende und negierte Integritätsbedingungen daraus abzuleiten.

Alle in diesem Kapitel vorgestellten Heuristiken zur Ermittlung von Kandidaten sind *begründbar*, aber sowohl die Heuristiken selbst als auch die Wichtung dieser Heuristiken wurden *intuitiv festgelegt*. Eine *formale Begründung* der Heuristiken und ihrer Wichtung wäre nur über eine *statistische Auswertung großer Datenbestände* möglich.

Das Zutreffen von Heuristiken zur Semantikabschätzung in Datenbanken ist von verschiedenen Punkten abhängig:

- vom Entwerfer (Qualifikation, Erfahrung, Eigenheiten, Anzahl der Entwerfer, Absprachen der Entwerfer untereinander, ..)
- vom Anwendungsgebiet (Größe und Strukturiertheit der Anwendung, Vorhandensein plausibler Bezeichnungen für dargestellte Zusammenhänge, restriktive Vorgaben der Unternehmen, Organisation innerbetrieblicher Prozesse, ..)
- von der Datenbank (Qualität der entworfenen Datenbank, Größe der Relationen, Größe der Datenbank, Verbindung der Relationen untereinander, verwendete Bezeichnungen, Umfang und Aussagefähigkeit der bekannten Daten, ..)

Damit müßten auch statistische Untersuchungen zum Zutreffen der Heuristiken in Abhängigkeit von diesen drei Punkten gemacht werden. Das ist natürlich nicht für jede Datenbank realisierbar. Weiterhin bestünde bei einer statistischen Untersuchung von großen Datenbankbestände

und Anpassung der Heuristiken zur Semantikabschätzung die Gefahr, daß auf diese Weise nur die am häufigsten vorkommenden Integritätsbedingungen beachtet werden und seltener vorkommende (und häufig auch kompliziertere Integritätsbedingungen) dabei nicht eingehen. Auf Grund dessen können die hier vorgestellten Heuristiken nur *intuitiv begründet* werden.

Mit Heuristikregeln können nur Kandidaten für Integritätsbedingungen gefunden werden, die ermittelten Kandidaten können *unvollständig* sein, es können auch *falsche Kandidaten* durch die Heuristiken abgeleitet werden. Man kann versuchen, dem weitgehend zu begegnen, indem *zahlreiche verschiedene Heuristikregeln* zur Abschätzung der Semantik *angewendet* und *gegeneinander gewichtet* werden. Damit werden die Abschätzungen der unbekannten Integritätsbedingungen genauer. Eine Validierung der Kandidaten im Dialog mit dem Entwerfer ist jedoch erforderlich, um sicherzustellen, daß keine falschen Integritätsbedingungen ermittelt werden.

5.3 Heuristiken zur Schlüsselsuche

Schlüssel stellen eine wichtige Information in Datenbanken dar. Durch einen Schlüssel werden die Tupel einer Relation identifizierbar. Deshalb muß mindestens ein Schlüssel zu jedem Relationen-Schema bekannt sein. Kann der Benutzer keinen oder keinen weiteren Schlüssel angeben, so sollen ihm bei einer Unterstützung der Semantikakquisition plausible Schlüsselkandidaten vorgeschlagen werden.

5.3.1 Heuristikregeln

Zur Suche nach Schlüsselkandidaten kann man Informationen über die Struktur des Relationen-Schemas, die Beispieldaten, den Umfang der Wertebereiche jedes Attributes, die bisher bekannten Integritätsbedingungen und die Transaktionen auswerten. Alle Attributmengen, über die noch keine Aussage zur Schlüsseleigenschaft bekannt ist, müssen untersucht werden. Für jedes Attribut wird dazu eine Wahrscheinlichkeit abgeschätzt, mit der dieses Attribut Teil eines Schlüssels des Relationen-Schemas ist. Es werden folgende Heuristikregeln für jedes Attribut betrachtet:

Ableitung künstlicher Schlüssel aus der Struktur

H_S1. Oft werden in Datenbanken Schlüssel künstlich eingeführt. Wenn ein Attribut mit dem *Typ Integer* und einer *sehr großen Länge* vereinbart wurde, so könnte dieses Attribut ein Schlüssel sein.

H_S2. Attributnamen können Hinweise auf Schlüssel liefern.
In [Lai95] wird für die Verwendung natürlicher Bezeichnungen in Programmen zur Erhöhung der Lesbarkeit plädiert. Auch in Datenbanken ist dieses sinnvoll und erfolgt häufig. Sind solche aussagekräftigen Bezeichnungen vorhanden, so kann durch ein Tool versucht werden, daraus Rückschlüsse über die Semantik zu ziehen.
Kommt in einem *Attributnamen* ein *Teilstring -name-, -nummer-, -nr-, -identifikator-, -id-, -key-, -kennzahl-, -knz-, -code-, -bezeichnung-, -bez-, -#-* usw. vor, so deutet das darauf hin, daß dieses Attribut Teil eines Schlüssels des Relationen-Schemas ist.

Diese beiden Heuristiken wurden auch in [StG88] vorgeschlagen, in [ChV92] wurden sie für die Ableitung von Schlüsselinformationen aus Formularen eingesetzt.

Auswertung von Daten

H_S3. Sind in den Daten der Relationen bei einem Attribut *laufende Nummern* eingetragen, so deutet das stark darauf hin, daß dieses Attribut ein Schlüssel oder Teil eines Schlüssels ist.

Diese Heuristik wurde in [ChV92] auf Formulare angewendet, sie kann auf relationale Datenbanken übertragen werden.

Kardinalität des Wertebereiches

H_S4. Attribute, die *viele verschiedene Werte* in einer Relation annehmen, sind eher Schlüssel oder Teil des Schlüssels als Attribute, die nur wenige verschiedene Werte in der Relation annehmen.

Schlußfolgerungen aus der bereits bekannten Semantik

H_S5. Vermutungen über Schlüssel lassen sich auch aus der *Menge der bekannten funktionalen und negierten funktionalen Abhängigkeiten* ableiten.
Ein Attribut, das auf der linken Seite vieler funktionaler Abhängigkeiten steht, das also andere Attribute bestimmen, ist mit größerer Wahrscheinlichkeit Teil eines Schlüssels als ein Attribut, das auf der linken Seite vieler negierter funktionaler Abhängigkeiten steht. Umgekehrt verhält es sich, wenn das Attribut auf der rechten Seite von Abhängigkeiten steht. Befindet es sich auf der rechten Seite einer funktionalen Abhängigkeit, so kann das Attribut zumindest kein minimaler Schlüssel sein. Steht es auf der rechten Seite einer negierten funktionalen Abhängigkeit, ist es also von anderen Attributen unabhängig, so deutet das wiederum darauf hin, daß es Schlüssel oder Teil eines Schlüssels der Relation ist.

Formal: Folgende Abhängigkeiten deuten darauf hin, daß ein Attribut A Schlüssel oder Teil eines Schlüssels ist:

- nichttriviale funktionale Abhängigkeiten X $\longrightarrow Y, A \subseteq X$
- negierte funktionale Abhängigkeiten Y $\not\longrightarrow A$.

Das Auftreten folgender Abhängigkeiten spricht dagegen, daß A Schlüssel oder Teil eines Schlüssels ist:

- nichttriviale funktionale Abhängigkeiten $Y \longrightarrow A$,
- negierte funktionale Abhängigkeiten $X \not\longrightarrow Y, A \subseteq X$.

Diese Heuristik ist besonders aussagekräftig, weil ihre Ergebnisse sich zunehmend mit der Menge der bekannten Integritätsbedingungen verbessern.

Auswertung von Transaktionen

H_S6. Aus bekannten Transaktionen lassen sich Schlüsselkandidaten ableiten.
Die Attribute, die bei *Updates nie* oder *nur sehr selten verändert* werden, sind mit größerer Wahrscheinlichkeit Schlüssel oder Teil des Schlüssels als die Attribute, die oft geändert werden.
Attribute, die in Transaktionen zur *Identifikation* von Tupeln einer Relation verwendet werden (diese stehen in SQL-Transaktionen in der where-Klausel), können Schlüssel des Relationen-Schemas sein.

5.3.2 Wichtung der Heuristikregeln

Die einzelnen Heuristikregeln werden gegeneinander gewichtet. Es soll eine Abschätzung dafür getroffen werden, mit welcher Plausibilität das Attribut A Teil eines minimalen Schlüssels ist. Dabei sollen Attribute, für die mehrere Heuristikregeln erfüllt sind, höher gewichtet werden als Attribute, für die weniger Heuristiken zutreffen. Die folgende einfache Näherungsformel wird eingesetzt, die diese angegebenen Forderungen erfüllt.

$$Pl(A \text{ ist Schlüsselattribut}) := \sum_{i=1}^{6}(w_i * H_Si(A))$$

$H_Si(A) \in [0..1]$: Ergebnis der Heuristikregel H_Si für das Attribut A
$w_i \in [0,1]$ und $w_1 + .. + w_6 = 1$

Diese Berechnung kann nicht exakt sein, da die einzelnen Heuristikregeln nicht voneinander unabhängig sind. Sie genügt aber den Anforderungen an die Abschätzung.

Die Gewichte der einzelnen Heuristiken werden aufgrund der vermuteten Aussagefähigkeit der Heuristiken festgelegt. Diese wird in folgender Übersicht angegeben:

besonders aussagekräftig	H_S3, H_S5, H_S6
aussagekräftig	H_S1, H_S2
nur im Zusammenhang mit anderen Heuristiken sinnvoll	H_S4

Die Gewichte der einzelnen Heuristiken werden entsprechend dieser Übersicht sinnvoll festgelegt, sodaß aussagekräftigere Heuristiken ein höheres Gewicht erhalten als die anderen Heuristikregeln.

Sind nicht alle Charakteristika auswertbar (z.B. es sind keine Transaktionen bekannt), so können diese nicht zur Abschätzung von Integritätsbedingungen herangezogen werden, die Gewichte der anderen Heuristiken werden dann entsprechend erhöht.

Auf diese Weise erhält man eine Plausibilitätsabschätzung für die Wahrscheinlichkeit, daß *ein Attribut Schlüssel* oder *Teil eines Schlüssels* ist.

5.3.3 Abschätzung von Schlüsseln und negierten Schlüsseln

Basierend auf der in Abschnitt 5.3.2 vorgestellten Abschätzung der Plausibilität, mit der ein Attribut Teil eines Schlüssels ist, soll für *Attributmengen* abgeschätzt werden, mit welcher Plausibilität diese *Schlüssel* des Relationen-Schemas sind. Die Anzahl der zu untersuchenden unbekannten Schlüssel kann sehr groß sein, deshalb wird eine einfache Abschätzung für die Plausibilität eines Schlüssels $X = \langle\, X_1..X_n \,\rangle$ vorgenommen. Es wird der Mittelwert der Plausibilität der einzelnen Attribute berechnet.

$$Pl(X \text{ ist Schlüssel}) := \frac{1}{n} \sum_{i=1}^{n} Pl(X_i \text{ ist Schlüsselattribut})$$

Die Verwendung des Mittelwertes für die Abschätzung der Schlüsseleigenschaften ist ebenfalls eine Vereinfachung. Dieser Mittelwert kann von den Ergebnissen, die die Heuristikregeln für eine Attributmenge ermitteln, abweichen. Der Aufwand bei der Berechnung der Heuristikfaktoren wird jedoch auf diese Weise verringert, da so die einzelnen Heuristikregeln nur einmal für jedes Attribut ausgewertet werden müssen.

Da ein negierter Schlüssel als Negation eines Schlüssel definiert wurde, kann man mit den in Abschnitt 5.3.1 vorgestellten Heuristikregeln auch eine *Abschätzung für negierte Schlüssel* vornehmen:

$$Pl(X \text{ ist negierter Schlüssel}) := 1 - \frac{1}{n} \sum_{i=1}^{n} Pl(X_i \text{ ist Schlüsselattribut})$$

Man erhält also durch die Auswertung von Heuristikregeln *Kandidaten für Schlüssel und für negierte Schlüssel*, sowie eine *Plausibilität* für diese Kandidaten.

5.3.4 Anpassung der Gewichte durch Lernalgorithmen

Die initialen Gewichte der Heuristikregeln wurden intuitiv festgelegt. Um diese optimal zu bestimmen, müßten große statistische Untersuchungen gemacht werden, die auch beachten müssen, daß für jedes Anwendungsgebiet und jeden Entwerfer andere Gewichte optimale Abschätzungen bringen. Dieses ist nicht möglich. Es soll deshalb an dieser Stelle eine andere Methode zur Anpassung der intuitiv festgelegten initialen Gewichte an die entsprechende Datenbank (und damit an das Anwendungsgebiet und den Entwerfer) vorgestellt werden. Es werden dazu in diesem Abschnitt zwei Lernalgorithmen vorgestellt, die die Abschätzung von Kandidaten mit den Ergebnissen der Validierung vergleichen und die Gewichte der Heuristikregeln für weitere Abschätzungen anpassen.

I Perzeptron-Lernalgorithmus

Ein bekannter Lernalgorithmus, der für diese Aufgabe verwendet werden kann, ist der Perzeptron-Lernalgorithmus aus dem Gebiet der neuronalen Netze ([Roj93]).

Darstellung der Heuristikabschätzung als Perzeptron. In Abbildung 5.3 wird zunächst angegeben, wie die Abschätzung der Semantikakquisition (für den Fall Schlüssel) als Perzeptron dargestellt wird, um diesen Algorithmus adaptieren zu können. An den Eingangsleitungen des Perzeptrons werden die Ergebnisse der Heuristikregeln $H_S1..H_Sn$ angetragen, diese sind mit den Gewichten $w_1..w_n$ versehen. An den Ausgangsleitungen wird als Ergebnis 0 oder 1 ausgegeben. Für die Plausibilitätsabschätzung durch ein Perzeptron bedeutet das Ergebnis, daß Schlüssel erwartet werden, wenn die Summe der gewichteten Heuristikregeln über dem Schwellwert θ liegt (Ergebnis 1), im anderen Fall wird ein negierter Schlüssel erwartet (Ergebnis 0).

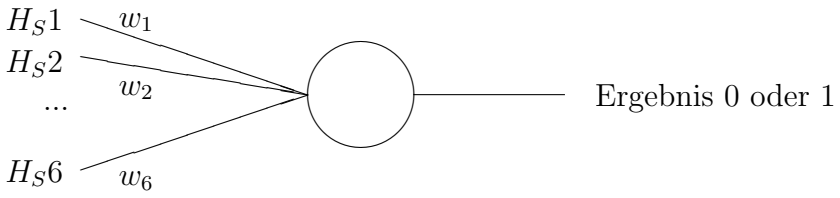


Abbildung 5.3: Darstellung der Abschätzung von Integritätsbedingungen als Perzeptron

Beim Perzeptron-Lernalgorithmus erfolgt ein *überwachtes Lernen*, bei dem die Ergebnisse des Perzeptrons mit den richtigen Ergebnissen verglichen werden und die Gewichte w_i so gesetzt werden, daß der Fehler des Perzeptrons verringert wird. Bei der Wichtung der Heuristikregeln wird die abgeschätzte Plausibilität von Schlüsselkandidaten mit dem Ergebnis der Validierung der Schlüsselkandidaten verglichen und es erfolgt eine Anpassung der Gewichte.

Konvergenz des Algorithmus. Voraussetzung für die Konvergenz des Perzeptron-Lernalgorithmus, also für das Finden von Gewichten, die in jedem Fall richtige Ergebnisse hervorbringen, ist die *lineare Trennbarkeit* von zwei Mengen A und B (in unserer Anwendung Schlüssel und negierte Schlüssel), für die es eine Belegung der Gewichte geben muß, sodaß für alle Elemente aus A gilt:

$$\sum_{i=1}^{n} w_i * H_Si > \theta$$

und für alle Elemente aus B gilt:

$$\sum_{i=1}^{n} w_i * H_Si \leq \theta$$

Die *lineare Trennbarkeit* ist bei der Abschätzung von *Schlüsseln und negierten Schlüsseln* durch die Heuristikregeln *nicht immer gegeben*, es kann Integritätsbedingungen geben, für die die Heuristikregeln identische Ergebnisse liefern, die jedoch unterschiedlich validiert werden. Eine *Konvergenz* des Algorithmus kann für solche Fälle *nicht erreicht* werden. Man kann aber auch in diesem Fall den Lernalgorithmus anwenden, dieser wird dann eine Gewichtung erreichen, die in den meisten Fällen richtige Ergebnisse bringt.

Ausführung des Perzeptron-Lernalgorithmus. Die *Gewichte* werden *initialisiert*, diese initialen Gewichte geben eine Vermutung an, wie gut die Ergebnisse der einzelnen Heuristikregeln auf existierende Schlüssel hinweisen. Der Schwellwert wird zunächst mit 0.5 festgelegt.

Eine Menge von bereits bekannten Integritätsbedingungen wird verwendet, um eine Überwachung beim Lernen zu realisieren. Durch Auswertung der Abschätzung dieser Integritätsbedingungen durch die Heuristikregeln und Vergleich mit den geltenden Integritätsbedingungen werden die Gewichte der Heuristikregeln verändert. Dabei sollen der Schwellwert θ und die Gewichte der Heuristikregeln bestimmt werden.

Für die Festlegung der Gewichte zur Plausibilitätsabschätzung von Schlüsseln werden folgende Integritätsbedingungen ausgewertet:

- geltende Schlüssel.
  Für alle gefundenen geltenden Schlüssel $X = \langle X_1..X_n \rangle$ werden die Gewichte w_i der Heuristikregeln $H_Si(X_i)$, die erfüllt waren, also richtige Hinweise gaben, wie folgt erhöht:

  $$w_i := w_i + \frac{H_Si(X_i)}{|X|}$$

- negierte Schlüssel.
  Für alle negierten Schlüssel $Y = \langle Y_1..Y_m \rangle$ werden die Gewichte w_i der Heuristikregeln $H_Si(Y_i)$, die Hinweise auf einen geltenden Schlüssel geben, also falsche Hinweise, wie folgt verringert:

  $$w_i := max\{0, w_i - \frac{H_Si(Y_i)}{|Y|}\}$$

Sind die Gewichte nach Auswertung aller Schlüssel und negierten Schlüssel neu festgelegt, so erfolgt eine *Normierung* der Gewichte, damit die Summe der Gewichte $w_1..w_n$ wieder 1 ergibt.

$$w_i := \frac{w_i}{w_1+..+w_n}$$

Es können Gewichte kleiner als 0 sein, wenn diese überwiegend falsche Ergebnisse brachten. Diese werden vor der Normierung auf 0 gesetzt, da keine Heuristiken betrachtet werden, die Hinweise auf negierte Schlüssel geben.
Anmerkung: Eine Betrachtung von Heuristikregeln, die negierte Integritätsbedingungen abschätzen, wäre eine Erweiterung der Methode. Es wäre möglich, solche Heuristikregeln aufzustellen, mit entsprechender Gewichtung im Bereich $[-1, 0]$ zu versehen und auf diese Weise in die Abschätzung von Schlüssel eingehen zu lassen.

Mit diesen veränderten Gewichten wird die Abschätzung der bereits bekannten Integritätsbedingungen vorgenommen, um den *Schwellwert* θ festzulegen.

Dieses soll an einem Beispiel erläutert werden. Es sollen folgende Integritätsbedingungen und deren Plausibilitätsabschätzung durch die Heuristiken bekannt sein:

	Validierung 1 - Schl, 0 - neg. Schl.	Abschätzung
c_1	1	0.5
c_2	0	0.2
c_3	1	0.8
c_4	0	0.3
c_5	0	0.3
c_6	1	0.4
c_7	0	0.3
c_8	1	0.35
c_9	0	0.25
c_{10}	0	0.5

Die Integritätsbedingungen werden nach den *abgeschätzten Plausibilitäten sortiert*. Die folgende Übersicht zeigt ein Beispiel dafür:

	Validierung 1 - Schl, 0 - neg. Schl.	Abschätzung
c_3	1	0.8
c_1	1	0.5
c_{10}	0	0.5
c_6	1	0.4
c_8	1	0.35
c_4	0	0.3
c_5	0	0.3
c_7	0	0.3
c_9	0	0.25
c_2	0	0.2

In dieser sortierten Folge wird der Schwellwert so festgelegt, daß die Summe der negierten Schlüssel, deren Abschätzung über dem Schwellwert liegt und der geltenden Schlüssel, deren Abschätzung unter dem Schwellwert liegt (also der Fehler der Abschätzung) minimal ist.

Für das Beispiel liegt der Schwellwert zwischen 0.3 und 0.35 ($0.3 < \theta < 0.35$). Der Fehler der Abschätzung beträgt dann $\frac{1}{10}$, da die Integritätsbedingungen c_{10} (negierter Schlüssel) in der Abschätzung über dem Schwellwert liegt.

Auf diese Weise können die Gewichte angepaßt werden. Da die *lineare Trennbarkeit* der Kandidaten für geltende und negierte Integritätsbedingungen *nicht gewährleistet* werden kann, ist es aber nicht möglich, eine Gewichtung zu erreichen, die in jedem Fall eine richtige Abschätzung liefert. Die *Validierung von abgeschätzten Kandidaten* ist deshalb in jedem Fall *erforderlich*.

II Einbeziehung von Entscheidungskosten in die Lernfunktion

Man kann anstelle des Perzeptron-Lernalgorithmus eine andere Anpassung der Gewichte anstreben. Es ist bei der Semantikakquisition der Fall, daß *nicht alle richtig bzw. falsch abgeschätzten Integritätsbedingungen* die *gleiche Bedeutung* für die Semantikakquisition haben. Einige falsch abgeschätzten Integritätsbedingungen führen zu größeren Fehlern als andere falsch abgeschätzte

Integritätsbedingungen. Eine andere, *differenziertere Fehlerfunktion* sollte deshalb herangezogen werden. Diese wurde in dem im folgenden kurz beschriebenen Artikel vorgestellt. Dieses Vorgehen kann zur Bestimmung von Gewichten für Heuristikregeln angewendet werden, eine Vorstellung folgt anschließend.

Wilke, Bergmann, Althoff: Fallbasierte Entscheidungsunterstützung in der Kreditwürdigkeitsprüfung. In den Artikeln [Wil96], [WiB96] und [WBA96] wird ein Lernalgorithmus zur Ermittlung von Eigenschaftsgewichten im fallbasierten Schließen vorgestellt, dessen Ziel nicht die Optimierung der Anzahl von korrekten Klassifikationen gegebener Falldaten, sondern die *Minimierung der durch Fehlklassifikationen entstehenden Kosten* ist.

Die allgemeine Vorgehensweise beim fallbasierten Schließen wird kurz erläutert. Dazu wird das Prozeßmodell nach Aamodt und Plaza verwendet. Anwendungsgebiete des vorgestellten Verfahren sind u.a. die Kreditwürdigkeitsprüfung. Bei der Kreditwürdigkeitsprüfung erfolgt eine Klassifikation des Kunden in die Klassen kreditwürdig bzw. nicht kreditwürdig. Für diese Klassifizierung sollen ähnliche bekannte Fälle herangezogen werden. Es wird eine Abstandsfunktion angegeben, mit der der nächste Nachbar in der Fallbasis ermittelt wird. In der Abstandsfunktion werden 136 gewichtete Attribute ausgewertet. Die Gewichte werden durch ein überwachtes Lernen ermittelt. Dazu wurden 685 Fälle ausgewertet, 70 % dienen zum Lernen der Gewichte, 30 % zur Überprüfung der Ergebnisse. Es traten 5 % unbekannte Werte in den Attributen auf. Der Lernalgorithmus zur Ermittlung der Gewichte wurde mit zufälligen Werten begonnen.

In einer *Confusionsmatrix P* wird die Wahrscheinlichkeit angegeben, mit der ein vorhergesagter Fall sich in dieser Klasse befindet.

P_{ij}	vorhergesagte Klasse	
richtige Klasse	solvent	insolvent
solvent	0.275	0.225
insolvent	0.125	0.375

Dazu wird eine *Matrix der Entscheidungskosten C* bestimmt, diese sieht für die Anwendung Kreditwürdigkeitsprüfung wie folgt aus:

C_{ij}	vorhergesagte Klasse	
richtige Klasse	solvent	insolvent
solvent	-1	1
insolvent	10	-10

Die Kosten der Entscheidung sind die gewichtete Summe, der in diese Klassen fallenden Entscheidungen, die Gewichte sind in der oberen Übersicht angegeben. Diese Gewichte werden durch die zu erwartenden Kosten und Gewinne aus der richtigen und falschen Klassifikation begründet.

$$\text{Kosten} = \sum_i \sum_j P_{ij} * C_{ij}$$

Es erfolgt der Versuch, die Gewichte zur Ähnlichkeitsbestimmung automatisch festzulegen, um bestimmte Eigenschaften der Klassifikation zu erreichen. In dem Lernschritt werden dabei aus den alten Gewichte, der Fehlerrate und einer Lernrate die neuen Gewichte ermittelt. Der folgende Algorithmus wird dazu eingesetzt:

1. Initialisiere den Gewichtsvektor $w = (w_1..w_n)$ und die Lernrate λ
2. Berechne den Wert der Fehlerfunktion $E(w)$ des anfänglichen fallbasierten Systems
3. **While** not(Stop-Kriterium) **do**
   (a) Lernschritt: $\forall a\ \hat{w}_a := w_a - \frac{\delta E}{\delta w_a} * \lambda$
   (b) Berechne $E(\hat{w})$
   (c) **If** $E(\hat{w}) < E(w)$ **then** $w := \hat{w}$ **else** $\lambda := \frac{\lambda}{2}$
4. Ausgabe des Gewichtsvektors $w = (w_1..w_n)$

Als *Fehlerfunktion* können verschiedene Funktionen eingesetzt werden. Sollen die Entscheidungskosten einbezogen werden, so müssen diese in der Fehlerfunktion mit den jeweils zugehörigen Werten der Confusionsmatrix gewichtet werden.

Es sind für den angegebenen Algorithmus mehrere *Abbruchkriterien* möglich:

- feste Anzahl von Lernschritten
- minimale Änderung der Fehlerfunktion: $|E(\hat{w}) - E(w)| \leq \varepsilon$
- minimale Änderung der Gewichte: $\sum_{a=1}^{n} |\frac{\delta E}{\delta w_a}| \leq \varepsilon$
- minimale Lernrate $\lambda \leq \varepsilon$

Als Abbruchkriterium können auch Kombinationen dieser vier Möglichkeiten gewählt werden.

In den Artikeln werden Auswirkungen von falsch gewählten Lernraten λ betrachtet, diese führen zu einer größeren Anzahl notwendiger Lernschritte.

Dieses Vorgehen kann man für die *Semantikakquisition* übernehmen, auch dort sind manche Fehlklassifikationen von größerer Bedeutung als andere. Bei der Festlegung der Gewichte der einzelnen Heuristikregeln soll das berücksichtigt werden. Die folgende Matrix der Entscheidungskosten zeigt eine allgemeine Aussage dazu:

C_{ij}	abgeschätzte Klasse	
richtige Klasse	geltende Integritätsbedingung	negierte Integritätsbedingung
geltende Integritätsbedingung	-5	5
negierte Integritätsbedingung	1	-1

Besonders kritisch ist also bei der Semantikakquisition der Fall, daß geltende Integritätsbedingungen als negierte Integritätsbedingungen abgeschätzt werden und deshalb nicht untersucht und entsprechend auch nicht gefunden werden. Weniger kritisch ist es, wenn negierte Integritätsbedingungen als geltende Integritätsbedingungen abgeschätzt werden und dadurch zusätzlich untersucht werden.

Die *Confusionsmatrix P* ergibt sich für die Semantikakquisition, indem ein Vergleich der Abschätzung von Integritätsbedingungen mit dem Ergebnis der Validierung der Integritätsbedingungen erfolgt. Die Matrix hat folgendes Aussehen, die Werte für diese ergeben sich für eine konkrete Datenbank:

P_{ij}	abgeschätzte Klasse	
richtige Klasse	geltende Integritätsbedingung	negierte Integritätsbedingung
geltende Integritätsbedingung	..	..
negierte Integritätsbedingung	..	..

Mit diesen Angaben und dem oben beschriebenen Algorithmus werden die Gewichte der Heuristikregeln schrittweise verändert. Nach jeder Änderung der bekannten Integritätsbedingungen ändert sich die Confusionsmatrix.

Die Gewichte werden durch die angegebene Matrix der Entscheidungskosten so angepaßt, daß möglichst *alle geltenden Integritätsbedingungen richtig abgeschätzt* werden, *auch wenn dadurch für die negierten Integritätsbedingungen falsche Abschätzungen* entstehen. Diese Voraussetzung sind für die Anwendung in einem Semantikakquisitionstool erwünscht.
Durch eine veränderte Matrix der Entscheidungskosten läßt sich das *Ziel des Lernens* verändern, falls das für eine andere Anwendung erwünscht ist.

5.4 Suche nach Kandidaten für funktionale Abhängigkeiten

Man kann sowohl im Datenbank-Entwurf als auch im Reverse-Engineering von Datenbanken nicht davon ausgehen, daß sich alle Relationen-Schemata einer relationalen Datenbank in 3. Normalform oder Boyce-Codd-Normalform befinden. Es können in Relationen-Schemata auch funktionale Abhängigkeiten, die nicht aus Schlüsseln resultieren, auftreten. Diese muß man ebenfalls finden, da sie für die Entwurfsaufgaben Normalisierung und Umstrukturierung benötigt werden. Die Anzahl der zu überprüfenden, noch unbekannten Abhängigkeiten eines Relationen-Schemas kann sehr groß sein, deshalb müssen diese vor der Diskussion abgeschätzt werden, um plausible Kandidaten für geltende funktionale Abhängigkeiten zu finden.

5.4.1 Heuristikregeln

In diesem Abschnitt werden verschiedene Heuristiken vorgestellt, mit denen Kandidaten für funktionale Abhängigkeiten abgeleitet werden können. Diese nutzen Wissen, das implizit in den Datenbanken dargestellt ist.

Auswertung der Daten

H_F1 In Relationen können abgeleitete Attribute auftreten. Die Werte dieser Attribute lassen sich aus den Werten anderer Attribute direkt berechnen. Mit der folgenden Heuristik wird versucht, funktionale Abhängigkeiten, die durch abgeleitete Attribute entstehen, zu ermitteln:
Gibt es in einer Relation mehrere Attribute ABC, die als Integer definiert sind, so wird überprüft, ob sich die Werte eines Attribute (C) aus zwei anderen Attribute (AB) durch Addition oder Multiplikation errechnen lassen. Ist das der Fall, so wird die funktionale Abhängigkeit $AB \longrightarrow C$ vermutet.

Das gleiche gilt, wenn die Werte eines Attributes B für alle Tupel einen festen Prozentsatz der Werte des Attributes A darstellen. In diesem Fall werden $A \longrightarrow B$ und $B \longrightarrow A$ Kandidaten für funktionale Abhängigkeiten.

H_F2 Man kann aus Daten einer Relation negierte funktionale Abhängigkeiten ableiten, dieses wurde im Kapitel 3 gezeigt. Über alle unbekannten funktionalen Abhängigkeiten können noch keine Aussagen durch die Daten getroffen werden. Sie können *geltende funktionale Abhängigkeiten* sein. Es können aber auch *negierte funktionale Abhängigkeiten* sein, die noch nicht erkannt wurden, weil in den Daten keine Tupel vorhanden sind, die die entsprechende funktionale Abhängigkeit widerlegen.

Man kann versuchen, diese beiden Fälle durch folgende Heuristik zu unterscheiden: Es wird untersucht, welche Attributmengen *gleiche Einträge* bei mehreren Tupeln aufweisen. Dieses Vorgehen soll an einem Beispiel erläutert werden.

Beispiel 5.1:

Gegeben sei eine Relation Personen:

Personennummer	Nachname	Vorname	Wohnort	PLZ	Straße	Nummer
271263	Meier	Klaus	Rostock	18119	Kirchenstr.	20
218263	Meier	Ilona	Berlin	10249	Mollstr.	149
948547	Mueller	Karl	Berlin	12621	Mittelweg	281
323983	Schmidt	Klaus	Rostock	18055	Gerberbruch	30
239283	Weber	Peter	Rostock	18055	Gerberbruch	15
043984	Lehmann	Iris	Rostock	18055	Grubenstr	2

In diesem Beispiel treten bei (Wohnort PLZ Straße) 2 gleiche Einträge und bei (Wohnort PLZ) 3 gleiche Einträge in der Relation auf. Man kann vermuten, daß zwischen diesen Attributen funktionale Abhängigkeiten gelten. Aus dieser Beispielrelation lassen sich also folgende *Kandidaten für funktionale Abhängigkeiten* ableiten:

(Wohnort	$\stackrel{?}{\longrightarrow}$	PLZ)	-	3 gleiche Einträge
PLZ	$\stackrel{?}{\longrightarrow}$	Wohnort	-	3 gleiche Einträge
(Wohnort	$\stackrel{?}{\longrightarrow}$	Straße)	-	2 gleiche Einträge
(PLZ	$\stackrel{?}{\longrightarrow}$	Straße)	-	2 gleiche Einträge
(Straße	$\stackrel{?}{\longrightarrow}$	Wohnort)	-	2 gleiche Einträge
Straße	$\stackrel{?}{\longrightarrow}$	PLZ	-	2 gleiche Einträge
(Wohnort PLZ	$\stackrel{?}{\longrightarrow}$	Straße)	-	2 gleiche Einträge
Wohnort Straße	$\stackrel{?}{\longrightarrow}$	PLZ	-	2 gleiche Einträge
PLZ Straße	$\stackrel{?}{\longrightarrow}$	Wohnort	-	2 gleiche Einträge

Die eingeklammerten Kandidaten sind keine unbekannten Abhängigkeiten, durch die Daten der angegebenen Relation Personen können diese als negierte funktionale Abhängigkeiten abgeleitet werden.

Die Heuristik liefert bei größeren Relationen *differenziertere Bewertungen* für die Kandidaten für funktionale Abhängigkeiten. Je mehr gleiche Werte in den Relationen auftreten, desto wahrscheinlicher müssen funktionale Abhängigkeiten gelten, da es anzunehmen ist, daß diese gleichen Werte nicht zufällig auftreten.

Mit dieser Methode kann man nur *Kandidaten* für funktionale Abhängigkeiten finden, die *nicht aus dem Schlüssel resultieren.*

Auswertung von Transaktionen

H_F3 Aus Update-Operationen einer Datenbank lassen sich ebenfalls Kandidaten für funktionale Abhängigkeiten erkennen.
Die Attribute einer Update-Operation, die zur Identifikation der zu ändernden Tupel verwendet wurden, bilden dabei die linke Seite der Abhängigkeit, die Attribute, die in der Operation geändert wurden, die rechte Seite.
Diese abgeleitete funktionale Abhängigkeit muß in der Relation gegolten haben, auf die die Transaktion angewendet worden ist. Man kann deshalb vermuten, daß diese funktionale Abhängigkeit auch allgemeingültig ist, sie muß jedoch ebenfalls validiert werden.

Auswertung struktureller Merkmale

H_F4 Meist werden inhaltlich zusammenhängende Attribute auch im Zusammenhang eingegeben und stehen deshalb nebeneinander in der Relation. Man kann deshalb davon ausgehen, daß zwischen Attributen, die dicht nebeneinander in der Relation auftreten, eher funktionale Abhängigkeiten auftreten als zwischen Attributen, die in der Relation weit auseinander liegen. Diese Heuristik kann nur im Zusammenhang mit anderen Heuristikregeln verwendet werden, da die Ergebnisse zu vage und zu wenig differenzierend sind.

Auswertung von strukturellen und semantischen Merkmalen

H_F5 Für eine zu untersuchende Abhängigkeit wird überprüft, wie die Abhängigkeiten in der Nähe dieser Abhängigkeit aussehen. Dieses soll zunächst an einem Beispiel dargestellt werden.

Gegeben sei eine Relation *Person = (Personennummer, Nachname, Vorname, Wohnort, PLZ, Straße, Nummer)*. Die negierten funktionalen Abhängigkeiten *Nachname* $\not\rightarrow$ *Wohnort* und *Nachname* $\not\rightarrow$ *Straße* seien bekannt.
Aus diesen negierten funktionalen Abhängigkeiten und der Reihenfolge der Attribute in der Datenbank ist zu vermuten, daß auch *Nachname* $\not\rightarrow$ *PLZ* gilt.

Soll die Abhängigkeit zwischen der Attributmenge X und dem Attribut Y festgestellt werden ($X \xrightarrow{?} Y$), so wird in dieser Prozedur untersucht, wie die Abhängigkeiten

$$X \xrightarrow{?} (Y-2) \qquad X \xrightarrow{?} (Y-1) \qquad X \xrightarrow{?} (Y+1) \qquad X \xrightarrow{?} (Y+2)$$

aussehen, wobei mit $(Y-1)$ das Attribut gemeint ist, das links neben dem Attribut Y in der Relation steht usw.
Es müssen dabei triviale funktionale Abhängigkeiten ausgeschlossen werden d.h, ist die rechte Seite der zu überprüfenden Abhängigkeit in der linken Seite enthalten, so kann daraus keine Aussage getroffen werden.

Sind die Abhängigkeiten in der Nähe überwiegend funktionale Abhängigkeiten, so deutet das darauf, daß die untersuchte Abhängigkeit ebenfalls funktional ist und umgekehrt.

5.4.2 Wichtung der Heuristiken

Die im vorherigen Abschnitt aufgezählten Heuristikregeln geben Hinweise auf geltende funktionale Abhängigkeiten. Dabei liefern die *Heuristikregeln H_F1 - H_F3 wesentlich zuverlässigere*

Ergebnisse als die beiden anderen Heuristiken. Die folgende Übersicht zeigt, welche Heuristikregeln für die meisten Anwendungen am aussagekräftigsten sind:

besonders aussagekräftig	H_F1, H_F3
aussagekräftig	H_F2, H_F5
nur im Zusammenhang mit anderen Heuristiken sinnvoll	H_F4

Die Gewichte der Heuristiken richten sich nach dieser Übersicht.

Für die Abschätzung der Plausibilität wird wieder eine gewichtete Summe verwendet.

$$Pl(X \longrightarrow Y) := \sum_{i=1}^{5} (w_i * H_F i(X, Y))$$

$H_F i(X, Y) \in [0..1]$: Ergebnis der Heuristikregel $H_F i$ für die Attribute (X, Y)
$w_i \in [0, 1]$ und $w_1 + .. + w_5 = 1$

Negierte funktionale Abhängigkeiten sind als Negation der funktionalen Abhängigkeiten definiert, deshalb können die Heuristiken auch zur Abschätzung einer negierten funktionalen Abhängigkeit verwendet werden.

$$Pl(X \not\longrightarrow Y) := 1 - \sum_{i=1}^{5} w_i * H_F i(X, Y)$$

Auf diese Weise erhält man gewichtete Kandidaten für funktionale und negierte funktionale Abhängigkeiten.

5.5 Suche nach Analoga

Inklusionsabhängigkeiten sind beim *Reverse-Engineering* von relationalen Datenbanken erforderlich, um Verbindungen zwischen den einzelnen Relationen zu finden. Im *Datenbank-Entwurf* kann man durch ermittelte Inklusionsabhängigkeiten in relationalen Datenbanken die eingetragenen Pfade in der zugehörigen Datenbank im Entity-Relationship Modell überprüfen und ggf. Pfade ergänzen.

Im folgenden Abschnitt wird eine Methode vorgestellt, mit der Kandidaten für Inklusions- und Exklusionsabhängigkeiten einer relationalen Datenbank ermittelt werden. Als Voraussetzung zur Suche nach Inklusions- und Exklusionsabhängigkeiten muß man alle Attribute $(R.X, S.Y)$ innerhalb einer Datenbank suchen, die die gleiche Bedeutung haben. Diese werden im folgenden als *Analoga* $(R.X \approx S.Y)$ bezeichnet. Es werden dabei sowohl die Problemfälle Synonyme als auch Homonyme berücksichtigt. *Synonyme* sind Attribute, die die gleiche Bedeutung haben, aber unterschiedliche Attributnamen besitzen. *Homonyme* sind Attribute mit gleichem Attributnamen aber unterschiedlicher Bedeutung. Die Menge der Analoga umfaßt Attribute, die den gleichen Attributnamen und gleiche Bedeutung haben sowie die Synonyme. Homonyme sind nicht in der Menge der Analoga enthalten.

Inklusions- und Exklusionsabhängigkeiten sind nur über Analoga sinnvoll, deshalb müssen diese ermittelt werden, um sinnvolle und plausible Kandidaten für Inklusion- und Exklusionabhängigkeiten zu finden. *Notwendige Voraussetzung* für die Ermittlung von Analoga sind *Attribute mit gleichen Typen und gleichen oder ähnlichen Längen.* Die Anzahl der Attributpaare mit dieser Eigenschaft kann in einer Datenbank sehr groß sein kann, deshalb müssen Heuristikregeln angewendet werden, um differenziertere Angaben über die Menge der Analoga abzuleiten.

5.5.1 Heuristikregeln

Folgende Heuristiken geben an, daß die Attribute X des Relationen-Schemas R und Y des Relationen-Schemas S Analoga einer Datenbank sind:

Strukturelle Merkmale

H_A1 *Gleiche Attributnamen* (X, Y) oder gleiche Teilstrings in Attributnamen weisen auf Analoga hin.

H_A2 Gleiche Typen der Attribute und gleiche oder ähnlich definierte Längen sind ein Hinweis auf eine gleiche Bedeutung der Attribute. Die Information, daß zwei Attribute die gleiche oder eine ähnliche Anzahl verschiedener Werte annehmen (in Abschnitt 3.3 eingeführt), ist eine Präzisierung der Typenübereinstimmung. Diese Heuristikregel ist besonders in Kombination mit anderen Heuristiken sinnvoll.

Auswertung der Daten

H_A3 *Gleiche Einträge in den Daten* sind ein weiterer Hinweis auf Analoga. In die Menge der Analoga sollen auch Attributpaare aufgenommen werden, über denen Exklusionsabhängigkeiten existieren. Deshalb sind gemeinsame Einträge oder Teilmengenbeziehungen zwischen den Werten ein Hinweis auf Analoga, sie sind jedoch keine notwendige Voraussetzung zur Bestimmung von Analoga.

Auswertung der Semantik

H_A4 Wurden zwischen zwei Relationen-Schemata R und S bereits Analoga gefunden, so können zwischen diesen auch weitere Analoga auftreten, da es sein kann, daß größere gleiche Attributsequenzen in diesen beiden Relationen-Schemata definiert sind.

H_A5 Sind die Attributmengen X und Y beide Schlüssel in R bzw. S, oder X und Y sind nicht Teil eines Primärschlüssels, so ist das ein weiterer (allerdings sehr vager) Hinweis auf gleiche Bedeutung der Attribute.

H_A6 Treten innerhalb von zwei Attributmengen in verschiedenen Relationen-Schemata funktionale Abhängigkeiten auf, so sind diese ein zusätzlicher Hinweis auf Analoga.
Beispiel: Sind in zwei Relationen-Schemata Personen und Studenten die funktionalen Abhängigkeiten Personen: Wohnort $\longrightarrow$ Vorwahl und Studenten: Wohnort $\longrightarrow$ Ortsvorwahl bekannt und man hat Personen.Wohnort und Studenten.Wohnort bereits als Analoga

ermittelt, so sind diese funktionalen Abhängigkeiten ein zusätzlicher Hinweis auf die Analoga Personen.Vorwahl und Studenten.Ortsvorwahl.
Diese Heuristik wird jedoch nur betrachtet, wenn bereits weitere Hinweise auf diese Analoga vorlagen, sie ist nur in Ergänzung zu den anderen Heuristiken sinnvoll.

H_A7 Ist eine Inklusionsabhängigkeit $R.X \subseteq S.Y$ bekannt, so sind $R.X$ und $S.Y$ Analoga.

H_A8 Sind Inklusionsabhängigkeiten $R.X \subseteq T.Z$ und $S.Y \subseteq T.Z$ bekannt, so sind $R.X$ und $S.Y$ Analoga.

Auswertung der Transaktionen

H_A9 Sind Transaktionen auf einer Datenbank bekannt und es werden in dieser Transaktionen Attribute der Datenbank miteinander verglichen, so kann man ableiten, daß diese Analoga sind.

Diese Heuristikregeln geben meist recht vage Hinweise auf Analoga in der Datenbank.

5.5.2 Wichtung der Heuristikregeln

Die einzelnen Heuristikregeln werden gewichtet. Dabei sollen Heuristikregeln, die plausiblere Hinweise auf Analoga geben, höher gewichtet werden als die anderen Heuristikregeln. Die folgende Übersicht zeigt die Bedeutung der Heuristikregeln:

besonders aussagekräftig	H_A7, H_A8
aussagekräftig	H_A1, H_A2, H_A3, H_A9
nur im Zusammenhang mit anderen Heuristiken sinnvoll	H_A4, H_A5, H_A6

Die einzelnen Heuristikregeln sollen gegeneinander gewichtet werden, um Analoga abzuleiten. Dabei wird wieder eine einfache Abschätzung verwendet. Die Wichtung der Analoga soll höher sein, wenn mehr Heuristikregeln erfüllt sind.

Dieser Forderung wird eine gewichtete Summe gerecht, die eine Abschätzung der Plausibilität für Analoga darstellt:

$$Pl(X \approx Y) := \sum_{i=1}^{9} w_i * H_Ai(X,Y)$$

$H_Ai(X,Y) \in [0..1]$ ist das Ergebnis der Heuristikregel H_Ai für die Attribute X, Y
$w_i \in [0,1]$ und $w_1 + .. + w_9 = 1$

Für Sequenzen von analogen Attributen wird darauf basierend folgende Abschätzung vorgenommen:

$$Pl(\langle X_1..X_n \rangle \approx \langle Y_1..Y_n \rangle) := \frac{1}{n} * \sum_{i=1}^{n} Pl(X_i \approx Y_i)$$

Mit dieser Abschätzung werden nicht nur Synonyme als Analoga ermittelt, es werden auch Homonyme ausgeschlossen bzw. mit schwächerer Wichtung versehen, da sich die Identifizierung gleicher Attribute nicht nur auf die Attributnamen stützt. So werden zum Beispiel Homonyme mit unterschiedlichen Typen ausgeschlossen. Homonyme, bei denen keine gemeinsamen Daten auftreten, gehen nur mit niedriger Heuristikwertung in die Analoga ein.

Die Suche nach Analoga über *sehr großen Datenbanken* ist nicht vollständig möglich, da die Anzahl der zu untersuchenden Attributpaare zu groß wäre. Eine *sinnvolle Einschränkung* dieses Problems ist die Ermittlung von *Relationenclustern* (inhaltlich zusammenhängende Relationen-Schemata eines Datenbank-Schemas), die in Abschnitt 7.5 vorgestellt wird, und die Suche nach Analoga innerhalb dieser Cluster.

5.6 Suche nach Kandidaten für Inklusions- und Exklusionsabhängigkeiten

Inklusions- und Exklusionsabhängigkeiten sind nur über Attributen, die als Analoga ermittelt wurden, sinnvoll. Der Suchraum zur Ermittlung von Inklusions- und Exklusionsabhängigkeiten wird dadurch eingeschränkt. In diesem Abschnitt wird gezeigt, wie man *Kandidaten für Inklusions- und Exklusionsabhängigkeiten* aus der Menge der *Analoga* ableiten kann.

Nach [MaR92] sind die meisten in der Praxis auftretenden Inklusionsabhängigkeiten schlüsselbasiert. Eine Inklusionsabhängigkeit $R.X \subseteq S.Y$ heißt *schlüsselbasiert*, wenn Y Schlüssel in S ist. Solche Inklusionsabhängigkeiten werden im *Entity-Relationship Modell strukturell* als Pfade dargestellt. Bei der Übersetzung in relationale Datenbanken wird der Schlüssel Y aus S als Fremdschlüssel in R mit der Bezeichnung X aufgenommen. Obwohl die meisten Inklusionsabhängigkeiten schlüsselbasiert sind, kann man die Suche nach Inklusionsabhängigkeiten nicht auf Analoga beschränken, bei denen mindestens eine Attributsequenz Schlüssel ist, da gefundene Inklusionsabhängigkeiten, die nicht schlüsselbasiert sind, wichtige Strukturänderungen nach sich ziehen können.

5.6.1 Suche nach Inklusions- und Exklusionsabhängigkeiten durch Auswertung von Analoga und Daten

Sind Analoga $R.X \approx S.Y$ ermittelt worden und sind keine Inklusions- oder Exklusionsabhängigkeiten zwischen den Relationen-Schemata R und S bekannt, so kann man durch Auswertung der vorhandenen Daten Kandidaten für Inklusions- oder Exklusionsabhängigkeiten ermitteln.

- Aus den Daten einer Datenbank kann man *Kandidaten für Inklusionsabhängigkeiten* ableiten. Sind die Werte des einen Attributes $R.X$ *vollständig* in den Werten eines anderen Attributes $S.Y$ *enthalten*, so kann man hier die Inklusionsabhängigkeit $R.X \subseteq S.Y$ vermuten. Die Plausibilität dieser Inklusionsabhängigkeit läßt sich durch folgende Formel abschätzen:

  $$Pl(R.X \subseteq S.Y) := Pl(R.X \approx S.Y)$$

  Sind die Werte von $S.Y$ Teilmenge der Werte von $R.X$, so kann man hier die Inklusionsabhängigkeit $S.Y \subseteq R.X$ vermuten. Hierbei gilt folgende Plausibilität:

$$Pl(S.Y \subseteq R.X) := Pl(R.X \approx S.Y)$$

- Wenn in den Attributen $R.X$ und $S.Y$ *gleiche Werte auftreten*, aber *keine Teilmengenbeziehungen* existieren, so kann hier eine der *Inklusionsabhängigkeiten* $R.X \subseteq S.Y$ oder $S.Y \subseteq R.X$ bestehen. Die Plausibilität dieser Kandidaten läßt sich nach folgenden Formeln abschätzen:

$$Pl(R.X \subseteq S.Y) := Pl(R.X \approx S.Y) * \frac{Anz(R.X \subseteq S.Y)}{\text{Tupelanzahl in } R}$$

- $Anz(R.X \subseteq S.Y)$ - Anzahl der Werte in $R.X$, die auch in $S.Y$ auftreten

$$Pl(S.Y \subseteq R.X) := Pl(S.Y \approx R.X) * \frac{Anz(S.Y \subseteq R.X)}{\text{Tupelanzahl in } S}$$

- $Anz(S.Y \subseteq R.X)$ - Anzahl der Werte in $S.Y$, die auch in $R.X$ auftreten

- Ein *Kandidat für eine Exklusionsabhängigkeit* ($R.X \mid\mid S.Y$) wird abgeleitet, wenn in den Instanzen *keine gemeinsamen Einträge* in $R.X$ und $S.Y$ auftreten.

$$Pl(R.X \mid\mid S.Y) := Pl(R.X \approx S.Y)$$

Im nächsten Abschnitt wird gezeigt, wie die Ermittlung von Kandidaten für Inklusions- und Exklusionsabhängigkeiten erfolgen kann, wenn zwischen den Relationen-Schemata bereits Inklusions- oder Exklusionsabhängigkeiten bekannt sind.

5.6.2 Suche nach Inklusions- und Exklusionsabhängigkeiten durch Auswertung von Analoga und Integritätsbedingungen

Wurde zwischen den Attributmengen von zwei Relationen-Schemata $R.X$ und $S.Y$ eine Inklusions- oder Exklusionsabhängigkeit gefunden, so sind zwischen den betreffenden Attributmengen die semantischen Beziehungen geklärt. Bei der Ableitung von *negierten Inklusionsabhängigkeiten bzw. Exklusionsabhängigkeiten* muß jedoch eine *Nachfrage zur Präzisierung* der Integritätsbedingungen erfolgen.

- Hat man bereits eine negierte Inklusionsabhängigkeit ($R.X \not\subseteq S.Y$) abgeleitet, so muß auch untersucht werden, ob zwischen den Knoten sogar eine *Exklusionsabhängigkeit* besteht. Die Plausibilität dieser Exklusionsabhängigkeit läßt sich folgendermaßen abschätzen:

$$Pl(R.X \mid\mid S.Y) := Pl(R.X \approx S.Y)$$

- Ist eine negierte Exklusionsabhängigkeit ($R.X \not\parallel S.Y$) bekannt, so muß auch eine *Inklusionsabhängigkeit* zwischen den Knoten untersucht werden ($R.X \subseteq S.Y$ oder $S.Y \subseteq R.X$). Die Plausibilitäten dieser Kandidaten ergibt sich auch hier aus der Plausibilität der Analoga und den vorliegenden Daten.

$$Pl(R.X \subseteq S.Y) := Pl(R.X \approx S.Y) * \frac{Anz(R.X \subseteq S.Y)}{\text{Tupelanzahl in } R}$$

- $Anz(R.X \subseteq S.Y)$: Anzahl der Werte in $R.X$, die auch in $S.Y$ auftreten

$$Pl(S.Y \subseteq R.X) := Pl(S.Y \approx R.X) * \frac{Anz(S.Y \subseteq R.X)}{\text{Tupelanzahl in } S}$$

- $Anz(S.Y \subseteq R.X)$: Anzahl der Werte in $S.Y$, die auch in $R.X$ auftreten

Sind zwischen den Relationen-Schemata R und S bereits *Integritätsbedingungen bekannt*, die über anderen Attributen der Relationen-Schemata definiert sind, so geht man folgendermaßen mit weiteren Analoga $R.X \approx S.Y$ vor:

- Wurden zwischen den Relationen-Schemata R und S bereits Inklusions- oder Exklusionsabhängigkeiten gefunden, so gelten zwischen weiteren Analoga ($R.X \approx S.Y$) über diesen Relationen-Schemata meist die gleichen Abhängigkeiten, deshalb werden diese in entsprechender Form in die Menge der Kandidaten aufgenommen.

Die auf diese Weise ermittelten Kandidaten für Inklusions- und Exklusionsabhängigkeiten müssen validiert werden.

5.6.3 Suche nach Inklusionsabhängigkeiten durch Auswertung von Transaktionen

Informationen über geltende Inklusionsabhängigkeiten sind zur Überwachung von Transaktionen erforderlich. Bekannte Transaktionen müssen diesen Integritätsbedingungen entsprechen. Umgekehrt kann man aus bekannten Transaktionen Vermutungen über geltende Inklusionsabhängigkeiten ableiten. Aufgrund dieser Annahme kann man Kandidaten für Inklusionsabhängigkeiten ermitteln, ohne entsprechende Analoga zu kennen.

Dieses Vorgehen soll hier anhand eines abstrakten Beispiels gezeigt werden. Es soll die Inklusionsabhängigkeit $R.X \subseteq S.Y$ in einer Datenbank gelten. Bestimmte Transaktionen ziehen dann folgende weitere Transaktionen nach sich:

Eine *Insert-Operation* in R zieht eine Überprüfung des Vorhandenseins eines Eintrages in S oder eine Insert-Operation in S nach sich.
Eine *Delete-Operation* in S bedingt ggf. eine Delete-Operation in R, falls dort ein entsprechendes Tupel vorhanden ist.

Eine *Update-Operation* in R kann auch ein Update in S bewirken.

Man kann aus solchen Transaktionen Schlußfolgerungen über geltende Inklusionsabhängigkeiten ziehen. Es wird dazu in bekannten Transaktionen nach Mustern in der oben dargestellten Form gesucht. Problematisch ist dabei, daß zwischen den Transaktionen weitere Transaktionen eingeschlossen sein können, damit ist die Analyse der Transaktionen schwieriger.

Diese Informationen sind jedoch schwer zu ermitteln und sie sind vage und können ebenso wie die anderen Heuristikregeln nur Kandidaten für Inklusionsabhängigkeiten liefern.

5.7 Kardinalitäten

Kardinalitäten sind Integritätsbedingungen, die im Entity-Relationship Modell strukturell dargestellt werden. Im relationalen Modell sind sie nicht direkt darstellbar. Man kann jedoch Kardinalitäten anhand von Daten im relationalen Datenmodell diskutieren und die abgeleiteten Informationen über Kardinalitäten speichern. Voraussetzung zur Ermittlung von Kardinalitäten ist das Vorhandensein von Fremdschlüsselbeziehungen im relationalen Modell.

In diesem Abschnitt soll gezeigt werden, wie die Fremdschlüsselbeziehungen im relationalen Datenmodell gefunden werden, durch die angegeben wird, *an welchen Stellen Kardinalitäten* ermittelt werden müssen. Eine *Abschätzung der konkreten Werte* der Kardinalitäten erfolgt *durch die Heuristiken nicht.*

5.7.1 Pfadinformationen im Entity-Relationship Modell

Hat man im Datenbank-Entwurf eine Datenbank im Entity-Relationship Modell, so kennt man die Pfade innerhalb der Datenbank. Bei der Übersetzung dieser Datenbank in eine relationale Datenbank werden aufgrund dieser Pfade Fremdschlüssel eingeführt. Durch diese Fremdschlüsselbeziehungen weiß man, an welchen Stellen im relationalen Datenmodell Kardinalitäten ermittelt werden müssen.

Die Plausibilität der Kardinalitäten ist dann:

$$Pl(\mathsf{card}(R, S)) := 1,$$

da für diese Fälle unbedingt die Kardinalitäten im Dialog mit dem Benutzer ermittelt werden müssen.

5.7.2 Suche nach Fremdschlüsselbeziehungen

Es kann sein, daß im Entity-Relationship Modell Pfade vergessen wurden, man kann versuchen, in der relationalen Datenbank weitere Fremdschlüsselbeziehungen zu finden, über denen ebenfalls Kardinalitäten ermittelt werden müssen.
Ist kein Entity-Relationship Modell bekannt (wie z.B. beim Reverse-Engineering von relationalen Datenbanken), so muß man versuchen, Fremdschlüsselbeziehungen in der Datenbank zu

finden. Über diesen Fremdschlüsselbeziehungen müssen die Kardinalitäten durch Diskussion von Beispieldatenbanken ermittelt werden.

$$Pl(\mathsf{card}(R,S)) := Pl(R.X \subseteq S.Y) * Pl(Y \text{ ist Schlüssel})$$

Damit lassen sich die Probleme der Suchraumeinschränkung zur Ermittlung von Kardinalitäten auf die Suche nach Fremdschlüsseln, also *Schlüssel* und *Inklusionsabhängigkeiten* zurückführen.

5.7.3 Direkte Erfragung von Fremdschlüsseln

Es kann trotz des Einsatzes verschiedener Heuristiken passieren, daß ein Relationen-Schema oder Mengen von Relationen-Schemata auftreten, für die *keine Verbindung* zu anderen Relationen-Schemata des Datenbank-Schemas gefunden werden konnte. Dafür kann es verschiedene Ursachen geben.

1. Es kann sein, daß zwischen den Relationen-Schemata keine Fremdschlüsselbeziehungen existierten und die Verbindung zwischen den Relationen-Schemata über Anwendungsprogramme hergestellt wurden.
2. Es ist möglich, daß vorhandene Analoga mit den Heuristiken nicht gefunden werden konnten und damit auch keine Inklusionsabhängigkeiten gefunden wurden.
3. Weiterhin kann die Datenbank noch unvollständig sein, im Entwurf sind erst Teildatenbanken ohne Zusammenhang zueinander erstellt worden.

In diesem Fall muß eine direkte Befragung des Benutzers nach Fremdschlüsseln, über die eine Verbindung hergestellt werden soll, erfolgen. Die Plausibilität dieser erfragten Verbindungen wird mit

$$Pl(\mathsf{card}(R,S)) := 1$$

gesetzt, da für diese expliziten Angaben des Benutzers die Ermittlung von Kardinalitäten in jedem Fall erfolgen muß.

Durch diese Methoden werden Kandidaten für Kardinalitäten gefunden, diese Kandidaten geben nur an, an welchen Stellen in der relationalen Datenbank Kardinalitäten ermittelt werden müssen. Die Erfragung der konkreten Werte der Kardinalitäten wird in Kapitel 9 gezeigt.

5.8 Verwendung der ermittelten Kandidaten

Die ermittelten Kandidaten für geltende und negierte Integritätsbedingungen liefern eine Abschätzung der noch unbekannten Integritätsbedingungen. Man kann durch die Heuristikregeln keine Integritätsbedingungen ableiten, wenn man verhindern will, daß falsche Integritätsbedingungen ermittelt werden. Man kann jedoch plausible Kandidaten für Integritätsbedingungen finden.

Durch die Abschätzung *sortieren* die gewichteten Heuristikregeln die *Menge der unbekannten Integritätsbedingungen* und ermöglichen damit eine gezielte Erfragung bestimmter plausibler Zusammenhänge.

Im folgenden Kapitel wird eine weitere Methode vorgestellt, mit der plausible Kandidaten für Integritätsbedingungen gefunden werden können.

Kapitel 6

Wiederverwendung von vorhandenen Datenbank-Schemata

Die Bedeutung der Wiederverwendung von Software wird in vielen Veröffentlichungen betont ([Che94], [BeE95], [Web93]). Eine dieser soll hier als Motivation zitiert werden.
"Bei der Erstellung komplexer Software spielt die Wiederverwendung vorhandener Bestandteile eine besonders große Rolle, da hierdurch sowohl die *Software-Qualität gesteigert*, als auch der gesamte *Erstellungs- und Wartungsaufwand erheblich reduziert* werden kann." [BeE95]

In diesem Kapitel soll erläutert werden, wie durch die Wiederverwendung bekannter Datenbanken plausible Kandidaten für geltende Integritätsbedingungen in einer aktuellen Datenbank ermittelt werden können. Die Wiederverwendung von bekannten Datenbanken kann zur Unterstützung aller Entwurfsentscheidungen im Datenbank-Entwurf verwendet werden. Für die vorliegende Arbeit ist nur die Semantikakquisition relevant, da das analoge Vorgehen zahlreiche weitere Anwendungsmöglichkeiten hat, sollen diese ebenfalls erläutert werden.

6.1 Zielstellung und allgemeines Vorgehen bei der Wiederverwendung von Datenbanken

Eine Wiederverwendung bereits bestehender Teile bietet sich besonders im *Datenbankbereich* an. Viele Vorgänge in der realen Welt sind in gleicher oder ähnlicher Form auf verschiedenen Einsatzgebieten von Datenbanken relevant, z.B. die Verwaltung von Personendaten, die Verwaltung von Adressen, Gehaltsabrechnung, Lagerungs-, Verkaufs- und Ausleihprozesse, Datumsangaben usw. In erstellten Datenbanken resultieren daraus gleiche oder ähnlich modellierte Teile, die gleiche oder ähnliche Integritätsbedingungen aufweisen, das gleiche Verhalten haben, über diesen ähnlichen Datenbanken werden häufig identische Transaktionen ausgeführt, usw.
Besonders, wenn Datenbanken für ein ähnliches Anwendungsgebiet erstellt werden, gibt es viele analoge Teile. Wünschenswert ist in solchen Fällen eine *Wiederverwendung von Informationen aus bereits existierenden Datenbank-Schemata*. Durch eine solche Wiederverwendung erreicht man, daß man Angaben, die bereits in ähnlicher Form in anderen Datenbanken vorhanden sind, z.B. die in Adreßinformationen geltenden funktionalen Abhängigkeiten oder die Verhaltensinformationen von Datumsangaben, nicht noch einmal spezifizieren muß. Auf diese Weise kann der Entwurf von neuen Datenbank-Schemata beschleunigt werden und Fehler dabei vermieden werden.

Datenbank-Schemata sind zur Wiederverwendung gut geeignet, da man hier auf *konzeptioneller oder logischer Ebene* eine *formale Beschreibung* ausnutzen kann. Nicht alles Hintergrundwissen liegt dabei formal und auswertbar vor, es gibt aber formale Angaben zur Struktur, Semantik, zum Verhalten und zu Transaktionen, die teilweise auch miteinander vergleichbar sind.

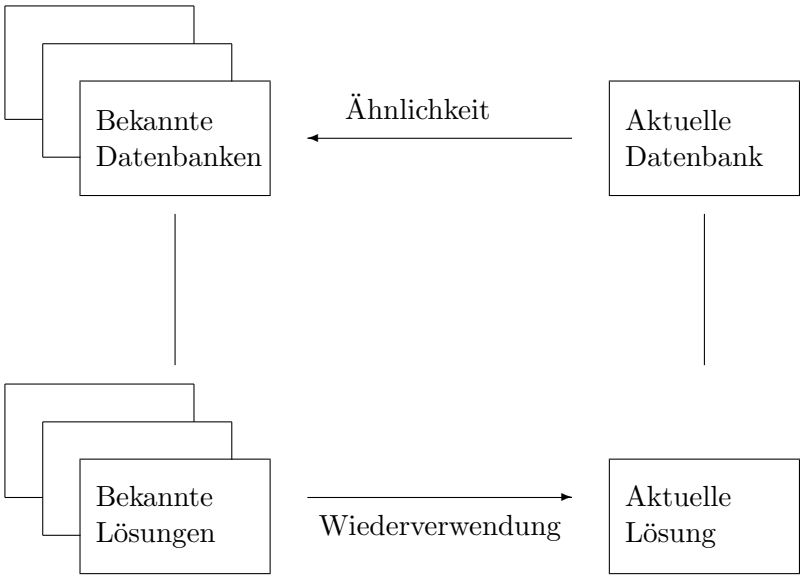


Abbildung 6.1: Wiederverwendung bekannter Datenbankinformationen bei Entwurfsaufgaben

In Abbildung 6.1 wird gezeigt, wie Entscheidungen beim Datenbank-Entwurf durch die Wiederverwendung vorhandener Datenbanken unterstützt werden können. Es sind dabei Datenbank-Schemata, die in Bibliotheken oder einer anderen Form gespeichert sind, vorhanden, zu diesen Datenbank-Schemata sind bestimmte Lösungen für die einzelnen Entwurfsschritte bekannt. Bekannt ist weiterhin ein aktuelles Datenbank-Schema, das erweitert oder vervollständigt werden soll. Gesucht ist die Lösung einer Entwurfsaufgabe für das aktuelle Datenbank-Schema. Dazu muß ermittelt werden, welche analogen Lösungen aus den bereits bekannten Datenbank-Schemata anwendbar sind. Es wird bestimmt, welches Datenbank-Schema oder welche Datenbank-Schemata in der Bibliothek Ähnlichkeiten zum aktuellen Datenbank-Schema aufweisen. Werden solche ähnlichen Datenbank-Schemata gefunden, so wird versucht, die Lösungen bestimmter Entwurfsaufgaben auf diesen bekannten Datenbank-Schemata für das aktuelle Datenbank-Schema wiederzuverwenden und anzupassen.

Schwerpunkt dieser Arbeit ist die *Akquisition von Integritätsbedingungen.* Abbildung 6.2 veranschaulicht, wie die Wiederverwendung für diese Aufgabe eingesetzt werden kann. Bekannt sind dabei Datenbanken mit zugehörigen Integritätsbedingungen. Weiterhin existiert eine aktuelle Datenbank, zu der die Semantik ermittelt werden soll. Findet man zu dieser aktuellen Datenbank ähnliche Datenbanken, so lassen sich zugehörige Integritätsbedingungen in die aktuelle Datenbank übernehmen. Da der Prozeß auf vagen Informationen beruht, lassen sich auf diese Weise nur Kandidaten für Integritätsbedingungen finden.

Die Wiederverwendung existierender Datenbank-Schemata zur Lösung von Entwurfsentscheidungen für neue Datenbanken weist viele Analogien zum fallbasierten Schließen auf.

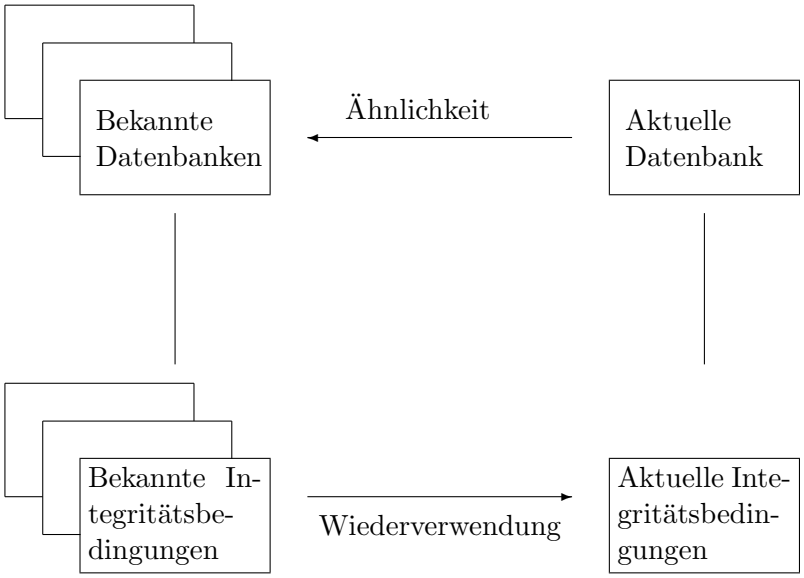


Abbildung 6.2: Wiederverwendung bekannter Datenbankinformationen bei der Semantikakquisition

Das fallbasierte Schließen wird in [Wil96] wie folgt erläutert: Die Lösung eines aktuellen Problems beginnt mit einer geeigneten Formulierung der Problemsituation, die dann mit den bereits bekannten Erfahrungen verglichen wird. Die bereits bekannte Lösung wird anschließend, in gegebenenfalls modifizierter Form, auf die aktuelle Situation übertragen. Somit ist die neue Problemstellung durch die bereits gemachten Erfahrungen gelöst. Auf diese Weise entsteht ein neuer Fall, der ebenfalls in der Fallbasis abgelegt werden kann, um so für zukünftige Problemstellungen zur Verfügung zu stehen.

Der Prozeß des fallbasierten Schließens besteht nach [LAB96] aus folgenden Teilaufgaben:

- retrieve – Zugriff auf den ähnlichsten Fall
- reuse – Wiederverwertung der Information dieses Falles für das aktuelle Problem
- revise – Bearbeitung der vorgeschlagenen Lösung
- retain – Abspeicherung von Teilen dieser Problemlöse-Episode für die Zukunft

Diese Phasen treten auch bei der Wiederverwendung von Datenbank-Schemata auf.

- retrieve – Suche nach ähnlichen Entities, Relationships und Teildatenbanken
- reuse – Übernahme von strukturellen Angaben, Integritätsbedingungen, Transaktionen, Beispieldaten, usw.
- revise – Validierung der übernommenen Informationen
- retain – Speicherung der vervollständigten Datenbanken
  – Aufbau von Bibliotheken

Die Aufgaben der einzelnen Phasen bei der Wiederverwendung werden in diesem Kapitel erläutert. Lösungen und Beispiele werden angegeben. Abschnitt 6.2 faßt noch einmal die Vorteile der Wiederverwendung von Datenbanken zusammen. In Abschnitt 6.3 wird ein kurzer

Überblick über relevante Veröffentlichungen zu diesem Thema gegeben.
Das Vorgehen bei der Wiederverwendung wird in den anschließenden Abschnitten ausführlich erläutert. Es orientiert sich an den oben dargestellten vier Teilaufgaben. Es müssen zunächst *ähnliche Teile in Datenbank-Schemata identifiziert* werden (Retrieve: Abschnitt 6.4). Im folgenden Abschnitt wird erläutert, wie gefundene ähnliche Datenbankteile zur Unterstützung des Entwurfes beziehungsweise Wiederentwurfes von Datenbanken angewendet werden können (Reuse: 6.5). Anschließend wird erläutert, welche Operationen ausgeführt werden müssen, um eine Anpassung und Validierung von übernommenen Informationen zu erreichen (Revise: 6.6). Im nächsten Abschnitt wird gezeigt, wie Bibliotheken zur Entwurfsunterstützung aufgebaut werden können und wie diese zur Unterstützung des Entwurfes neuer Datenbanken verwendet werden können (Retain: 6.7).

6.2 Vorteile der Wiederverwendung von Datenbankteilen

Sowohl im *Datenbank-Entwurf* als auch im *Re-Engineering* (bei dem neben dem Reverse-Engineering auch ein Entwurfsschritt zur Änderung und Erweiterung eines Datenbank-Schemas auftritt) kann die Wiederverwendung von bereits entworfenen Teilen einer Datenbank den *Entwurf beschleunigen*. Es können Fehler und Unvollständigkeiten im strukturellen Entwurf vermieden werden, da *Strukturangaben* aus einer bereits entworfenen Datenbank übernommen werden. *Integritätsbedingungen, Verhaltensinformationen, Transaktionen, Optimierungsschritte und Beispieldaten* können ebenfalls übernommen werden, wenn ähnliche Datenbanken existieren. Bereits entworfene Datenbanken sind in irgendeiner Form überprüft und korrigiert worden, man übernimmt also aus diesen *bewährte Lösungen* für die Entwurfsaufgaben. Es können durch existierende ähnliche Datenbanken wichtige Anhaltspunkte für den Entwurf einer Datenbank abgeleitet werden, die *Diskussion von Unvollständigkeiten* in Datenbanken kann dadurch *sinnvoll gesteuert* werden.

Intelligente Tools zur Unterstützung des Datenbank-Entwurfes müssen *sinnvolle Vorschläge für alle Entwurfsentscheidungen* unterbreiten. Solche Vorschläge zu generieren, ist für keine Entwurfsaufgabe trivial. Durch Auswertung von bereits bekannten Datenbank-Schemata findet man solche Vorschläge für Entwurfsaufgaben in einfacherer Form als durch eine allgemeine Bewertung von Datenbanken, durch die Auswertung von Hintergrundwissen oder durch die Erstellung von allgemeingültigen Regeln zum Generieren von Entwurfsentscheidungen. Da die Qualität solcher aus vorhandenen Datenbank-Schemata abgeleiteten Vorschläge mit zunehmender Anzahl der auswertbaren vorhandenen Datenbanken steigt, erreicht man eine Lernfähigkeit der Tools zur Unterstützung des Datenbank-Entwurfes.

Durch Vergleich von unvollständigen Informationen einer zu entwerfenden Datenbank und durch Auswertung von Bibliotheken mit vorhandenen Datenbanken und vollständigen Datenbankinformationen kann man Vorschläge für Entwurfsentscheidungen übernehmen und anpassen. Es ist also möglich, nur *durch strukturelle Vergleiche oder durch Vergleiche von unvollständigen Datenbank-Informationen weitere Vorschläge zur Struktur der Datenbank, begründete Kandidaten für Integritätsbedingungen, Verhaltensinformationen, Transaktionen, Optimierungsvorschläge, Testdaten, usw.* zu erhalten.

6.3 Literaturübersicht

Die Wiederverwendung bereits bestehender Komponenten zur Erstellung neuer Teile ist eine vielversprechende Methode zur Entwurfsunterstützung. Sie kann sowohl für die Softwareentwicklung als auch für den Datenbank-Entwurf angewendet werden. Diese Aufgabe ist jedoch kompliziert, da eine Analyse und der Vergleich von Software beziehungsweise Datenbanken notwendig ist. Es gibt einige Veröffentlichungen, in denen dieses Thema behandelt wird.

Storey, Dey, Goldstein, Chiang, Sundaresan: Tool zur Unterstützung des Datenbank-Entwurfes durch Auswertung vorhandener Datenbanken. In [SDG95] wird ein System vorgestellt, das den Entwurf von Datenbanken durch Ausnutzen von bereits bekannten Entwürfen unterstützt. Sind zu einem Gebiet noch keine Datenbanken vorhanden, so wird ein *allgemeines, einfach strukturiertes Betriebsmodell* eingesetzt. In einer *Wissensbasis* werden Informationen über Spezialisierungen, Synonyme usw. gespeichert.
Kernstück des Programms ist das *Auffinden von gleichen, ähnlichen oder unterschiedlichen Attributen, Entities, Relationships und Datenbanken.* Die Suche nach ähnlichen Einheiten stützt sich ausschließlich auf die Bezeichnungen (Auswertung gleicher Namen, gleicher Teilstrings in den Namen oder von explizit spezifizierten Synonymen). Die Methode wurde anhand von bekannten Beispielen aus der Datenbankliteratur getestet. Diese Beispiele sind alle von einer Gruppe von Entwerfern mit gleicher Erfahrung und gleichem Wissensstand und teilweise sogar von dem gleichen Entwerfer erstellt worden. Die Ergebnisse der vorgestellten Methode sind für diese speziellen Testfälle gut, jedoch kann dieses aufgrund der Auswahl der Testfälle keine allgemeingültige Aussage sein.
Die ermittelten gleichen Datenbanken werden verwendet, um Vorschläge zur Erweiterung von Datenbank-Entwürfen zu machen. Weiterhin werden sie benötigt, um *Abstraktionen als Lernprozeß* zu erreichen. Dabei werden aus mehreren Datenbanken zu einem Gebiet die gemeinsamen Teile herausgefiltert und als Modell für die weitere Verwendung zur Entwurfsunterstützung gespeichert.

In diesem Kapitel werden einige Vorgehensweisen aus dem Artikel übernommen, an den entsprechenden Stellen wird darauf hingewiesen. Die Heuristiken zur Suche nach gleichen Einheiten werden in diesem Kapitel wesentlich erweitert. Durch die hier vorgestellte Methode ist es möglich, universelle Konstrukte miteinander zu vergleichen, damit ist auch der Vergleich von strukturell unterschiedlich dargestellten Einheiten (Entities und Relationships untereinander) möglich. Auch beim Lernschritt möchte ich für eine Erweiterung der Methode plädieren. Diese wird ebenfalls in diesem Kapitel erläutert.

Castano, De Antonellis: Ermittlung von Schema Indexen, Datenbankclustern und ähnlichen Schemata – Einsatz dieser zur Wiederverwendung von Datenbanken. In zahlreichen Arbeiten von Castano und De Antonellis wird gezeigt, wie verschiedene Vorgehensweisen und Algorithmen zur Unterstützung einer Wiederverwendung von Datenbanken eingesetzt werden können.

Indexierung:
Für die automatische Ausführung vieler Methoden ist es erforderlich, die wichtigsten Entities in den Datenbanken zu kennen. In [CAZ92], [CaA94], [CAF95] und [CaA96] wird eine Methode vorgestellt, mit der diese automatisch aus konzeptuellen Schemata (Entity-Relationship

Diagrammen) abgeleitet werden. Dazu werden für jedes Entity folgende Merkmale ungewichtet ausgewertet:

- Summe der direkten Verbindungen des Entities
- Anzahl der Attribute
- Hierarchiestufen

Durch Addition dieser Eigenschaften wird eine Entity-Bedeutung für jedes Entity ermittelt. Es wird der Mittelwert über den so abgeschätzten Entity-Bedeutungen gebildet, jedes Entities, dessen Wert über dem Mittelwert liegt, wird als *Schema Descriptor (SD)* bezeichnet.

Ermittlung von Schemaähnlichkeit:
In [CAZ92] wird die Schemaähnlichkeit über den Vergleich der Schema Descriptoren (SD) ermittelt, dazu wird der Quotient $\frac{2 * \text{gemeinsame SD von S1 und S2}}{\text{SD von Schema S1 + SD von Schema S2}}$ ermittelt.
Durch diesen Vergleich läßt sich für eine Menge von Schemata eine Ähnlichkeitsmatrix aufstellen.

In [CAF95] und [CaA96] wird eine Methode zur Berechnung der Schemaähnlichkeit angegeben, die sich nicht auf Schema Descriptoren (SD) beschränkt, hier wird die Schemaähnlichkeit über den Vergleich aller Informationen (aller Enities) von konzeptuellen Schemata durchgeführt.

Schemaabstraktion:
In [CaA94], [CAF95] und [CaA96] wird angegeben, wie eine Schemaabstraktion auf der Basis der Schemaähnlichkeit verwendet werden kann. Dazu werden ähnliche Komponenten mehrerer Datenbanken ermittelt und als abstrakte Schemata gespeichert.

Aufbau von Bibliotheken:
Die vorgestellte Ermittlung von Schemaähnlichkeit wird zur Organisation von Bibliotheken verwendet. In [CAZ92] und [CaA93] wird eine Methode vorgestellt, mit der Bibliotheken mit Datenbanken im Entity-Relationhip Modell aufgebaut werden.

Durch Auswertung der Ähnlichkeitsmatrix, die aufgrund der Ermittlung von Schemaähnlichkeit entsteht, läßt sich ein binärer Baum aufbauen, in den alle Schemata bzgl. ihrer Ähnlichkeit einsortiert werden. Dieser Baum beinhaltet an den Blättern die repräsentierten Schemata, die jeweils ähnlichsten Knoten werden zu einem darüberliegenden Knoten zusammengefaßt, in diesem Knoten sind die gemeinsamen Merkmale der Schemata enthalten. Das Vorgehen wird solange fortgesetzt, bis auf diese Weise ein binärer Baum entstanden ist. Zusätzlich sind in diesen Clustern Metakonzepte enthalten, diese bestehen aus 1. einer textuellen Beschreibung, 2. vorgeschlagenen Verfeinerungen, 3. vorgeschlagenen Typen zur Wiederverwendung, 4. vorgeschlagenen Aktionen.

Börner, Pippig: Konzeptuelle Clusterung für das fallbasierte Schließen. In [BöP96] wird eine Methode zum fallbasierten Schließen mit folgenden Besonderheiten vorgestellt.
Die bekannten Fälle werden in einem binären Baum gespeichert, das Kriterium zur Einordnung in den Baum ist die strukturelle Ähnlichkeit der Fälle. Diese Aufgabenstellung ist ähnlich der Aufgabenstellung, die von [CaA93] behandelt wird, das Vorgehen beim Aufbau des Baumes ist analog.

Zur Lösung eines Falles wird in dem Baum top-down nach den ähnlichsten Fall gesucht. Dazu wird jeweils die ähnlichste Fallklasse ermittelt und deren Söhne werden untersucht. Sind zwei Fallklassen in gleichem Maße ähnlich, so müssen beide weiter verfolgt werden.

Werden zwei Fälle mit gleicher Ähnlichkeit gefunden, so wird der Fall ausgewählt, bei dem die häufiger angewendeten Konstrukte vorkommen. Man geht davon aus, daß diese die bewährten Lösungen sind. Gibt es auch hierbei zwei gleiche Fälle, so wird die kürzere Lösung gewählt.

Die Anwendung ähnlicher Zugänge in den Arbeiten von Castano/De Antonellis und Börner/Pippig bei der Organisation von Bibliotheken für die Wiederverwendung von Datenbanken und das fallbasierte Schließen weist auf die Ähnlichkeit der beiden Gebiete hin.

Bergmann, Eisenecker: Unterstützung der Wiederverwendung objektorientierter Software. In [BeE95] wird für objektorientierte Software untersucht, wie das Auffinden von geeigneten Programmteilen zur Wiederverwendung durch *fallbasiertes Schließen* unterstützt werden kann. Es wird festgestellt, daß jede Form der Wiederverwendung von Software eine Art fallbasierten Schließens mit den Phasen Retrieve, Reuse, Revise und Retain ist.
Dazu werden strukturelle und semantische Merkmale herangezogen. Darauf basierend wird eine optimale Zuordnung von Objekten zweier Mengen gesucht. Diese zeichnet sich durch eine *maximale ermittelte Gesamtähnlichkeit* aller zugeordneten Paare aus. Eine Untersuchung aller möglichen Zuordnungen aufgrund der Komplexität $O\frac{n_1!}{(n_1-n_2)!}$, wobei n_1 die Anzahl der Objekte der ersten Menge und n_2 die Anzahl der Objekte der zweiten Menge ist, ist nicht realisierbar. Es wird statt dessen ein *Hill-Climbing-Algorithmus* verwendet (Komplexität $O(n_1 * n_2)$), dieser sucht zu jedem Objekt der ersten Menge das lokal ähnlichste. Damit wird nur eine Annäherung an die Zielfunktion erreicht, nach Aussage der Autoren sind durch diesen theoretischen Fehler jedoch keine praktischen Auswirkungen zu erwarten.
Tests der Methode werden beschrieben. Dabei basiert die Suche nach Ähnlichkeiten auf syntaktischen und funktionalen Merkmalen. Ausgewertet werden die Anzahl der gefundenen Ähnlichkeiten und die Verwendbarkeit dieser. Es wurde festgestellt, daß die Auswertung von syntaktischen Merkmalen (Methodennamen, Anzahl und Klasse der Parameter und Rückgabewerte) wesentlich schlechtere Ergebnisse bringt als die Auswertung von syntaktischen und funktionalen Merkmalen (Ordnung der Elemente einer Klasse, Möglichkeiten des Zugriffs auf die Elemente, usw.).

Daraus kann die Schlußfolgerung gezogen werden, daß auch bei der Wiederverwendung von Datenbank-Schemata eine Beschränkung auf die Ermittlung struktureller Ähnlichkeiten nicht ausreicht. Deshalb wird in diesem Kapitel ein Ansatz vorgestellt, der alle verfügbaren Informationen zur Suche nach ähnlichen Datenbankteilen auswertet.

Abstandsfunktionen im fallbasierten Schließen. Beim fallbasierten Schließen werden Abstandsfunktionen eingesetzt, um Cluster in der Menge der vorhandenen Fälle zu bilden und um den ähnlichsten Fall zum fallbasierten Schließen zu ermitteln. Dazu sind verschiedene Abstandsfunktionen bekannt. Die einfachste Abstandfunktion ist die *Hamming distance*. Bei dieser wird die Anzahl der unterschiedlichen Merkmale ermittelt.
Eine Erweiterung dieser Abstandsfunktion stellt das *Tversky-Contrast Model* dar. In dieser werden übereinstimmende und unterscheidende Merkmale getrennt betrachtet. Bei einer Beschränkung auf Boolsche Attribute entsteht folgende Abstandfunktion:

$$T(x,y) = \alpha f(A) - \beta f(B) - \gamma f(C)$$

A - Menge der Attribute mit gleichen Werten für x und y,
B - Menge der Attribute mit Werte 1 für x und Wert 0 für y,
C - Menge der Attribute mit Werte 0 für x Wert 1 für y
α, β und γ sind positive reelle Zahlen

In den Attributwerten können unbekannte Werte auftreten, diese werden durch eine andere Abstandsfunktion, die von Richter in [Ric92a] vorgestellt wurde und im System PATDEX eingesetzt wird, behandelt.

$$sim_{PAT}(x,y) = \frac{\alpha|E|}{1|E| + 2|C| + \frac{1}{2}|U| + \frac{1}{2}|R|}$$

E - Menge der Attribute mit übereinstimmenden Werten für x und y,
C - Menge der Attribute mit unterschiedlichen Werten für x und y,
U - Menge der Attribute mit unbekannten Werten für x oder für y,
R - Menge der Attribute mit einem redundanten Wert für x

Weitere Modifikationen der vorgestellten Abstandsfunktionen wurden in [Pfe96] zusammengetragen.

6.4 Retrieve: Ermittlung gleicher oder ähnlicher Datenbankteile

Voraussetzung für die Wiederverwendung von Datenbank-Informationen ist die *Identifikation von gleichen oder ähnlichen Teilen* in Datenbanken. Dieses ist eine komplizierte Aufgabe, da Datenbankteile eine sehr komplexe Struktur darstellen und deshalb komplizierte Vergleichsoperationen über den Datenbanken erfolgen müssen.

Die *Aufgabenstellung* für die Identifikation von gleichen oder ähnlichen Teildatenbanken läßt sich formal beschreiben. Für zwei Datenbanken D_1 und D_2 werden Teildatenbanken $D_1' \subseteq D_1$ und $D_2' \subseteq D_2$ gesucht, für die gilt: $D_1' \approx D_2'$ (D_1' und D_2' sind einander ähnlich).

Einfacher als die Ermittlung von identischen oder ähnlichen Datenbankteilen ist die Identifikation identischer oder ähnlicher Attribute, Entities und Relationships der zu vergleichenden Datenbanken.

In Abschnitt 6.4.1 wird ein buttom-up-Zugang zum Vergleich von Datenbankteilen vorgestellt. Dieser beginnt mit dem Vergleich von Attributen, darauf basierend wird der Vergleich von Entities vorgestellt. Die Ermittlung ähnlicher Relationships basiert u.a. auf dem Vergleich der zugehörigen Entities und Attribute. Zur Ermittlung gleicher oder ähnlicher Datenbankteile werden die Vergleichsoperationen von Entities, Relationships und Attributen herangezogen.

Ein Entity oder Relationship eines Datenbank-Schemas kann Ähnlichkeit zu mehreren Entities oder Relationships eines anderen Datenbank-Schemas aufweisen. Die Suche nach ähnlichen Teildatenbanken ist deshalb keine triviale Aufgabe. Die Informationen über ähnliche Entities und Relationships von zwei Schemata werden dazu in einen Graphen aufgenommen (Abschnitt 6.4.2). Es wird eine optimale Zuordnung der ähnlichen Teile mit Hilfe von Algorithmen der Graphentheorie bestimmt (Abschnitt 6.4.3). Ein umfangreiches Beispiel in Abschnitt 6.4.4 erläutert das in den vorherigen Abschnitten beschriebene Vorgehen bei der Ermittlung ähnlicher Datenbankteile. Möglichkeiten, um bei der Ermittlung ähnlicher Teildatenbanken den Suchraum einzuschränken, werden in Abschnitt 6.4.5 vorgestellt.

6.4.1 Ähnlichkeitsmaß für Attribute, Entities und Relationships

Der Vergleich der einzelnen Datenbank-Komponenten ist voneinander abhängig. Die folgende Übersicht zeigt, welche Datenbankcharakteristika zur Bestimmung von ähnlichen Komponenten bei einem buttom-up-Zugang ausgewertet werden müssen:

Komponenten	ausgewertet werden:
Attribut	- Attributmerkmale
Entity	- Entitymerkmale,
	- zugehörige Attribute
Relationship	- Relationshipmerkmale,
	- zugehörige Attribute,
	- zugehörige Entities bzw. Relationships
Datenbankteile	- ähnliche Entities,
	- ähnliche Relationships
	- ähnliche Entities und Relationships

Es gibt Wechselwirkungen zwischen den einzelnen Datenbankkomponenten. In gewisser Weise können auch zugehörige ähnliche Relationships der Entities Hinweise auf ähnliche Entities geben. Damit man mit dieser Ermittlung ähnlicher Komponenten jedoch nicht in Zyklen läuft, erfolgt nur ein Vergleich in einer Richtung, also top-down oder buttom-up. Da ein *buttom-up Vergleich* den Vorteil hat, daß mit einfachen atomaren Komponenten begonnen wird und komplexere Ähnlichkeitsvergleiche darauf aufbauen, wird hier der buttom-up-Vergleich zur Ermittlung ähnlicher Teildatenbanken gewählt.

Für den Vergleich aller Komponenten wird eine Ähnlichkeitsfunktion *sim (= similarity)* $sim(A, B) \longrightarrow [0..1]$ verwendet. Diese gibt an, wie ähnlich die Komponenten A und B sind.

$$\begin{aligned} 1 &\quad \text{- völlige Übereinstimmung nach den Heuristikregeln} \\ 0 &\quad \text{- keine Übereinstimmung} \\ 0 < x < 1 &\quad \text{- Ähnlichkeit, aber keine vollständige Übereinstimmung} \end{aligned}$$

Die Bestimmung gleicher und ähnlicher Komponenten durch die Ähnlichkeitsfunktion wird im folgenden erläutert.

Die einzelnen Ähnlichkeitsmaße für den Vergleich von Attributen, Entities oder Relationships werden im folgenden angegeben.

I Ermittlung gleicher oder ähnlicher Attribute

Die Informationen über gleiche und ähnliche Attribute wird zur Bestimmung gleicher oder ähnlicher Datenbankteile nicht direkt ausgewertet. Sie gehen aber in die Ähnlichkeitsfunktion von Entities und Relationships ein. Folgende Merkmale liefern Hinweise auf ähnliche Attribute A und B:

Auswertung der Strukturangaben

Die Attributbezeichnungen und die Wertebereiche der Attribute können ausgewertet werden, um ähnliche Attribute zu bestimmen.

$H_{Attr}1$. Gleiche Attributnamen oder gleiche Teilstrings in den Attributnamen deuten auf ähnliche Attribute hin.

$H_{Attr}2$. Gleiche Attributtypen, gleiche oder ähnliche Längenangaben sind ein Hinweis auf ähnliche Attribute.

$H_{Attr}3$. Die Heuristikregel $H_{Attr}2$ kann durch Auswertung der Anzahl der verschiedenen Werte, die ein Attribut annehmen kann, ergänzt werden. Diese Heuristik liefert eine zusätzliche Angabe zum Wertebereich der Attribute.

Auswertung der Semantikinformationen

Geltende Integritätsbedingungen liefern ebenfalls Hinweise auf ähnliche Attribute.

$H_{Attr}4$. Sind die Attribute A und B in beiden Entities oder Relationships Schlüssel bzw. beide nicht im Schlüssel enthalten, so ist das ein zusätzlicher Hinweis auf ähnliche Attribute.

$H_{Attr}5$. Sind die Attribute A und B von den gleichen Attributen funktional abhängig, beziehungsweise die gleichen Attribute hängen von den Attributen A und B ab, so deutet das ebenfalls auf ähnliche Attribute hin.

Auswertung der Daten

Vorhandene Daten sind ein weiteres Merkmal, das zur Ermittlung ähnlicher Attribute herangezogen werden kann.

$H_{Attr}6$. Wenn bei Attributen gleiche Daten auftreten, so ist das ein weiterer Hinweis auf gleiche Attributbedeutung.

Diese Merkmale, die auf gleiche Attribute A und B deuten, müssen ausgewertet und dabei gegeneinander gewichtet werden. Dabei wird ein *Maß für die Ähnlichkeit (sim)* von Attributen ermittelt. Die ermittelte Ähnlichkeit soll größer sein, wenn mehr Heuristikregeln erfüllt sind. Abhängigkeiten zwischen den Heuristikregeln werden dabei nicht betrachtet, um folgende einfache Abschätzung der Ähnlichkeit verwenden zu können:

$$sim(A,B) := \sum_{i=1}^{6} w_i * H_{Attr}i(A,B)$$

$H_{Attr}i$ - Ergebnis der Heuristikregel $\in [0..1]$
$w_i \in [0..1]$ und $w_1 + .. + w_6 = 1$

Die Gewichtung der Heuristikregeln richtet sich nach der Zuverlässigkeit der Heuristiken und der Anwendbarkeit der Heuristiken. Die folgende Tabelle gibt diese an.

besonders aussagekräftig	$H_{Attr}1, H_{Attr}6$
aussagekräftig	$H_{Attr}2, H_{Attr}3$
nur im Zusammenhang mit anderen Heuristiken sinnvoll	$H_{Attr}4, H_{Attr}5$

Die Gewichtung der Heuristikregeln wird entsprechend dieser Übersicht festgelegt. Es ist möglich, daß *nicht alle Heuristikregeln auswertbar* sind (z.B. können in einer Datenbank noch keine Integritätsbedingungen bekannt sein). In diesem Fall werden die entsprechenden Heuristiken aus der Ähnlichkeitsbestimmung herausgenommen, die anderen Heuristikregeln werden dann höher gewichtet.

Eine *Erweiterung der Heuristikregeln* z.B. um Vergleichsoperationen, die *Verhaltensinformationen* und *Transaktionen* auswerten, ist möglich, sofern diese Informationen vorhanden sind. Auch hierbei wird nach identischen oder ähnlichen Verhaltensinformationen der Attribute und Transaktionen über den Attributen als zusätzliches Merkmal für die Ähnlichkeit von zwei Attributen gesucht.
Diese läßt sich nicht vollständig ermitteln, man kann aber eine einfachere Form anwenden, z.B. einen Vergleich, welche Attribute von den Transaktionen betroffen werden. Gibt es in zwei Datenbank-Schemata ähnliche Attributmengen, über denen jeweils die gleichen Transaktionen ausgeführt werden, so ist das ein weiterer Hinweis auf ähnliche oder identische Attribute.

Die *Bestimmung der Ähnlichkeit* von zwei *Attributmengen* kann nach folgender Ähnlichkeitsfunktion ermittelt werden:

$$sim(A_1..A_n, B_1..B_m) := max|\frac{2 * \sum_{i=1}^{n} sim(A_i, B_j)}{n+m}|$$

$j \in 1..m$, jedes j taucht nur einmal auf

Durch diese Formel werden fehlende beziehungsweise zusätzliche Attribute schwächer berücksichtigt als unterschiedliche Attribute.

II Gleiche oder ähnliche Entities

Bei der Ermittlung ähnlicher Entities in zwei Datenbanken D_1 und D_2 werden alle Entities der Datenbanken paarweise miteinander verglichen. Abbildung 6.3 zeigt zwei zu vergleichende Entities.

Abbildung 6.3: Vergleich von zwei Entities

Die Ähnlichkeitsabschätzungen für die Attribute der Entities $sim(A_{11}..A_{1n}, A_{21}..A_{2m})$ können für den Vergleich von Entities angewendet werden. Zusätzlich werden folgende weitere Merkmale herangezogen:

Auswertung von Strukturangaben

$H_{Ent}1$. Gleiche Entitynamen oder gleiche Teilstrings in den Entitynamen deuten auf gleiche oder ähnliche Entities hin.

Auswertung von Integritätsbedingungen

$H_{Ent}2$. Gleiche Schlüsselattribute von zwei Entities sind ein Hinweis auf ähnliche Entities. Dabei wird für die Schlüsselattribute A und B die Ähnlichkeit $sim(A, B)$ ermittelt.

Die Heuristik $H_{Ent}1$ sollte höher gewichtet werden als $H_{Ent}2$, da die Bezeichnungen der Entities plausiblere Hinweise geben als die Schlüsselattribute.

Die Ermittlung ähnlicher Entities erfolgt durch eine Mittelwertberechnung über die Charakteristika der Entities und ihrer zugehörigen Attribute.

$$sim(E_1, E_2) := \frac{1}{2} \sum_{i=1}^{2} w_i * H_{Ent}i(E_1, E_2) + \frac{1}{2} sim(A_{11}..A_{1n}, A_{21}..A_{2m})$$

$H_{Ent}i$ - Ergebnis der Heuristikregel $\in [0..1]$
$w_i \in [0, 1]$ und $w_1 + w_2 = 1$

Auf diese Weise werden Plausibilitäten für die Ähnlichkeit von zwei Entities ermittelt.

III Gleiche oder ähnliche Relationships

Gleiche oder ähnliche Relationships können nur in Zusammenhang mit ihrer *Umgebung in der Datenbank* ermittelt werden. Zur Bestimmung der *Ähnlichkeiten von zwei Relationships* $R_1 \in D_1$ und $R_2 \in D_2$ werden auch die *Entities*, über denen die Relationships definiert sind (E_{11}, E_{12} in D_1 und E_{21}, E_{22} in D_2) und die *Attribute der Relationships* ($B_{11}..B_{1n}$ - Attribute von R_1 und $B_{21}..B_{2m}$ - Attribute von R_2) herangezogen. Abbildung 6.4 zeigt zwei zu vergleichende Relationships aus den Datenbanken D_1 und D_2.

Es können neben dem Vergleich der zugehörigen Entities und Attribute weitere Heuristiken angegeben werden, die die Ähnlichkeit von zwei Relationships bestimmen:

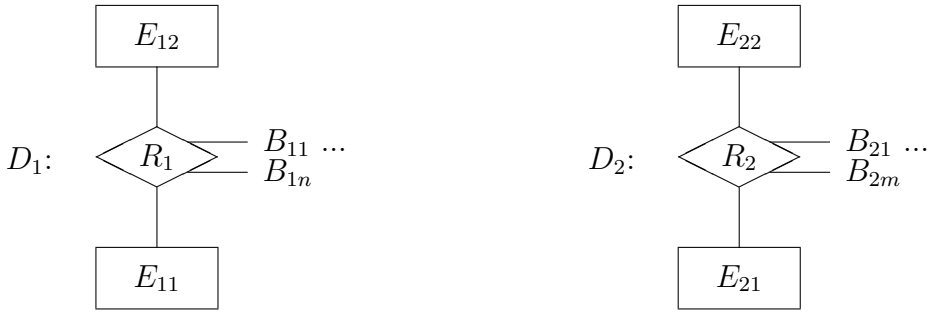


Abbildung 6.4: Vergleich von zwei Relationships

Auswertung von Strukturangaben

$H_{Rel}1$ Gleiche Relationshipnamen oder gleiche Teilstrings in den Relationshipnamen deuten auf gleiche oder ähnliche Relationships hin.

Auswertung von Integritätsbedingungen

$H_{Rel}2$ Gleiche Schlüsselattribute von zwei Relationships sind ebenfalls ein Hinweis auf gleiche Relationships.

$H_{Rel}3$ Bestehen in zwei Datenbanken gleiche Inklusions- und Exklusionsabhängigkeiten, so ist das ein ergänzender Hinweis auf gleiche Relationships.

Bei diesen Heuristiken sollte die Heuristik $H_{Rel}1$ besonders stark, die Heuristiken $H_{Rel}2$ und $H_{Rel}3$ schwächer gewichtet werden.

Neben diesen Heuristiken, die nur die Informationen des Relationships auswerten, kann auch eine Heuristikregeln aufgestellt werden, die die Kardinalität zwischen einem Relationship und einem zugehörigen Entity auswertet.

$H_{Rel}4$ Gleiche Kardinalitäten $\mathsf{card}(R_1, E_{11})$ und $\mathsf{card}(R_2, E_{21})$ an den Pfaden der Relationships R_1, R_2 zu einem zugehörigen Entity E_{11} bzw. E_{21} sind ein ergänzender Hinweis auf gleiche oder ähnliche Relationships. Dieses Merkmal wird zusammen mit der Ähnlichkeit der zugehörigen Entities der Relationships ausgewertet.

Zur Bestimmung ähnlicher Relationships wird der Mittelwert über den auswertbaren Eigenschaften (Relationshipeigenschaften, Ähnlichkeit der Attribute des Relationships, Ähnlichkeit der zugehörigen Entities, einschließlich Kardinalitäten) berechnet:

$$sim(R_1, R_2) := \tfrac{1}{4} \sum_{i=1}^{3} w_i * H_{Rel}i(R_1, R_2) + \tfrac{1}{4} sim(B_{11}..B_{1n}, B_{21}..B_{2m}) +$$
$$\tfrac{1}{8} sim(E_{11}, E_{12}) + \tfrac{1}{8} H_{Rel}4(R_1, E_{11}, R_2, E_{21}) +$$
$$\tfrac{1}{8} sim(E_{21}, E_{22}) + \tfrac{1}{8} H_{Rel}4(R_1, E_{12}, R_2, E_{22})$$

$H_{Rel}i$ - Ergebnis der Heuristikregel $\in [0, 1]$
$w_i \in [0, 1]$ 0..1 und $w_1 + w_2 + w_3 = 1$

Bei binären Relationships gibt es zwei Möglichkeiten, die zugehörigen Entities zu vergleichen (E_{11} — E_{21} und E_{12} — E_{22}, beziehungsweise E_{11} — E_{22} und E_{12} — E_{12}). Es wird die Zuordnung mit der größeren Ähnlichkeit gewählt.

Bei *Higher-Order-Entity-Relationship Modellen* können Relationships über anderen Relationships definiert sein. In diesem Fall muß beim Vergleich anstelle der Ähnlichkeit der zugehörigen Entities die Ähnlichkeit von Relationships einer niedrigeren Ordnung eingesetzt werden.

Man kann mit dieser Methode auch *ternäre und höhere Relationships* vergleichen, in diesem Fall müssen mehrere zugehörige Entities bzw. Relationships verglichen werden und in die Abschätzung der Ähnlichkeit der Relationships aufgenommen werden. Die Gewichte in der Ähnlichkeitsfunktion müssen entsprechend geändert werden.

IV Ähnlichkeit zwischen Entity und Relationship

In Datenbanken können ähnliche Einheiten auftreten, die strukturell unterschiedlich dargestellt wurden. Um diese zu finden, muß es möglich sein, auch Entities und Relationships untereinander zu vergleichen. Abbildung 6.5 zeigt zu vergleichende Entities und Relationships.

Abbildung 6.5: Vergleich von Entities und Relationships

Ein Beispiel aus der Literatur, das zeigt, daß so ein Vergleich sinnvoll ist, sind Datenbanken in [BCN92] und [Tha98]. In diesen sind Beispieldatenbanken mit Universitätsinformationen enthalten, bei Batini, Ceri, Navathe werden dabei Studentendaten als Entity dargestellt, bei Thalheim als einstelliges Relationship über einem Entity Personen. Trotz dieser unterschiedlichen strukturellen Darstellung haben das Entity bzw. Relationship mit den Studenteninformationen die gleiche Bedeutung in den jeweiligen Datenbanken.

Beim Vergleich von Entities und Relationships können ebenfalls die Attribute der Konstrukte ausgewertet werden. Weiterhin können folgende Merkmale herangezogen werden:

Auswertung von Strukturangaben

$H_{ER}1$ Gleiche Entity- und Relationshipnamen und gleiche Teilstrings in den Entity- und Relationshipnamen weisen auf eine gleiche Bedeutung der Konstrukte hin.

Auswertung von Integritätsbedingungen

$H_{ER}2$ Gleiche Schlüsselattribute des Entities bzw. Relationships liefern Hinweise auf eine gleiche Bedeutung.

Dazu wird folgende Formel verwendet:

$$sim(E,R) := \frac{1}{4} sim(A_1..A_n, B_1..B_m) + \frac{1}{4} \sum_{i=1}^{2} w_i * H_{ER}i(E,R)$$

$H_{ER}i$ - Ergebnis der Heuristikregel $\in [0..1]$
$w_i \in [0,1]$ und $w_1 + w_2 = 1$

Diese Abschätzung wurde so gewählt, daß die Plausibilitätsabschätzung maximal 0.50 betragen kann, da die strukturellen Eigenschaften gegen eine Ähnlichkeit sprechen.

V Ähnlichkeit zwischen Attribut und Entity oder Relationship

In den bisher beschriebenen Fällen wurde gezeigt, wie Ähnlichkeiten in Datenbank-Schemata, die die gleiche Granularität haben, gefunden werden können. Es kann auch sein, daß Datenbank-Schemata den *gleichen Inhalt* aber eine *unterschiedliche Granularität* haben, der Detailliertheitsgrad der Datenbank-Schemata also unterschiedlich ist.

Es ist in diesem Fall möglich, daß in zwei Datenbank-Schemata identische Informationen auftauchen, die in dem einen Datenbank-Schema als Attribut, in dem anderen als Entity oder Relationship dargestellt werden. Diese Ähnlichkeiten sind schwieriger zu finden als die bisher aufgezählten Fälle. Man kann folgende Heuristikregeln auswerten, um solche Ähnlichkeiten zu finden:

Auswertung der Strukturangaben

$H_{AE}1$ Die Bezeichnung des Attributes wird mit der Bezeichnung des Entities oder Relationships verglichen. Werden gleiche Namen oder gleiche Teilstrings in den Namen gefunden, so deutet das auf Ähnlichkeiten hin.

$H_{AE}2$ Besitzt das zu vergleichende Attribut den gleichen Wertebereich wie ein Attribut des Entities oder Relationships, so können hier ebenfalls Ähnlichkeiten bestehen.

Auswertung der Daten

$H_{AE}3$ Wenn bei dem Attribut und einem Attribut des Entities oder Relationships gleiche Daten auftreten, so ist das ein weiterer Hinweis auf gleiche Bedeutung.

Die angegebenen Heuristikregeln werden in folgender Abschätzung ausgewertet:

$$sim(A, E) := \sum_{i=1}^{3} w_i * H_{AE}i(A, E)$$

$H_{AE}i$ - Ergebnis der Heuristikregel $\in [0..1]$
$w_i \in [0, 1]$ und $w_1 + .. + w_3 = 0.5$

Die folgende Übersicht zeigt, wie zuverlässig die Hinweise der einzelnen Heuristikregeln für die meisten Fälle sind:

aussagekräftig	$H_{AE}1, H_{AE}3$
nur im Zusammenhang mit anderen Heuristiken sinnvoll	$H_{AE}2$

Die Gewichtung der Heuristikregeln erfolgt entsprechend dieser Übersicht.

VI Vergleich von Datenbanken mit unterschiedlicher Granularität

Es wurde schon im vorherigen Abschnitt erwähnt, daß der Vergleich von Datenbank-Schemata mit unterschiedlicher Granularität schwierig, jedoch für eine Wiederverwendung von Informationen in vielen Fällen notwendig ist. Es wurde bisher ein Vergleich zwischen Attributen und Entities bzw. Relationships vorgestellt.

Es ist auch möglich, daß identische Informationen in kleineren Teildatenbanken zu finden sind, z.B. können die Informationen, die in einem Datenbank-Schema in zwei Entities, die durch ein Relationship verbunden sind, auftreten, sich in einem anderen Datenbank-Schema in einem Entity befinden. Um solche Ähnlichkeiten zu finden, ist ein komplexerer Vergleich von Teildatenbanken erforderlich. Man kann dazu verschiedene kleine Teildatenbanken angeben, die untereinander verglichen werden. Verglichen werden dabei Bezeichnungen in der Datenbank, zugehörige Attribute, Wertebereiche, konkrete Datenwerte und Integritätsbedingungen.

Dieser Vergleich ist viel aufwendiger als der Vergleich von Entities oder Relationships. Will man dieses Verfahren einsetzen, so muß deshalb die Suche sehr gezielt erfolgen, der Suchraum muß stark eingeschränkt werden. Eine Möglichkeit dazu ist die Ermittlung ähnlicher Kernrelationen (Abschnitt 7.7) zweier Datenbank-Schemata und die Clusterbildung (Abschnitt 7.5) um dieser Kernrelationen. Zwischen diesen Clustern können dann gezielt komplexere Ähnlichkeitsvergleiche durchgeführt werden.

Bisher wurde dargestellt, wie atomare Komponenten zwischen zwei Datenbanken verglichen werden können, wobei der Vergleich von Attributen, Entities und Relationships gut realisierbar ist, dafür lassen sich Ähnlichkeitsmaße ermitteln. Im nächsten Abschnitt wird dargestellt, wie aufgrund der angegebenen Ähnlichkeitsmaße für die Datenbank-Komponenten der Vergleich von Teildatenbanken erfolgen kann.
Es werden dazu der Vergleich von Entities, Relationships und Entities und Relationships untereinander herangezogen. Das Ähnlichkeitsmaß zum Vergleich von Attributen mit Entities oder Relationships und von komplexeren Datenbankteilen wird im folgenden nicht in die Erläuterung einbezogen, die so ermittelten Ähnlichkeiten können aber in analoger Form ausgewertet werden.

6.4.2 Erstellung eines Graphen zur Lösung des Zuordnungsproblems

Im vorigen Abschnitt wurden *Heuristiken* angegeben, um *Attribute, Entities* und *Relationships* miteinander zu vergleichen. Es kann dabei sein, daß ein Entity oder Relationship der einen Datenbank Ähnlichkeiten zu mehreren Entities bzw. Relationships der anderen Datenbank aufweist.

Diese Möglichkeit wurde in [SDG95] nicht beachtet. In [BeE95] wird auf die Komplexität der Suche nach einer optimalen Zuordnung hingewiesen. Es wird dort ein Hill-Climbing-Algorithmus eingesetzt, der für jedes Objekt ein maximal ähnliches ermittelt. Dadurch findet man lokale Maxima, nicht notwendig ein globales Maximum.

Die Ermittlung eines Ähnlichkeitsmaßes für die Datenbank-Komponenten basiert auf Abschätzungen und intuitiven Wichtungen. Trotzdem soll für die Ermittlung von ähnlichen Datenbankteilen eine exakte Berechnung, die auf diesen Abschätzungen basiert, eingesetzt werden, da diese effizient zu ermitteln ist. Es wird vorgestellt, wie eine *optimale Zuordnung ähnlicher Datenbankkomponenten* durch einen *Graphmatchingalgorithmus* gefunden werden kann. Dazu wird ein bipartiter Graph aufgebaut, in dem alle ähnlichen Datenbankkomponenten mit den ermittelten Ähnlichkeitsbewertungen eingetragen werden. Dieser wird als *Ähnlichkeitsgraph* bezeichnet.

Definition 6.1 *Ein* **bipartiter Graph** G *besteht aus einer nichtleeren Menge von Knoten* $V(G)$*, die sich in zwei disjunkte Teilmengen* $S \dot{\cup} T$ *unterteilen läßt und einer Menge von Kanten* $E(G) \subseteq \{(s,t)|s \in S, t \in T\}$*, sodaß alle Kanten in* $E(G)$ *einen Knoten aus* S *und einen Knoten aus* T *verbinden.*

Für alle Kanten $e_i \in E(G)$ werden mit $a(e_i)$ der *Anfangspunkt* der Kante in S ($a(e_i) \in S$) und mit $e(e_i)$ der *Endpunkt* der Kante in T ($e(e_i) \in T$) bezeichnet.

Die Kanten des Graphen sind *gewichtet*. Es gibt eine Funktion w, die die Kantengewichtungen festlegt ($w : E(G) \longrightarrow [0,1]$).

Algorithmus zum Aufbau des Ähnlichkeitsgraphen

In den bipartiten Graphen werden alle Entities und Relationships der Datenbanken D_1 und D_2 aufgenommen, die nach der Abschätzung durch die Ähnlichkeitsfunktion einander ähnlich sind. Im folgenden wird die Konstruktion des Graphen erläutert.

1. Begonnen wird mit einem leeren Graphen.
2. Alle ermittelten ähnlichen Entities/ Relationships $K_1 \in D_1$ und $K_2 \in D_2$ werden in den bipartiten Graphen als *Knoten* aufgenommen, wobei K_1 in S und K_2 in T aufgenommen wird. Zwischen diesen Knoten wird eine *gewichtete Kante* in den Graphen eingetragen, die Gewichtung w der Kante ist der über das Ähnlichkeitsmaß *sim* bestimmte Wert.

Im Abschnitt 6.4.4 wird ein umfangreiches Beispiel für den Aufbau eines Ähnlichkeitsgraphen gezeigt.

Einführung eines Schwellwertes

Es ist sinnvoll, nur solche Knoten und Kanten aufzunehmen, deren Ähnlichkeitsmaß über einem bestimmten *Schwellwert* ϵ liegt. Das Beispiel in Abschnitt 6.4.4 zeigt, daß durch das Ähnlichkeitsmaß viele Zuordnungen mit geringer Plausibilität ermittelt werden, die nicht sinnvoll erscheinen. Diese möchte man weitgehend ausschließen. Außerdem wird die Kantenanzahl im Graphen auf diese Weise beschränkt. Das vereinfacht die Suche nach einer optimalen Zuordnung.
In [SDG95] wird ein Schwellwert ϵ mit 0.33 angenommen. Dort werden Konstrukte, deren Ähnlichkeit mit einer geringeren Plausibilität abgeschätzt wird, nicht berücksichtigt.
In dieser Arbeit wird der Schwellwert etwas niedriger angesetzt, da beim Vergleich von Relationships vier gleichgewichtete Merkmale ausgewertet werden. Bereits bei einem vollständig zutreffenden Merkmal sollen diese Relationships berücksichtigt werden. Deshalb wird für die hier vorgestellte Ähnlichkeitsfunktion ein Schwellwert von 0.25 empfohlen.
Es ist auch möglich, einen *variablen Schwellwert* einzusetzen, dieser kann, wenn keine Zuordnungen ähnlicher Komponenten gefunden werden, schrittweise herabgesetzt werden. Der *optimale Wert für den Schwellwert* ist abhängig von der Anwendung und dem Einsatz der ermittelten ähnlichen Datenbankteile. Möchte man auf jeden Fall alle Zuordnungen von ähnlichen Teildatenbanken finden, auch wenn dabei teilweise falsche Zuordnungen gefunden werden, dann muß der *Schwellwert niedrig* sein. Sollen in erster Linie sinnvolle Zuordnungen ermittelt werden, auch wenn dabei teilweise plausible Zuordnungen ähnlicher Teildatenbanken nicht gefunden werden, so muß der *Schwellwert hoch* sein.

6.4.3 Matchingalgorithmus über dem Graphen

Die Informationen über gleiche oder ähnliche Entities und Relationships sind in einem *gewichteten bipartiten Graphen* eingetragen. Man sucht eine *eineindeutige Zuordnung* dieser ähnlichen atomaren Komponenten zueinander. Damit kann man diese Datenbankanwendung auf ein Grundproblem der Graphentheorie zurückführen.

Kennt man Hintergrundwissen der Anwendungsgebiete und die Bedeutung von zwei Datenbanken, so kann man ersehen, welche Teile der Datenbanken identisch oder ähnlich sind. Dabei gibt es jeweils nur wenige sinnvolle Zuordnungen (in den meisten Fällen nur eine) zwischen den Entities und Relationships. Will man den Prozeß der Suche nach ähnlichen Datenbankteilen automatisieren und das Zuordnungsproblem automatisch lösen, so muß man versuchen, die Zielfunktion so festzulegen, daß man sich diesen richtigen Zuordnungen möglichst weit nähert.

Auf dem Graphen, in dem ähnliche Entities und Relationships eingetragen werden, wird ein *Matchingalgorithmus* mit dem Ziel ausgeführt, möglichst große ähnliche Datenbankteile mit möglichst hoher Ähnlichkeit zu finden.

Definition 6.2 *Eine Kantenmenge in einem Graphen wird* **Matching** M *genannt, wenn es keine Kanten gibt, die einen gemeinsamen Knoten haben.*
$\nexists\, m_1, m_2 \in M, m_1 \neq m_2$ *mit* $a(m_1) = a(m_2)$ *oder* $e(m_1) = e(m_2)$.

Es soll also eine eineindeutige Zuordnung der Entities/Relationships der Datenbank zueinander gefunden werden, dabei sollen möglichst viele Komponenten miteinander verbunden werden, die Ähnlichkeit der einzelnen Komponenten soll möglichst groß sein.

Daraus ergibt sich die *Zielfunktion* für das Matching. Es wird ein Matching M gesucht, mit

$$w(M) \sum_{m \in M} w(m) \text{ ist maximal}$$

$w(M)$ wird als das *Gewicht des Matchings* bezeichnet. Das Matching, das die Zielfunktion erfüllt, wird als *maximales Matching* bezeichnet.

Definition 6.3 *Ein Matching wird als* **maximales Matching** M *bezeichnet, wenn es kein Matching M' gibt, für das gilt: $w(M') > w(M)$.*

Definition 6.4 *Ein Matching M über einem Graphen G wird als* **perfektes Matching** *bezeichnet, wenn alle Knoten $V(G)$ durch eine Kante von M berührt werden.*

Da der Algorithmus zur Bestimmung eines maximalen Matching nicht trivial ist, wird zunächst eine einfache Methode gezeigt, mit der aber nicht in jedem Fall ein maximales Matching gefunden werden kann. Anschließend wird ein exakter Algorithmus zur Bestimmung des maximalen Matching gezeigt (Ungarische Methode).

Finden eines guten Matching

Eine sehr einfache Methode zur Bestimmung eines Matchings, die aber nicht in jedem Fall ein maximales Matching ermittelt, ist folgende:

Man beginnt mit einem leeren Matching M.
Für alle Kanten $E(G)$ des Graphen G wird *nach ihrer Wichtung sortiert* untersucht, ob diese Kante im Matching ergänzt werden können.

INPUT: Graph G mit Knotenmenge V und Kantenmenge E

OUTPUT: Matching M des Graphen G

```
M := ∅;
E' := E(G);
REPEAT
    SELECT k ∈ E' mit höchster Gewichtung;
    E' := E' − k;
    (* Es wird untersucht, ob es keine Kante in M gibt, die den gleichen
       Anfangs- oder Endpunkt wie k hat, nur dann kann die Kante k
       ins Matching aufgenommen werden. *)
    IF (∄ m ∈ M mit a(k) = a(m) oder e(k) = e(m)) THEN
        M := M ∪ k;
    END;
UNTIL E' = ∅;
```

Algorithmus 6.1: Ermittlung eines guten Matching

Die Kanten in M beinhalten nach Ausführung des Algorithmus das ermittelte Matching.

Dieser Algorithmus bringt bessere Ergebnisse als der Hill-Climbing-Algorithmus, der in [BeE95] angewendet wurde. Da die Kanten nach ihrer Wichtung sortiert ergänzt werden, ist sichergestellt, daß zu jedem Knoten die Kante mit der größten Gewichtung ausgewählt ist. Das Verfahren kann deshalb kein schlechteres Ergebnis als der Hill-Climbing-Algorithmus ermitteln. Im Gegensatz zum Hill-Climbing-Algorithmus ist das hier vorgestellte Verfahren unabhängig von der Reihenfolge der Knoten im Graphen.

Der Algorithmus zur Festlegung eines guten Matching hat eine Komplexität in der Ordnung des Suchproblems nach der Kante mit der jeweils höchsten Gewichtung $O(|E|^2)$.

Ein Beispiel für die Anwendung dieses Algorithmus zur Suche nach ähnlichen Datenbankteilen folgt in Abschnitt 6.4.4.

Ungarische Methode

Die Ungarische Methode ist einer der bekanntesten und wichtigsten kombinatorischen Algorithmen. Er geht auf König (1916) und Egarváry (1931) zurück. Die Ungarische Methode ist ein Algorithmus zur Ermittlung eines maximalen Matchings für bipartite Graphen.

Für ungewichtete bipartite Graphen ist der Ungarische Algorithmus in [LoP86] angegeben, für gewichtete bipartite Graphen ist der Algorithmus komplizierter, dieser ist in [Jun94] angegeben und bewiesen.

In diesem Abschnitt soll der Ungarische Algorithmus für die Datenbankanwendung (gewichtete bipartite Graphen) angegeben und anhand von Beispielen erläutert werden.

Grundidee der Ungarischen Methode. Die Definition des maximalen Matching zeigt, daß man ein maximales Matching finden kann, wenn man alle existierenden Matchings heranzieht und das Matching mit dem größten Gewicht auswählt.
Um die Betrachtung aller Matchings zu vermeiden, wird im Ungarischen Algorithmus eine *obere Schranke der Gewichte* konstruiert, die größer oder gleich dem Gewicht aller existierenden Matchings ist.
Mit dem Ungarischen Algorithmus sucht man den Fall der Gleichheit. Man sucht ein Matching M', dessen Gewicht mit der oberen Schranke übereinstimmt. Findet man so ein Matching, so weiß man, daß es kein Matching gibt, das eine höheres Gewicht haben kann. Man hat also ein maximales Matching gefunden.

Im folgenden wird das Vorgehen beim Ungarischen Algorithmus ausführlich erläutert, der Algorithmus dazu angegeben und anhand eines Beispiels demonstriert. Abbildung 6.6 zeigt ein abstraktes Beispiel, das zur Erläuterung herangezogen werden soll.

Der Beispielgraph enthält fünf Knoten, die Kanten des Graphen und die Bewertungen der Kanten sind aus Abbildung 6.6 zu entnehmen.

Erstellung eines vollständigen bipartiten Graphen. Voraussetzung, um die ungarische Methode für gewichtete Graphen anwenden zu können, ist das Vorhandensein von *vollständigen bipartiten Graphen*. Dabei muß die Knotenanzahl in S und T gleichgroß sein, ist das in dem Graphen nicht der Fall, so werden *Knoten* in S bzw. T *ergänzt*. Zwischen jedem Knoten aus S und jedem Knoten aus T muß eine Kante existieren. Fehlen in dem Graphen Kanten, so werden

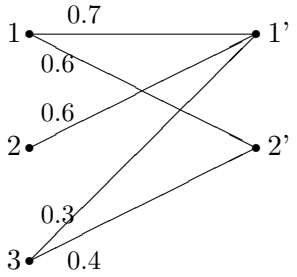


Abbildung 6.6: Beispielgraph

an den betreffenden Stellen *Kanten mit der Gewichtung 0 ergänzt.*
Für den Beispielgraphen aus Abbildung 6.6 wird der durch diese Ergänzungen entstehende vollständige bipartite Graph in Abbildung 6.7 gezeigt.

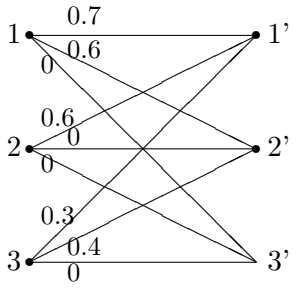


Abbildung 6.7: Vollständiger bipartiter Graph zum Beispielgraph

Aufgabenstellung des Ungarischen Algorithmus. Gesucht ist ein maximales Matching M für einen Graphen G. Zur Suche nach diesem Matching M wird als Abbruchkriterium eine obere Schranke für das Gewicht jedes Matchings benötigt. Es werden deshalb zwei Vektoren $u = (u_1, .., u_n)$ und $v = (v_1, .., v_n)$ eingeführt, die folgende Eigenschaft haben:

$$u_i + v_j \geq w_{ij} \tag{6.1}$$

Erfüllen die Vektoren diese Bedingung, so heißen sie *zulässig.*

Aufgrund der Eigenschaft zulässiger Vektoren (6.1) gilt:

$$w(M') \leq w(M) \leq \sum_{i=1}^{n} (u_i + v_i) \tag{6.2}$$

wobei M' - ein beliebiges Matching und
M - ein maximales Matching ist

Man sucht also mit dem Ungarischen Algorithmus eine *Belegung für die Vektoren u und v*, sowie ein *Matching* M', sodaß in 6.2 die Gleichheitsbedingung erfüllt ist, in diesem Fall weiß man, daß das Matching M' ein maximales Matching ist. Der ungarische Algorithmus konstruiert eine solche Lösung, er wird im folgenden angegeben.

(a) Die Vektoren u und v werden mit zulässigen Initialwerten belegt. Dazu können z.B. die Werte v_j auf 0 gesetzt und die Werte $u_i = max(w_{ij})$ über alle j gesetzt werden.

(b) In dem Graphen G werden alle *Kanten (ij) markiert*, für die folgende Eigenschaft erfüllt ist:

$$u_i + v_j = w_{ij} \tag{6.3}$$

Es wird ein *perfektes Matching* (perfektes Matching: alle Knoten des Graphen sind im Matching enthalten) im Graphen G gesucht, das nur markierte Kanten enthält. Findet man ein perfektes Matching, so ist dieses eine *Lösung* des Zuordnungsproblemes.

Begründung dieses Schrittes: Da man nach einem Matching sucht, für das die Gleichheit in der Formel 6.2 erfüllt ist, dürfen in diesem Matching nur Kanten vorkommen, für die $u_i + v_j = w_{ij}$ gilt. Bereits eine Kante (ij) im Matching, für die $u_i + v_j > w_{ij}$ gilt, bewirkt, daß die Gleichheit verletzt ist und damit das Matching kein maximales Matching ist.

Findet man kein perfektes Matching über den markierten Kanten, so müssen die Vektoren u und v verändert werden.

Der Algorithmus wird mit (c) fortgesetzt.

(c) Gibt es kein perfektes Matching im Graphen G, das nur die markierten Kanten einbezieht, so muß der Fall vorliegen, daß es eine Teilmenge $J \subseteq S$ gibt, für die gilt:

$$|(N(J))| < |J| \tag{6.4}$$

$N(J)$ sind *Nachbarn* von J, das sind solche Knoten in T, die durch Knoten in J über eine *markierte Kante* verbunden sind.

Es ist leicht nachvollziehbar, daß bei dem in 6.4 beschriebenen Fall nicht jedem Knoten aus J ein Knoten aus T eineindeutig zuordnet werden kann, deshalb ist kein perfektes Matching möglich.

Man beachte: Die Eigenschaft ist auch erfüllt, wenn in J nur ein Knoten enthalten ist, und $N(J)$ leer ist.

Da kein perfektes Matching möglich ist, das die Gleichheit in der Formel 6.2 erfüllt, muß die obere Schranke der Gewichte verändert werden. Die Vektoren u und v werden dazu wie folgt verändert:

$$u_i := \begin{cases} u_i - \delta, & i \in J \\ u_i, & i \notin J \end{cases}$$

$$v_j := \begin{cases} v_j + \delta, & j \in N(J) \\ v_j, & j \notin N(J) \end{cases}$$

$$\delta := \min \{u_i + v_j - w_{ij}\} : i \in J, j \notin N(J)$$

Der Beweis, daß die so veränderten Vektoren zulässig sind, wurde in [Jun94] geführt.

Für diese veränderten Vektoren wird erneut ein perfektes Matching gesucht, dazu wird der Algorithmus mit (b) fortgesetzt.

Der Beweis, daß der angegebene Algorithmus ein maximales Matching mit der Komplexität $O(|V(G)|^3)$ bestimmt, wurde in [Jun94] geführt.

Dieser Algorithmus soll für den Beispielgraphen aus Abbildung 6.6 erläutert werden:

(a) Die initialen Vektoren werden wie folgt belegt:

u=(0.7, 0.6, 0.4), v=(0, 0, 0)

(b) In dem Graphen können Kanten markiert werden, weil sie die Bedingung 6.3 erfüllen. Abbildung 6.8 zeigt die markierten Kanten des Graphen.

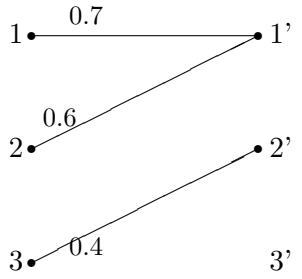


Abbildung 6.8: Markierte Kanten des Graphen (1)

Auf diesem Graphen ist kein perfektes Matching möglich, deshalb werden die Vektoren u und v verändert.

(c) Es muß nach einer Knotenmenge $J \subseteq S$ gesucht werden, in der mehr Knoten enthalten sind als Nachbarn zu dieser Menge existieren.

gefunden wird : $J = \{1,2\}$
$N(J) = \{1'\}$

Entsprechend erfolgt das Verändern der Vektoren u und v, sodaß diese folgende Werte annehmen:

δ = min ($u_1 + v_2 - w_{12}$, $u_1 + v_3 - w_{13}$, $u_2 + v_2 - w_{22}$, $u_2 + v_3 - w_{23}$)
= min (0.1, 0.7, 0.6, 0.6)= 0.1

Daraus ergeben sich folgende Werte für u und v:

u= (0.6, 0.5, 0.4)
v= (0.1, 0, 0)

Mit diesen Vektoren wird der Algorithmus fortgesetzt:

(b) Abbildung 6.9 zeigt, welche Kanten markiert werden können, weil mit den geänderten Vektoren die Bedingung 6.3 erfüllt ist.

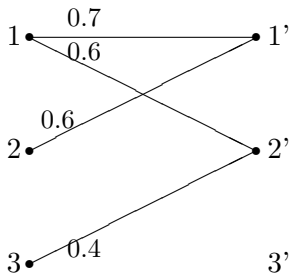


Abbildung 6.9: Markierte Kanten des Graphen (2)

Auch in diesem Graphen kann kein perfektes Matching gefunden werden, deshalb wird der Algorithmus fortgesetzt.

(c) Es wird wieder nach einer Knotenmenge $J \subseteq S$ gesucht, in der mehr Knoten enthalten sind als Nachbarn zu dieser Menge existieren.

gefunden wird dabei: J = { 1, 2, 3}
$N(J)$ = { 1', 2'}

Entsprechend werden die Vektoren u und v verändert:

δ = min ($u_1 + v_3 - w_{13}$, $u_2 + v_3 - w_{23}$, $u_3 + v_3 - w_{33}$)
= min (0.6, 0.5, 0.4)= 0.4

Daraus ergeben sich folgende Werte für u und v:

u= (0.2, 0.1, 0)
v= (0.5, 0.4, 0)

Mit diesen Vektoren wird der Algorithmus fortgesetzt:

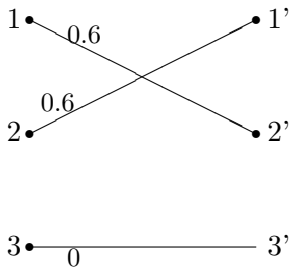


Abbildung 6.10: Markierte Kanten des Graphen (3)

(b) In Abbildung 6.10 werden die Kanten gezeigt, die aufgrund der Vektoren u und v markiert werden können:

Auf diesem Graphen ist ein perfektes Matching möglich, dieses beinhaltet alle in Abbildung 6.10 markierten Kanten. Das Gewicht des Matchings ist 1.2.

Soweit zur Erläuterung der Ungarischen Methode an einem abstrakten Beispiel. Das durch die Ungarische Methode ermittelte maximale Matching zeigt eine eineindeutige Zuordnung von ähnlichen Datenbankteilen an. Die Anfangspunkte der Kanten des Matching zeigen auf die Knoten der Teildatenbank D'_1, die Endpunkte auf die Knoten der Teildatenbank D'_2. Diese Teildatenbanken D_1 und D_2 sind einander nach den Heuristikregeln ähnlich.

Das Gewicht des maximalen Matching $w(M)$ wird als *Gesamtähnlichkeitsmaß* für die Ähnlichkeit der Datenbankteile bezeichnet. Dieses wird benötigt, wenn für eine Datenbank zu mehreren anderen Datenbanken Ähnlichkeiten gefunden werden. In diesem Fall muß man auswählen, welche Datenbank herangezogen wird, um Informationen wiederzuverwenden. Das Gesamtähnlichkeitsmaß ist dafür ein Kriterium. In diesem spiegelt sich sowohl der *Grad der Ähnlichkeit* als auch die *Größe der ähnlichen Teile* wider.

Der Ungarische Algorithmus wird in Abschnitt 6.4.4 für ein Datenbankbeispiel gezeigt.

Zerlegung eines Graphen in unabhängige Komponenten

Der Matchingalgorithmus kann auf Teilgraphen des Ähnlichkeitsgraphen, zwischen denen keine Kante existiert, getrennt ausgeführt werden. Zur Zerlegung des Graphen kann folgender Algorithmus verwendet werden:

Aus der Menge der Knoten $V(G)$ des Graphen G, wird ein beliebiger Knoten ausgewählt und in eine Komponente übernommen. Für diesen Knoten werden alle Kanten ermittelt, die ihn berühren. Alle Knoten, die ebenfalls von diesen Kanten berührt werden, werden ebenfalls in die Komponente aufgenommen (falls sie noch nicht in der Komponente vorhanden sind). Dieses Vorgehen wird fortgesetzt, bis die Komponente auf diese Weise nicht mehr erweitert werden kann. Die Knoten der Komponente werden aus dem Graphen gelöscht, mit dem Restgraphen wird die Methode wiederholt.

Bei dem beschriebenen Vorgehen handelt es sich um eine Tiefe-zuerst-Suche.

INPUT: Graph G mit Knotenmenge V und Kantenmenge E

OUTPUT: disjunkte Zerlegung des Graphen in $G_1 = (V_1, E_1), G_2 = (V_2, E_2), .., G_n = (V_n, E_n)$

```
i := 1;
WHILE (V ≠ ∅) DO
BEGIN
    SELECT v ∈ V;
    V_i := v;
    E_i := ∅;
    bilde_komponente(v, V_i, E_i, E);
    V := V − V_i;
    E := E − E_i;
    i := i + 1;
END;

PROCEDURE bilde_komponente(v, VAR V_i, VAR E_i, E);
BEGIN
    FOR ALL k ∈ E mit ((a(k) = v) ∨ (e(k) = v') oder (a(k) = v') ∨ (e(k) = v))
                  und k ∉ E_i DO
    BEGIN
        E_i := E_i ∪ k;
        IF (v' ∉ V_i) THEN
            V_i := V_i ∪ v';
            bilde_komponente(v', V_i, E_i, E);
        END;
    END;
END;
```

Algorithmus 6.2: Zerlegung eines Graphen in zusammenhängende Komponenten

Die Zerlegung eines Graphen in Teilgraphen hat die Komplexität $O(|E|)$.

Besonderheiten des Matchings in der Datenbankanwendung

In Datenbanken werden sehr viele Fälle auftreten, in denen die *Zuordnung* der ähnlichen Komponenten nach Abschätzung durch Heuristiken *eindeutig* ist. In diesen Fällen gibt es ein Entity oder Relationship der einen Datenbank, das Ähnlichkeit zu einem Entity beziehungsweise Relationship der anderen Datenbank aufweist. Beide Datenbankkomponenten weisen keine weiteren Ähnlichkeiten zu anderen Knoten auf. In diesen Fällen ermittelt man bei der Zerlegung des Graphen in unabhängige Komponenten, Teilgraphen, die nur aus jeweils einem Knoten aus D_1 und einem Knoten aus D_2 bestehen.
Weiterhin wird es viele zusammenhängende Komponenten geben, die nur aus *wenigen Knoten* bestehen.
Große, stark miteinander verbundene Teilgraphen werden selten auftreten, da durch den eingeführten Schwellwert die Anzahl der Kanten im Ähnlichkeitsgraphen stark reduziert wird.
Deshalb ist eine Zerlegung des Ähnlichkeitsgraphen in unabhängige Komponenten sinnvoll, um das Problem zu untergliedern. In der Datenbankanwendung wird dadurch häufig eine Untergliederung in viele kleine Teilgraphen erreicht, für diese kann der Matchingalgorithmus ausgeführt werden.

Für die Datenbankanwendung müssen nach Ermittlung einer Zuordnung die künstlich hinzugefügte Knoten und Kanten aus dem Matching entfernt werden, da diese hinzugefügten Kanten beim Matching zur Suche nach einem perfekten Matching dienen, aber keine Bedeutung in den Datenbanken haben.

6.4.4 Beispiel für die Bestimmung ähnlicher Teildatenbanken

Die Ermittlung von ähnlichen Datenbankteilen durch die Ähnlichkeitsfunktion und den Matchingalgorithmus soll an einem umfangreichen Beispiel demonstriert werden.

Zu vergleichende Datenbanken. Es werden zwei einfache Universitätsdatenbanken zugrunde gelegt, für diese wird versucht, gleiche Teile zu identifizieren. Die Abbildungen 6.11 und 6.12 zeigen die beiden Datenbank-Schemata.

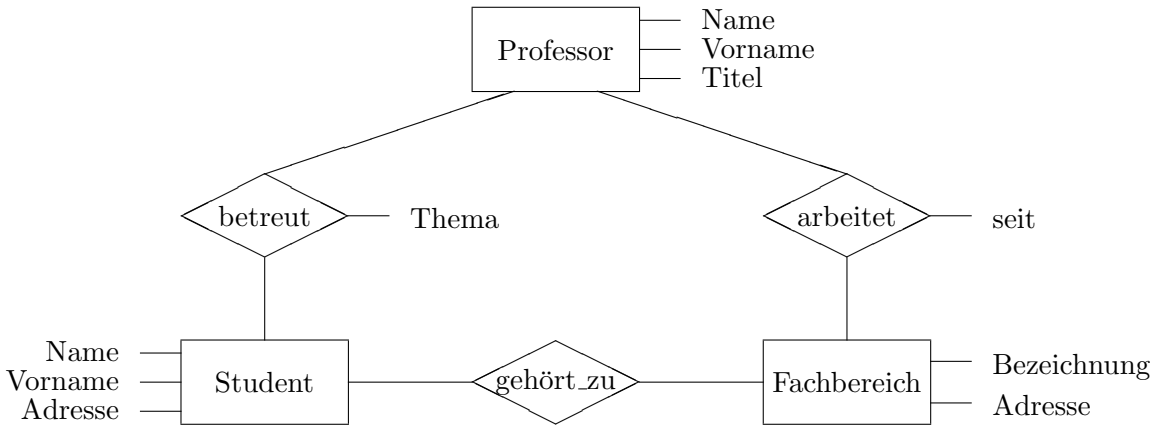


Abbildung 6.11: Beispieldatenbank Universität1

Die zweite Universitätsdatenbank in Abbildung 6.12 ist einem Beispiel aus [BCN92] ähnlich.

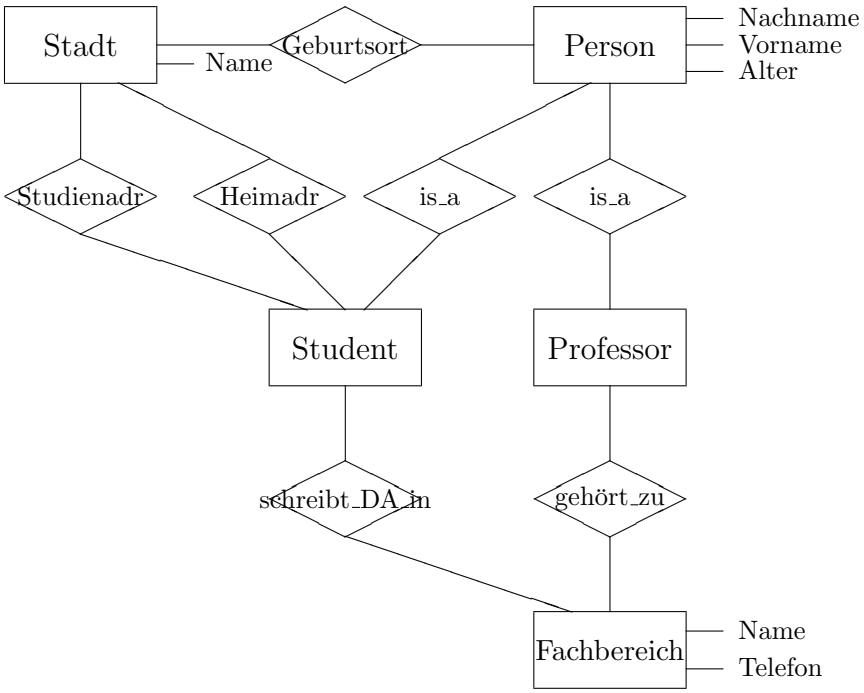


Abbildung 6.12: Beispieldatenbank Universität2

Aufbau eines bipartiten Ähnlichkeitsgraphen. Zur Bestimmung ähnlicher Komponenten sollen hier nur die Bezeichnungen in der Datenbank (Attributnamen, Entitynamen, Relationshipnamen) ausgewertet werden.

Durch Anwendung der Ähnlichkeitsfunktion für Entities und Relationships auf diese Datenbanken entsteht der Ähnlichkeitsgraph in Abbildung 6.13.

Es gibt weitere Komponenten, für die nach der Ähnlichkeitsfunktion aufgrund gleicher Attributbezeichnungen ein Zusammenhang ermittelt wurde.

- *sim*(Universität1.Professor ≈Universität2.Fachbereich)=17
- *sim*(Universität1.Student ≈Universität2.Fachbereich)=17
- *sim*(Universität1.Professor ≈Universität2.Stadt)=17
- *sim*(Universität1.Student ≈Universität2.Stadt)=17

Diese weiteren Kanten des Graphen liegen jedoch unter dem Schwellwert ε=25 und wurden deshalb nicht aufgenommen.

Die *Berechnung der Kantengewichtung* in Abbildung 6.13 soll anhand eines *Beispiels* vorgeführt werden. Für die Ähnlichkeit zwischen Universität1.gehört_zu und Universität2.schreibt_DA_in wird folgende Formel angewendet:

sim(gehört_zu ≈schreibt_DA_in) :=

$\frac{1}{4}$ *sim*(Relationshipnamen) + $\frac{1}{4}$ *sim* (Ähnlichkeit zugeh. Attribute) +
$\frac{1}{4}$ *sim*(Ähnlichkeit der Entities Student) + $\frac{1}{4}$ *sim*(Ähnlichkeit der Entities Professor)

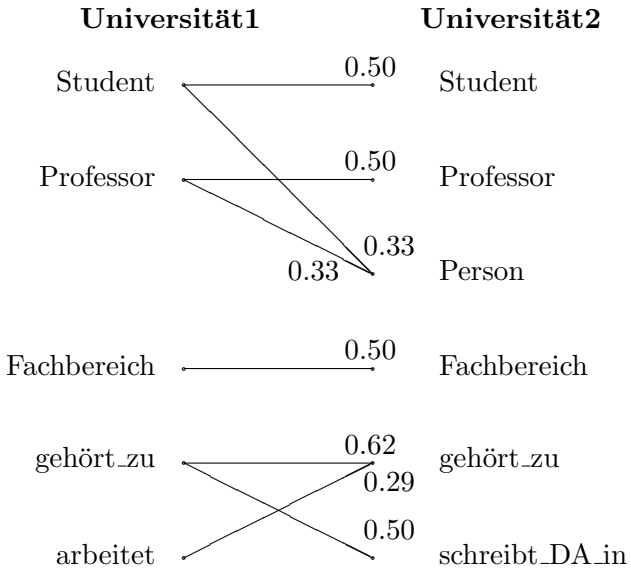


Abbildung 6.13: Bipartiter Ähnlichkeitsgraph der Datenbanken Universität1 und Universität2

Daraus ergibt sich:

$$\text{sim (gehört\_zu} \approx \text{schreibt\_DA\_in)} := \tfrac{1}{4} * 0 + \tfrac{1}{4} * 1 + \tfrac{1}{4} * 0.50 + \tfrac{1}{4} * 0.50 = 0.50$$

Anmerkung: Da in den Datenbanken keine Kardinalitäten bekannt sind, wird die Ähnlichkeit der zugehörigen Entities des Relationships hier höher gewichtet als in Abschnitt 6.4.1 angegeben.

Matchingalgorithmus. Der entstandene Ähnlichkeitsgraph kann nach dem Algorithmus zur Zerlegung des Graphen in drei zusammenhängende Komponenten (Abbildung 6.14) unterteilt werden.

Für diesen Graphen kann nach dem Algorithmus zur Bestimmung eines guten Matchings das in Abbildung 6.14 dargestellte Matching ermittelt werden.

Nach dem Verfahren zum Finden eines guten Matchings werden folgende Matchings der Komponenten ermittelt:

- Das Matching der zweiten Komponente ist trivial, es gibt hier nur eine mögliche Zuordnung, diese ist auch die maximale.
- In der ersten Komponente ist das ermittelte Matching das maximale Matching.
- In der dritten Komponente wird durch das Vorgehen zur Ermittlung eines guten Matchings kein maximales Matching gefunden. Das Matching mit den Kanten gehört_zu — schreibt_DA_in und arbeitet — gehört_zu hat das Gewicht 0.79, dieses Gewicht ist also größer als das Gewicht des ermittelten Matching.
  Da für diese Komponente kein maximales Matching ermittelt wurde, wird deshalb anschließend die Ungarische Methode für diese Komponente kurz dargestellt.

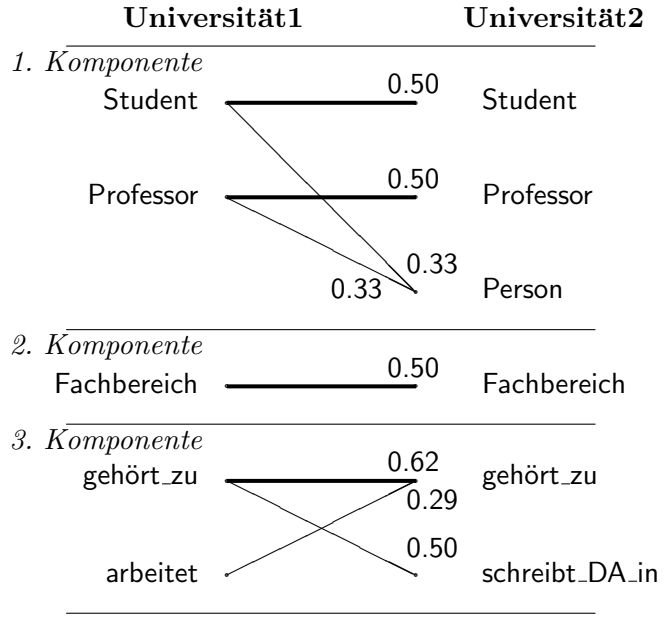


Abbildung 6.14: Gutes Matching des Ähnlichkeitsgraphen

Die Initialwerte für die Vektoren u und v sind in 6.5 dargestellt:

$$\begin{pmatrix} 0.62 & 0.50 \\ 0.29 & 0 \end{pmatrix} \quad \begin{pmatrix} 0.62 \\ 0.29 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 \end{pmatrix} \tag{6.5}$$

Für folgende Kanten des Graphen ist die Gleichheit in Formel 6.3 erfüllt:

- (gehört_zu (1) — gehört_zu (1'))
- (arbeitet (2) — gehört_zu (1'))

Es ist kein perfektes Matching des Graphen über diesen Kanten möglich. Folgende Knotenmenge in Universität1 beinhaltet mehr Knoten als Nachfolger dieser Knotenmenge in Universität2 existieren:

J= (gehört_zu (1), arbeitet(2)), $N(J)$= gehört_zu (1')

Daraus ergibt sich der Wert $\delta = min\ \{\ 0.12,\ 0.29\} = 0.12$ und die Vektoren u und v werden wie folgt verändert:

$$\begin{pmatrix} 0.62 & 0.50 \\ 0.29 & 0 \end{pmatrix} \quad \begin{pmatrix} 0.5 \\ 0.17 \end{pmatrix} \qquad \begin{pmatrix} 0.12 & 0 \end{pmatrix} \tag{6.6}$$

Für folgende Kanten ist jetzt die Gleichheit in 6.3 erfüllt:

- (arbeitet (2) — gehört_zu (1'))
- (gehört_zu (1) — schreibt_DA_in (2'))

Für diesen Graphen ist ein perfektes Matching möglich, dieses umfaßt genau die beiden markierten Kanten. Das ermittelte maximale Matching für den gesamten Ähnlichkeitsgraphen ist in Abbildung 6.15 dargestellt.

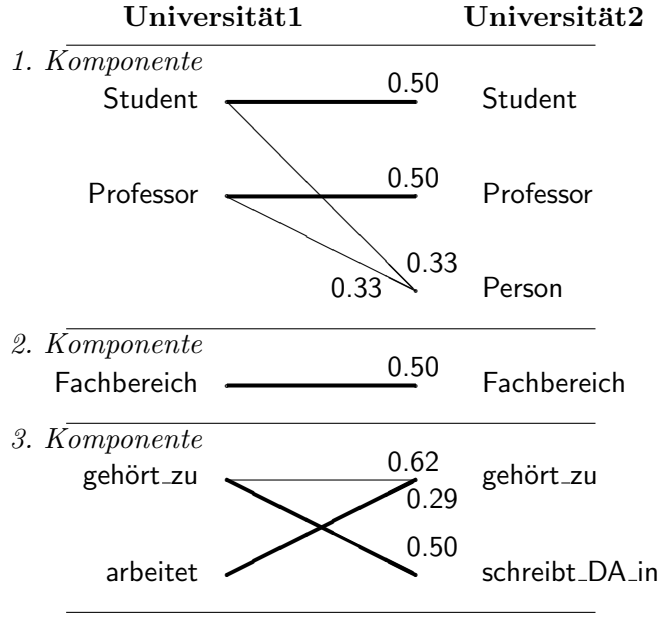


Abbildung 6.15: Maximales Matching des Ähnlichkeitsgraphen

Das maximale Matching des Ähnlichkeitsgraphen in Abbildung 6.15 wird im Datenbankkontext wie folgt interpretiert. Die Knoten des Graphen, die durch das Matching berührt werden, sind Teil der ähnlichen Datenbanken Universität1' und Universität2'.

Das *Gesamtähnlichkeitsmaß* für diese Datenbanken beträgt nach der Zielfunktion des Matching 2.29.

Die Anwendung dieser ähnlichen Datenbankteile wird in Abschnitt 6.7.3 erläutert, dort wird das Beispiel fortgesetzt, nachdem zunächst die Verwendung ähnlicher Datenbankteile in Abschnitt 6.5 und die Validierung in Abschnitt 6.6 theoretisch erläutert werden.

Vor- und Nachteile der Verwendung eines Graphmatchingalgorithmus

Der Einsatz eines Graphmatchingalgorithmus ist sinnvoll, da erprobte Konzepte und bekannte Algorithmen der Graphentheorie auf den Datenbankbereich angewendet werden.

Es handelt sich bei dem hier beschriebenen Vorgehen um eine *offene Methode*, es ist möglich, *Datenbanken in anderen Modellen* (oder allgemein: andere Konstrukte) miteinander zu vergleichen. Dabei benötigt man nur eine entsprechende angepaßte Ähnlichkeitsfunktion.

Ein weiterer Vorteil der Methode ist, daß man Datenbanken mit unterschiedlicher Granularität miteinander vergleichen kann.

Es ist durch das Verfahren auch möglich, *heterogene Datenbanken* (Datenbanken in unterschiedlichen Datenmodellen, z.B. Entity-Relationship Modell und objektorientierte Modelle) miteinander zu *vergleichen* und ähnliche Datenbankteile zu identifizieren. Es muß dazu jedoch eine geeignete Ähnlichkeitsfunktion verwendet werden, der Vergleich von Datenbanken in unterschiedlichen Datenmodellen durch eine einheitliche Ähnlichkeitsfunktion ist jedoch sehr schwierig. Die Übernahme und Anpassung von gefundenen gleichen oder zueinander ähnlichen Teilen ist dann ebenfalls sehr kompliziert.

Problematisch bei der Verwendung des Graphmatchingalgorithmus ist, daß die *Verbindung der Entities und Relationships innerhalb des Datenbank-Schemas* (bzw. in anderen Modellen zwischen anderen Konstrukten) in dem Verfahren nicht direkt beachtet wird. Diese Einschränkung ist notwendig, um das Matchingproblem lösbar zu machen. In dem hier angegebenen Fall fließt die Verbindung der Relationships zu den zugehörigen Entities über die Ähnlichkeitsfunktion für Relationships in den Matchingalgorithmus ein. In dieser wird zur Bestimmung der Ähnlichkeit von Relationships der Kontext in den Datenbanken (zugehörige Entities und Kardinalitäten zwischen Relationships und Entities) berücksichtigt.

6.4.5 Suchraumeinschränkung bei der Ermittlung ähnlicher Datenbankteile

Bei der vorgestellten Ermittlung ähnlicher Datenbankteile sind sehr viele Vergleichsoperationen erforderlich, da paarweise alle Entities/Relationships der Datenbank verglichen werden. Bei der Suche nach ähnlichen Datenbankteilen in Bibliotheken ist dieser Aufwand nicht vertretbar oder sogar nicht möglich. Im folgenden werden deshalb einige Möglichkeiten vorgestellt, um die Suche nach ähnlichen Datenbankteilen zu effektivieren. Dabei sollen möglichst wenige verwendbare Informationen verloren gehen.

Effektivierung des Aufbaus eines Ähnlichkeitsgraphen. Man kann die Anzahl der zu vergleichenden Entity- und Relationshippaare reduzieren, indem man den *Aufbau des Ähnlichkeitsgraphen* in *zwei Stufen* vornimmt.
Dabei wird im ersten Schritt die *Ähnlichkeit von Entities* paarweise untersucht, der Graph wird entsprechend aufgebaut.
Die *Ähnlichkeit von Relationships* wird nur dann untersucht, wenn die zugehörigen Entities in den Ähnlichkeitsgraphen aufgenommen wurden. Damit wird festgelegt, welche Relationships paarweise verglichen werden, der Vergleich erfolgt nicht über allen Relationshippaaren. Der Suchraum zum Vergleichen von Relationships wird stark eingeschränkt, teilweise können dadurch einander ähnliche Relationships nicht erkannt werden. Auch der *Vergleich von Entities und Relationships* untereinander ist dadurch nicht möglich.

Dieses effektive Verfahren schränkt also in einigen Fällen die Menge der Kanten im Ähnlichkeitsgraphen ein, das kann dazu führen, daß eine richtige Zuordnung nicht gefunden wird. Es sollte deshalb nur beim Vergleich von großen Datenbanken angewendet werden, wenn das vollständige Vergleichen der Relationships zu zeitaufwendig ist.

Speicherung von Angaben zum Anwendungsgebiet. Eine Suche nach ähnlichen Teildatenbanken kann gezielter erfolgen, wenn man zu jeder Datenbank speichert, für welches *Anwendungsgebiet* oder für welche Anwendungsgebiete diese erstellt wurde. Sucht man ähnliche Datenbankteile, so werden zuerst Datenbanken verglichen, die das gleiche oder ein ähnliches Anwendungsgebiet haben. Auf diese Weise können jedoch interessante Analogien zwischen verschiedenen Anwendungsgebieten verloren gehen. Besser als eine Einschränkung des Suchraumes durch das Anwendungsgebiet ist deshalb eine *Sortierung des Suchraumes* aufgrund des Anwendungsgebietes. Dabei wird zuerst versucht, Ähnlichkeiten zu Datenbanken des gleichen oder eines ähnlichen Anwendungsgebietes zu finden. Werden dort keine Analogien ermittelt, so werden die anderen gespeicherten Datenbank-Schemata untersucht. Das Verfahren kann solange fortgesetzt werden, bis ein ähnliches Datenbank-Schema gefunden wurde oder alle vorhandenen Datenbank-Schemata überprüft wurden.

Automatische Ermittlung von Hauptinhalten einer Datenbank. Im vorigen Abschnitt wurde eine Suchraumeinschränkung erläutert, die auf explizit vom Benutzer angegebenen Anwendungsgebieten beruht. Es ist auch möglich, durch Auswertung von Heuristikregeln und ohne Interaktion mit dem Benutzer festzustellen, welche Entities oder Relationships am wichtigsten in einer Datenbank sind. Das Vorgehen dazu wird in Abschnitt 7.7 als Suche nach Kernrelationen[1] einer Datenbank erläutert. Kernrelationen sind Entities oder Relationships einer Datenbank, die in der Regel die Hauptinhalte der Datenbank repräsentieren. Die Bezeichnungen dieser Entities und Relationships können als *automatisch ermittelte Stichworte* über das entsprechende Datenbank-Schema verwendet werden. Mit diesen Informationen kann ebenfalls die oben angegebenen Suchraumsortierungen vorgenommen werden.

Einbeziehung von Metainformationen. In Kapitel 7 dieser Arbeit wird beschrieben, wie sich Metainformationen (Kernrelationen, Cluster [2]) über die Integritätsbedingungen des Datenbank-Schemas ableiten lassen. Diese Metainformationen können auch verwendet werden, um ähnliche Datenbanken zu ermitteln, ohne dabei alle Datenbankeinheiten paarweise zu vergleichen.

Man kann folgendes Vorgehen verwenden, um in zwei Datenbanken D_1 und D_2 ähnliche Teile $D_1' \subseteq D_1$ und $D_2' \subseteq D_2$ mit $D_1' \approx D_2'$ zu finden:

1. Es werden die Kernrelationen als wichtigste Einheiten der Datenbanken D_1 und D_2 ermittelt, in Abschnitt 7.7 wird das Vorgehen erläutert. Dabei werden verschiedene Heuristiken eingesetzt, um diese zu finden. Auf diese Weise versucht man, die *Inhalte der Datenbanken* zu erfassen.

2. Zwischen zwei Datenbanken wird versucht, *gleiche oder ähnliche Kernrelationen* zu finden. Für die Ermittlung von Ähnlichkeiten zwischen den Kernrelationen wird das in Abschnitt 6.4.1 beschriebene Abstandsmaß verwendet.
   Kann man ähnliche Kernrelationen identifizieren, so gibt es ähnliche Inhalte in den Datenbanken. Man möchte deshalb die ähnlichen Teile der Datenbanken lokalisieren.

3. Um nicht paarweise alle Entities/Relationships der Datenbanken D_1 und D_2 auf Ähnlichkeiten untersuchen zu müssen, wird folgendes Vorgehen gewählt. Es werden *Cluster*

[1] Kernrelationen - wichtigste Einheiten einer Datenbank, Kapitel 7
[2] Cluster - inhaltlich zusammenhängende Teile einer Datenbank, Kapitel 7

um die gefundenen ähnlichen Kernrelationen gebildet, dazu werden durch Heuristiken inhaltlich zusammenhängende Teile um die Kernrelationen herum bestimmt.

4. Zwischen den *Clustern* wird versucht, *Ähnlichkeiten* festzustellen. Dazu werden wieder die in Abschnitt 6.4.1 vorgestellten Heuristiken verwendet, die die Ähnlichkeit zwischen jeweils zwei Entities bzw. Relationships der Cluster ermitteln. Es wird ein bipartiter Graph aufgebaut, in dem jeweils die ermittelten Ähnlichkeitsfaktoren eingetragen werden. Durch einen Graphmatchingalgorithmus wird eine eineindeutige Zuordnung von einander ähnlichen Teilen ermittelt.

5. Wurden in den Clustern Ähnlichkeiten gefunden, so ist es möglich, daß in der *Umgebung* der Cluster in den Datenbanken weitere ähnliche Teile vorhanden sind. Diese werden anschließend in analoger Form gesucht. Dazu werden alle Knoten der Datenbanken untersucht, die direkte Nachbarn von ermittelten ähnlichen Knoten sind. Diese sind durch Pfade der Länge 1 mit Knoten, die im Ähnlichkeitsgraphen enthalten sind, verbunden. Das Vorgehen wird inkrementell fortgesetzt, bis keine weiteren ähnlichen Knoten mehr gefunden werden.

Durch diese vier vorgestellten Möglichkeiten versucht man den Suchraum so einzuschränken, daß möglichst wenig Informationen verloren gehen.

6.5 Reuse: Übernahme von Informationen aus ähnlichen Datenbanken

Die Wiederverwendung von Datenbanken kann *alle Teilaufgaben des Entwurfes bzw. Wiederentwurfes* von Datenbanken unterstützen, da dabei aus bereits bekannten Datenbanken Lösungen übernommen werden können. Der Strukturentwurf von Datenbanken kann durch diese Methode unterstützt werden (Vervollständigung und Erweiterung der Datenbank in Abschnitt 6.5.1). Die Unterstützung der Semantikakquisition durch eine Wiederverwendung von Datenbanken - der Schwerpunkt dieses Arbeit - wird in Abschnitt 6.5.2 gezeigt. Die Ermittlung ähnlicher Teildatenbanken kann auch zur Unterstützung weiterer Entwurfsaufgaben eingesetzt werden, diese werden in diesem Abschnitt aufgezählt und erläutert.

Voraussetzung ist jeweils die Suche nach gleichen oder ähnlichen Teilen $D_1' \subseteq D_1$ und $D_2' \subseteq D_2$. Dabei sei ohne Beschränkung der Allgemeinheit D_1 eine bereits vorhandene Datenbank, D_2 der aktuelle Entwurf.

6.5.1 Ergänzen und Erweiterung von Strukturangaben

Gibt es ähnliche Teildatenbanken D_1' und D_2', so kann anhand dieser mit dem Benutzer die Vollständigkeit der strukturellen Angaben diskutiert werden. Weist D_1' mehr strukturelle Informationen auf als D_2', so kann im Dialog ermittelt werden, ob diese Informationen auch für die Datenbank D_2' relevant sind.

Im einzelnen kann man durch Auswertung ähnlicher Datenbanken folgende *Ergänzungen* der strukturellen Angaben vorschlagen:

- **Ergänzung von Attributen**
  Gibt es in einer Datenbank D_1' ein Entity oder Relationship E_1, das ähnlich einem Entity oder Relationship E_2 in D_2' ist, jedoch mehr Attribute aufweist, so kann dem Entwerfer vorgeschlagen werden, diese zusätzlichen Attribute auch in E_2 zu ergänzen.

- **Ergänzung von Pfaden**
  Gibt es in einer Datenbank D_1' zwei Entities oder Relationships, zu denen in D_2' ähnliche Entities oder Relationships existieren und in D_1' ist zwischen diesen Komponenten ein Pfad vorhanden, der in D_2' nicht existiert, so kann man dem Entwerfer vorschlagen, diesen Pfad auch in D_2' zu ergänzen.

Auf diese Weise können systeminitiierte Vorschläge zur Vervollständigung von Datenbanken gemacht werden. Es können ebenfalls strukturelle *Erweiterungen* durch Auswertung ähnlicher Datenbanken vorgeschlagen werden können. Kennt man in zwei Datenbanken D_1 und D_2 ähnliche Teile D_1' und D_2', so kann man aus dieser Information dem Benutzer eine Erweiterung des Entwurfes durch eine *insight-out-Strategie* vorschlagen. Man weiß, daß die Entities und Relationships der Datenbankteile $D_1 - D_1'$ nicht in der Datenbank D_2 vorkommen, sonst wären sie in D_1' enthalten. Deshalb kann man folgende Erweiterungen von D_2 vorschlagen:

- **Ergänzung von Relationships**
  Ein Relationship R aus $D_1 - D_1'$, dessen zugehörige Knoten (Entities oder bei Modellen mit Higher-Order-Relationships auch Relationships) alle in D_1' liegen, kann auch in D_2 zur Ergänzung vorgeschlagen werden.

- **Ergänzung von Entities und Relationships**
  Man kann auch ganze Teildatenbanken (zusammenhängende Entities und Relationships) aus einer Datenbank D_1 in D_2 übernehmen, wenn es einen gleichen oder ähnlichen Knoten in D_1' und D_2' gibt.
  Es werden Knoten E_1 in D_1' gesucht, die eine direkte Verbindungen zu Knoten in $D_1 - D_1'$ haben. Es wird vorgeschlagen, diese Knoten an den zu E_1 ähnlichen Knoten E_2 in der Datenbank D_2 zu ergänzen.

Auf diese Weise kann man sinnvolle Entwurfsvorschläge für die Erweiterung struktureller Angaben erreichen. Geltende ähnliche Datenbankteile können verwendet werden, um eine umfangreiche Entwurfsunterstützung zu realisieren.
Im Abschnitt 6.7 wird gezeigt, wie dieses durch Bibliotheken (Erstellung von Bibliotheken und Anwendung dieser zur Entwurfsunterstützung) erfolgen kann.

6.5.2 Übernahme von Integritätsbedingungen

Im Mittelpunkt dieser Arbeit steht die Suche nach Integritätsbedingungen in Datenbanken. Auch für diese Aufgabe kann die Suche nach gleichen oder ähnlichen Datenbanken $D_1' \approx D_2'$ verwendet werden. Werden gleiche oder ähnliche Teile von Datenbanken gefunden, so kann man annehmen, daß diese die gleiche Semantik haben und deshalb die gleichen Integritätsbedingungen gelten. Die bekannten Integritätsbedingungen einer Teildatenbank D_1' können deshalb in eine gleiche oder ähnliche Teildatenbank D_2' übernommen werden. Die Methode kann eingesetzt werden, um *sinnvolle Kandidaten für Integritätsbedingungen* zu finden, es muß jedoch eine Validierung erfolgen.

Die folgende Übersicht zeigt, welche Integritätsbedingungen nach Finden gleicher oder ähnlicher Teile als Kandidaten übernommen werden können:

- **Funktionale Abhängigkeiten**
  Wurden zwei ähnliche Attributmengen $A_{11}..A_{1n}$ in D'_1 und $A_{21}..A_{2n}$ in D'_2 gefunden, und eine funktionale Abhängigkeiten $A_{1i} \longrightarrow A_{1j}, i, j \subseteq 1..n$ ist bekannt, so kann auch die zugehörige funktionale Abhängigkeit in D'_2 gelten.

- **Schlüssel**
  Konnten zwei ähnliche Entities E_1 in D'_1 und E_2 in D'_2 gefunden werden, die gleiche Attribute $A_{11}..A_{1n}$ bzw. $A_{21}..A_{2n}$ haben, und Attribute aus $A_{11}..A_{1n}$ sind Schlüssel in E_1, so können die zugehörigen Attribute aus $A_{21}..A_{2n}$ Schlüssel in E_2 sein.

- **Inklusions- und Exklusionsabhängigkeiten**
  Konnten in zwei Datenbanken oder Datenbankteilen zwei ähnliche Entities oder Relationships E_{11}, E_{12} in D'_1 und E_{21}, E_{22} in D'_2 mit gleichen oder ähnlichen Attributen $A_{11}..A_{1n}$ - Attribute von E_{11}, $B_{11}..B_{1n}$ - Attribute von E_{12}, $A_{21}..A_{2n}$ - Attribute von E_{21}, $B_{21}..B_{2n}$ - Attribute von E_{22} gefunden werden und es gibt eine Inklusions- oder Exklusionsabhängigkeit $A_{1i} \subseteq B_{1i}$ bzw. $A_{1i} \mid\mid B_{1i}$ in D'_1, dann muß auch die Inklusions- oder Exklusionsabhängigkeit $A_{2i} \subseteq B_{2i}$ bzw. $A_{2i} \mid\mid B_{2i}$ in D'_2 überprüft werden.

- **Kardinalitäten**
  Konnten in zwei Datenbanken gleiche Relationships und zugehörige gleiche Entities E_1, R_1 in D'_1 und E_2, R_2 in D'_2 gefunden werden und die Kardinalität $\mathsf{card}(R_1, E_1)$ gilt in D'_1, so kann man annehmen, daß $\mathsf{card}(R_2, E_2)$ in D'_2 den gleichen Werte annimmt.

Im Kapitel 5 wurden Methoden zur Ableitung von Kandidaten für Integritätsbedingungen vorgestellt, bei denen neben den Kandidaten selbst auch eine Plausibilität für diese ermittelt wurde. Diese Plausibilität ist einsetzbar, um die Diskussion der Integritätsbedingungen zu steuern.

Auch für die aus ähnlichen Datenbanken abgeleiteten Kandidaten sind Plausibilitäten wünschenswert. Man kann als Plausibilitätsfunktion das Ähnlichkeitsmaß verwenden. Beispiel: Ein ermittelter Kandidat für eine funktionale Abhängigkeit in E_2 aus D_2, die in E_1 aus D_1 existierte, erhält so eine Plausibilität, die sich aus dem ermittelten Ähnlichkeitsmaß der Entities E_1 und E_2 und der Ähnlichkeit der Attributmengen, über denen die funktionale Abhängigkeit definiert ist, ergibt.

Die aus D'_1 übernommenen Integritätsbedingungen müssen in D'_2 validiert werden, da die Bestimmung der ähnlichen Teildatenbanken $D'_1 \approx D'_2$ sich auf vage Informationen stützt. Damit wird sichergestellt, daß keine falschen Kandidaten abgeleitet werden können. Zur Validierung können die in Kapitel 9 beschriebenen Methoden verwendet werden. Das Kapitel 11 zeigt die Verbindung der Wiederverwendung von Datenbanken mit den anderen Methoden zur Semantikakquisition.

6.5.3 Übernahme von Verhaltensinformationen, Transaktionen, Testdaten, Optimierung

Ebenso wie die strukturellen Angaben und Integritätsbedingungen können auch Angaben zum Verhalten der Datenbank, Beispieltransaktionen, Optimierungsschritte und auch Testdaten

übernommen werden, sofern diese in einer existierenden ähnlichen Datenbank vorhanden sind. Voraussetzung ist auch hier die Suche nach gleichen oder ähnlichen Datenbankteilen. Da sich diese Suche nach ähnlichen Datenbankteilen auf vage Informationen stützt, muß auch bei diesen Informationen eine Überprüfung und Anpassung bei der Übernahme erfolgen.

6.5.4 Reihenfolge der Übernahme von Informationen

Es gibt Anwendungen, bei denen nur eine Art von Informationen durch die Wiederverwendung vorgeschlagen werden soll, z.B. die Strukturangaben, die Integritätsbedingungen, Verhaltensinformationen, Optimierungen usw.

Man kann durch die Wiederverwendung von vorhandenen Datenbank-Schemata jedoch den vollständigen Entwurfsprozeß unterstützen. Sollen dabei mehrere Arten von Informationen übernommen werden, so sollte man mit den einfacher nachvollziehbaren Angaben (Struktur) beginnen.

Wurde die Übernahme von Informationen mit dem Benutzer diskutiert und erfolgreich durchgeführt, so ist das ein Hinweis darauf, daß die ermittelten Ähnlichkeiten (die sich auf vage Informationen stützten) richtig waren. Es ist in diesen Fällen häufig möglich, weitere Informationen aus dem gleichen Datenbank-Schema zu übernehmen.

Man kann jedoch die Übernahme von Informationen (z.B. Verhaltensinformationen) nicht auf solche Datenbank-Teile beschränken, aus denen sich z.B. auch die Strukturinformationen ableiten ließen. Würde man das machen, so könnte man z.B. keine Übernahme von identischen Verhaltensinformationen in unterschiedlichen Anwendungen finden.

Die erfolgreiche Übernahme von einer Art von Informationen verspricht also, daß auch weitere Informationen übernommen werden können. War die Übernahme einer Art von Informationen nicht möglich, so heißt das jedoch nicht, daß keine Informationen aus dem entsprechenden Datenbank-Schema übernommen werden können.

6.6 Revise: Validierung der übernommenen Informationen

Alle Informationen, die in der Phase Reuse übernommen wurden, müssen validiert werden. Dazu kann es teilweise möglich sein, die Korrektheit dieser Lösungen automatisch zu überprüfen. Z.B. kann man bei der Übernahme von Integritätsbedingungen überprüfen, ob diese zueinander konfliktfrei sind und ob sie konform mit bekannten Daten sind.

Es ist jedoch für keine Entwurfsaufgabe möglich, das Überprüfen der übernommenen Lösungen vollständig automatisch durchzuführen. Deshalb ist es immer erforderlich, die Lösungen mit dem Entwerfer zu diskutieren und dadurch zu überprüfen und evt. auch zu vervollständigen.

Für die Überprüfung vorgeschlagener Integritätsbedingungen kann eine Erfragung der Gültigkeit von Beispieldatenbanken und eine pseudonatürlichsprachige Erfragung eingesetzt werden. Diese wird in Kapitel 9 gezeigt.

6.7 Retain: Erstellung und Verwendung von Bibliotheken

Für die Übernahme von Informationen aus vorhandenen Datenbanken ist es notwendig, zahlreiche Datenbanken zu speichern. Dieses kann sinnvollerweise in Bibliotheken erfolgen. In Abschnitt 6.7.1 wird angegeben, wie solche Bibliotheken aufgebaut werden, die Verwendung dieser wird in Abschnitt 6.7.2 beschrieben. Im Abschnitt 6.7.3 folgt ein Beispiel für das Vorgehen.

6.7.1 Erstellung von Bibliotheken zur Entwurfsunterstützung

In Datenbanken, die einander ähnlich sind, werden gleiche und unterschiedliche Aspekte auftreten. Diese möchte man in Bibliotheken speichern, um sie für spätere Entwürfe wiederzuverwenden. Die Beschränkung auf gemeinsame Teile und Speicherung der gemeinsamen Teile als Musterdatenbanken, die in [SDG95] gewählt wurde, bedeutet, daß bereits vorhandene Informationen verloren gehen. Deshalb ist es empfehlenswert, nicht nur die gemeinsamen Aspekte, sondern auch die unterschiedlichen Gesichtspunkte der Datenbanken in Bibliotheken zu speichern, um diese wiederzuverwenden.

Erstellung des obligatorischen Teiles

Es ist möglich, zu verschiedenen Anwendungsgebieten Musterentwürfe in Bibliotheken zu speichern. Diese kann man aus verschiedenen Datenbanken $D_1..D_n$ zu einem Gebiet ableiten. Mit der in diesem Kapitel beschriebenen Methode sind die gemeinsamen (gleichen oder ähnlichen) Teile der Datenbanken $D_1..D_n$ ermittelbar. Es werden dazu die Datenbankteile $D_1..D_n$ gesucht, die folgende Bedingung erfüllen: *maximiere sim* $(D'_1, D'_2) \wedge .. \wedge$ *sim* (D'_1, D'_n).

Bilden abgeschlossener Datenbanken. Die gespeicherten *Datenbanken in der Bibliothek* müssen *in sich abgeschlossen* sein, es gibt eine Besonderheit, die dazu gegebenenfalls behandelt werden muß:
Es kann sein, daß in $D'_1..D'_n$ Relationships auftreten, ein zugehöriges Entity aus $D_1..D_n$ kommt jedoch nicht vor. In diesem Fall ist es sinnvoll, auch das zugehörige Entity in D_1 aufzunehmen, damit die Datenbank D'_1 abgeschlossen ist und kein Relationship "in der Luft hängt".

Klärung der Synonyme zwischen den Datenbanken. Will man ermittelte ähnliche Teildatenbanken als allgemeingültige Datenbank D' zu einem Anwendungsgebiet speichern, so ist es sinnvoll, dazu nicht einfach eine der Teildatenbanken D'_i auszuwählen, sondern bei nichtübereinstimmenden Bezeichnungen in den Entities und Relationships der Datenbank, die durch die Ähnlichkeitsfunktion als Synonyme erkannt wurden, im Dialog mit dem Benutzer eine *allgemeingültige Bezeichnung* zu finden.

Klärung der Zugehörigkeit von Attributen. Weiterhin soll bei Entities oder Relationship, die als ähnlich identifiziert wurden, aber unterschiedliche Attribute ausweisen, eine Nachfrage erfolgen, welche *Attribute* für eine allgemeine Datenbank zutreffend sind. Das ist besonders dann erforderlich, wenn Bezeichnungen der zugehörigen Entities oder Relationships der Datenbank aufgrund der Klärung von Synonymen verändert wurden, da sich auf diese Weise auch die Bedeutung der Entities und Relationships etwas ändern kann.

Speicherung optionaler Teile

Neben der Speicherung des gemeinsamen Teiles von mehreren Datenbanken zu einem Gebiet ist es sinnvoll, auch die Teile zu speichern, die nur in einer oder in einigen Datenbanken zu dem Gebiet vorkommen ($D_1 - D'_1, D_2 - D'_2, .., D_n - D'_n$). Dabei wird ebenfalls gespeichert, wie häufig diese optionalen Teile auftreten. Auf diese Weise erhält man zu jedem Anwendungsgebiet einen *obligatorischen Teil*, sowie *verschiedene optionale Teile*.

Auch diese optionalen Teile sollen *abgeschlossene Datenbanken* darstellen, es müssen also zu allen Relationships der Teildatenbanken ebenfalls alle zugehörigen Entities in die optionalen Teildatenbanken aufgenommen werden. Dabei werden für die ergänzten Knoten die Bezeichnungen des obligatorischen Teiles, die vom Benutzer eingegeben wurden, verwendet.
Auf diese Weise kann auch die *Verbindung zwischen dem obligatorischen Teil und jeder optionalen Teildatenbank* hergestellt werden, da in diesen Teilen Entities mit gleichen Bezeichnungen auftreten.

6.7.2 Entwurfsunterstützung durch Bibliotheken

Mit Hilfe solcher Bibliotheken kann man dem Benutzer sinnvolle Entwurfsvorschläge unterbreiten, dadurch den Entwurf einer Datenbank beschleunigen und die Wiederholung von Fehlern vermeiden.

Beim Entwurf von Datenbanken wird folgendes Vorgehen gewählt:
Der gespeicherte obligatorische Teil zu einem Gebiet wird als Ausgangspunkt genommen und mit dem Benutzer diskutiert.
Anschließend wird im Dialog mit dem Benutzer untersucht, ob auch vorkommende optionale Teile an den obligatorischen Teil angebunden werden müssen. Es werden dabei die optionalen Teile (nach ihrer Häufigkeit sortiert) diskutiert.

Auf diese Weise erhält man einen konzeptuellen Entwurf einer Datenbank. Die Vorschläge zur schrittweisen Erstellung und Erweiterung werden durch Auswertung der vorhandenen Datenbank-Schemata abgeleitet.

6.7.3 Beispiel für den Aufbau von Bibliotheken (Fortsetzung von 6.4.4)

Das in Abschnitt 6.4.4 gezeigte Beispiel soll hier für die Erläuterung des Aufbaues von Bibliotheken verwendet werden.

Speicherung des obligatorischen Teiles. Für die Universitätsdatenbanken aus 6.4.4 können die ermittelten gemeinsamen Teile als obligatorischer Teil einer Universitätsdatenbank gespeichert werden.

Durch den Matchingalgorithmus wurden die in Abbildung 6.11 und 6.12 dargestellten Teildatenbanken Universität1 und Universität2 als ähnlich identifiziert. Abbildung 6.16 und 6.17 zeigen die ermittelten ähnlichen Teile.

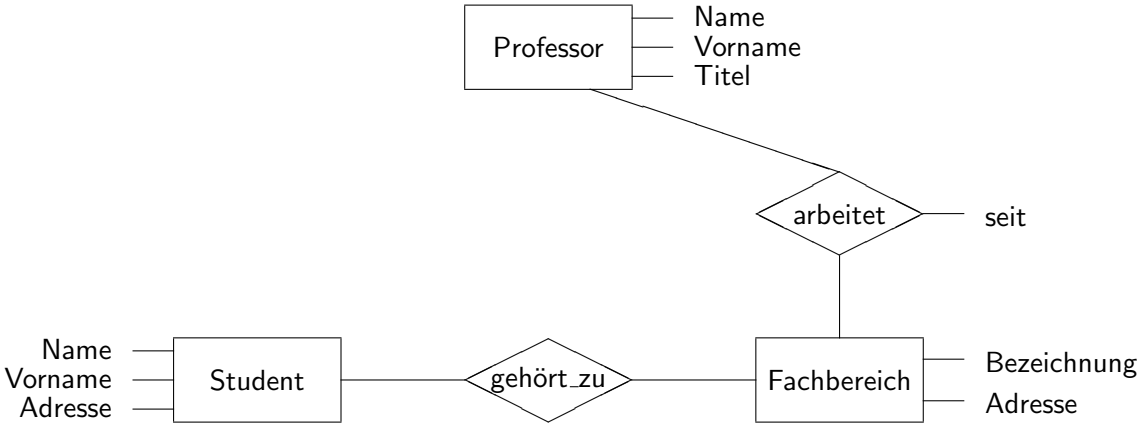


Abbildung 6.16: Beispieldatenbank Universität1'

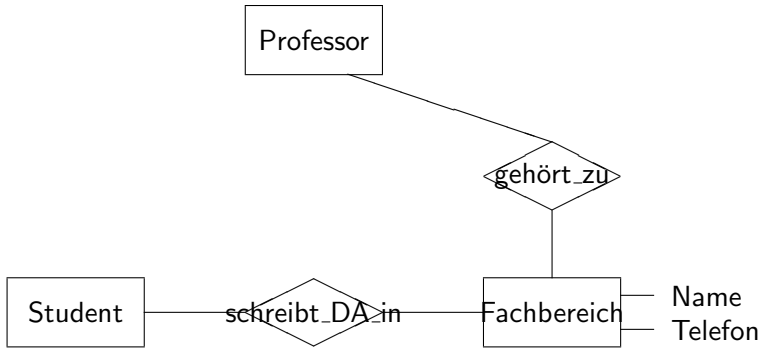


Abbildung 6.17: Beispieldatenbank Universität2'

In diesen Datenbanken existieren Bezeichnungen für Relationships, die in Universität1 und Universität2 unterschiedlich sind. Diese sollen im Dialog mit dem Benutzer durch allgemeingültige Bezeichnungen und Attribute ersetzt werden.
Dazu können folgende Fragestellungen verwendet werden:

- Ähnliche Relationships über den Entities Professor und Fachbereich wurden mit —arbeitet— und —gehört_zu— bezeichnet. Bitte geben Sie eine allgemeine Bezeichnung ein:
  $\Longrightarrow$ ist_angestellt

- Folgende Attribute traten bei dem Entity —Fachbereich— auf, welche sind allgemeingültig ?

  - Bezeichnung ⟸
  - Adresse ⟸
  - Name
  - Telefon ⟸

Nach Klärung der nichtübereinstimmenden Bezeichnungen und Attributzuordnung ist die in Abbildung 6.18 dargestellte Datenbank als allgemeingültiger Teil einer Universitätsdatenbank entstanden:

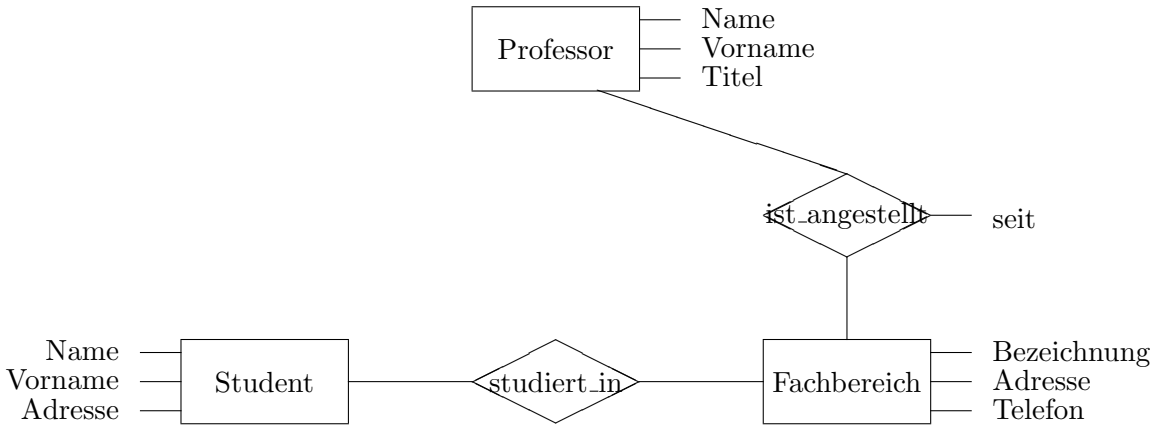


Abbildung 6.18: Obligatorischer Teil der Datenbank Universität

Speicherung optionaler Teile. Die Datenbanken Universität1-Universität1' und Universität2-Universität2' bilden die optionalen Teile für das Anwendungsgebiet Universität. Auch diese sollen gespeichert werden, da ihre Wiederverwendung in neuen Datenbanken möglich ist. Abbildung 6.19 und 6.20 zeigen optionale Teildatenbanken, in denen Entities ergänzt wurden, um die Datenbanken abzuschließen.

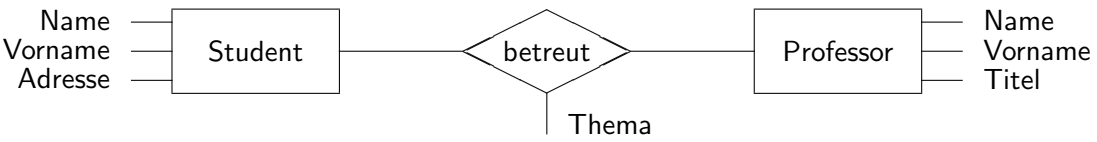


Abbildung 6.19: Optionaler Teil 1 der Datenbank Universität

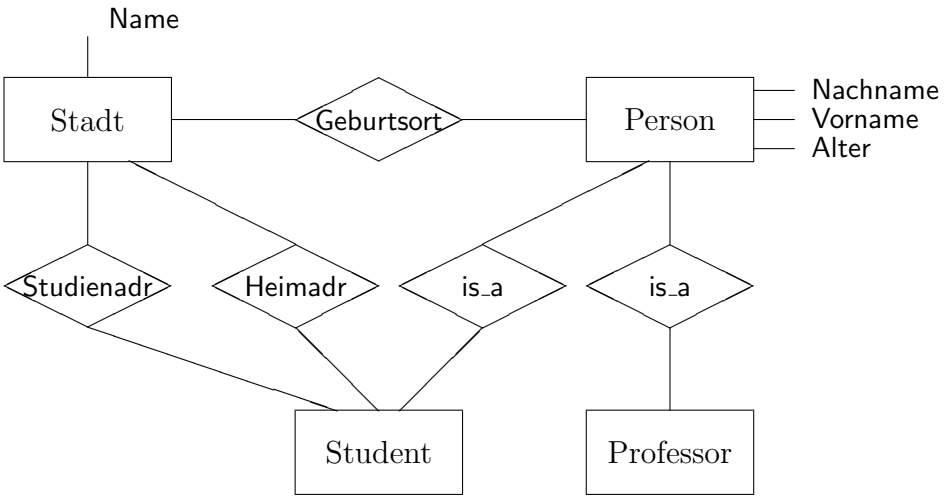


Abbildung 6.20: Optionaler Teil 2 der Datenbank Universität

Veränderte Methode zum Aufbau und Einsatz von Bibliotheken

Es ist möglich, das vorgestellte Verfahren zum Aufbau und zur Verwendung von Bibliotheken durch eine andere Methode zu ersetzen.

Bei dieser Methode werden die entworfenen Datenbank-Schemata nicht komplett gespeichert, sie werden in sinnvolle Einheiten untergliedert, diese werden getrennt gespeichert. Für die Untergliederung der Datenbank-Schemata kann die in Abschnitt 7.5 verwendete Bildung von Relationenclustern verwendet werden. Zu den einzelnen Einheiten wird gespeichert, für welches Anwendungsgebiet oder welche Anwendungsgebiete sie erstellt wurden.

Die Entwurfsunterstützung unter Verwendung der so entstehenden Bibliotheken kann wie folgt durchgeführt werden.

Es wird zunächst erfragt, für welches Anwendungsgebiet das neue Datenbank-Schema erstellt werden soll. Alle vorhandenen Einheiten zu diesem Anwendungsgebiet werden mit dem Benutzer diskutiert, die relevanten Einheiten werden übernommen.
Wurden überlappenden Einheiten übernommen, so kann die Verbindung über die gemeinsamen Teile hergestellt werden. Um diese zu finden, wird zwischen den einzelnen Einheiten mit den beschriebenen Methoden nach ähnlichen Teilen gesucht. Werden solche gefunden, so kann die Vereinigung der einzelnen Teile über diese erfolgen. Dadurch werden auch die Redundanzen zwischen den Teilen beseitigt.

Sind die Einheiten, die der Benutzer für das neue Datenbank-Schema ausgewählt hat, nicht überlappend, so ist eine Vereinigung der einzelnen Teile nicht automatisch möglich, es können auch keine Vorschläge dazu generiert werden. In diesem Fall muß der Benutzer die Vereinigung der Teildatenbanken zu einem Datenbank-Schema vornehmen.

Mit der Entwurfsunterstützung durch Bibliotheken kann ein erster Entwurf und Anregungen für Erweiterungen und Veränderungen realisiert werden. Mit zunehmender Menge auswertbarer gespeicherter Datenbanken wird durch die vorgestellte Entwurfsunterstützung erreicht, daß verschiedene relevante Aspekte für jedes Anwendungsgebiet als gespeicherte und wiederverwendbare Einheit vorliegen.

6.8 Anwendungen der Methode

In diesem Kapitel wurde angedeutet, wie der Strukturentwurf, die Spezifikation von Integritätsbedingungen, Verhaltensinformationen, Transaktionen, Testdaten, Optimierung usw. durch die Auswertung ähnlicher Datenbank-Schemata unterstützt werden können.

Inhalt dieser Arbeit ist nur die Akquisition von Integritätsbedingungen, die vorgestellte Methode bietet jedoch so viel Potential für andere Entwurfsaufgaben, daß ich auf diese geschlossenere Darstellung nicht verzichten wollte. Alle Entwurfsaufgaben können durch den Zugang unterstützt werden, bei dem entsprechende Informationen, die in den vorhandenen Datenbanken bekannt sind, auch für das aktuelle Datenbank-Schema übernommen werden.

Tools, die den Datenbank-Entwurf unterstützen, benötigen sinnvolle Vorschläge für die Entwurfsaufgaben. Durch Auswertung und Übernahme vorhandener Informationen sind diese Vorschläge oft leichter zu generieren als durch das Aufstellen allgemeiner Regeln.

Es gibt weitere Aufgaben des Datenbank-Entwurfes, bei denen ähnliche Datenbankteile identifiziert werden müssen. Diese werden im folgenden kurz erläutert, es kann für diese ein analoges Vorgehen hilfreich sein.

6.8.1 View-Integration, Integration von modular entworfenen Datenbanken

Die Verbindung einzelner Views kann über gleiche Datenbankteile hergestellt werden. Dazu müssen diese identifiziert werden, es kann dazu ein Verfahren wie in Abschnitt 6.4 vorgestellt, verwendet werden. Bei der View-Integration sollte der Schwellwert höher angesetzt werden als bei der Wiederverwendung von Datenbankteilen, da man keine Ähnlichkeit von Komponenten ausnutzen kann, man benötigt identische Teile für sinnvolle Vorschläge zur View-Integration.

Bei der Entwurfsstrategie *Design-by-Units* wird ein *modularer Entwurf* vorgenommen, es werden verschiedene Units entworfen. Diese müssen verbunden werden, dabei ist die Ermittlung identischer Teile ebenso wie bei der View-Integration erforderlich.

6.8.2 Ermittlung von Zusammenhängen in föderierten Datenbanken

In föderierten Systemen versucht man, mehrere Datenbanken (auch in unterschiedlichen Datenmodellen) zu vereinigen. Um dabei Fehler zu vermeiden und sinnvolle Ergebnisse zu erreichen, muß man identische Informationen identifizieren können.
Ein manuelles Vorgehen ist häufig aufgrund der Größe und Anzahl der einbezogenen Datenbank-Schemata nicht möglich. Dafür kann die hier vorgeschlagene Methode zur Ermittlung gleicher oder ähnlicher Datenbankteile verwendet werden. Die Ähnlichkeitsfunktion kann auch auf den Vergleich von Datenbanken in anderen Datenmodellen angepaßt werden. Es ist auch möglich, Datenbanken in unterschiedlichen Datenmodellen miteinander zu vergleichen. Der vorgestellte Matchingalgorithmus kann dazu verwendet werden. In diesem Fall ist es aber wesentlich komplizierter, wirksame Ähnlichkeitsfunktionen aufzustellen. Es lassen sich jedoch Vorschläge für identische Teile generieren, die den Benutzer unterstützen können, die Verbindung zwischen den Datenbanken herstellen kann.

6.8.3 Data Warehouses

In Data Warehouses sollen verschiedene Datenbanken, die sich in verschiedenen Datenmodellen befinden können, ausgewertet werden, um aus diesen bestimmte Entscheidungen abzuleiten.

Dazu müssen zwischen den Datenbanken Beziehungen hergestellt werden, um datenbankübergreifende Auswertungen zu machen. Diese Verbindungen manuell herzustellen, ist sehr aufwendig. Hilfreich ist es deshalb auch in diesem Fall, Vermutungen über identische Teile der Datenbanken automatisch abzuleiten.

6.8.4 Benutzeradaption

Ein Nebeneffekt der Anwendung einer Wiederverwendungskomponente ist eine *Anpassung der Entwurfsunterstützung für alle Entwurfsaufgaben an den Benutzer*. Jeder Benutzer hat einen bestimmten Entwurfsstil und entwirft dementsprechend Datenbanken. Entwirft ein Benutzer zum wiederholten Male auf ähnliche Weise, so können vorhandene Informationen übernommen werden. Dabei erfolgt eine Diskussion von bereits bekannten Lösungen aus analogen Datenbankteilen. Diese erfolgt in dem Sinne, daß Fragen über das Zutreffen der bereits bekannten Lösung für das aktuelle Problem gestellt werden.
Durch Auswertung von Datenbanken, die der gleiche Benutzer entworfen hat, kann dadurch eine *Anpassung an den Benutzer* erreicht werden. Durch Auswertung von Datenbanken des gleichen Anwendungsgebietes kann eine *Anpassung an das Gebiet* erreicht werden.

Eine Benutzeradaption läßt sich auch für die Semantikakquisition erreichen. Integritätsbedingungen können dabei aus ähnlichen Datenbanken, die der gleiche Benutzer entworfen hat, übernommen werden, sodaß wiederholte Eingaben von Integritätsbedingungen über gleichen oder ähnlichen Datenbankteilen vermieden werden. Diese übernommenen Kandidaten für Integritätsbedingungen müssen nur vom Benutzer bestätigt werden.

Kapitel 7

Metainformationen über die Datenbank

In den beiden vorherigen Kapiteln wurde gezeigt, wie unbekannte Integritätsbedingungen mit Hilfe von verschiedenen Heuristikregeln abgeschätzt werden. Man kann neben Kandidaten für Integritätsbedingungen weitere Informationen (*Metawissen oder Metainformationen*) über die Semantik einer Datenbank ableiten. In den Kapiteln 5 und 6 erfolgten bereits Vorgriffe auf dieses Kapitel, da man durch die Metainformationen ebenfalls die Bedeutung der Datenbank erschließen kann und diese deshalb zur Ermittlung von Kandidaten für Integritätsbedingungen eingesetzt werden.

Nach [Sch91] wird *Metawissen* wie folgt definiert.
Metawissen: Teilgebiet Wissensverarbeitung, Expertensysteme
Regeln einer höheren Ebene in Produktionssystemen. Sie realisieren eine *Kontrollstrategie* für Regeln einer niedrigeren Stufe; ein infiniter Regreß wird durch eine Rangfolge-Auswahl auf der höchsten Ebene vermieden.

In diesem Kapitel wird gezeigt, wie Metainformationen über die Datenbank sich durch Auswertung von Heuristikregeln ableiten lassen. Dabei wird versucht, das Hintergrundwissen, das sich in den Charakteristika der Datenbank widerspiegelt, auszuwerten. Die Ermittlung von folgenden Metainformationen wird beschrieben:

- Cluster
  (Attribut-Cluster, Relationen-Cluster)
- voneinander unabhängige Einheiten
  (voneinander unabhängige Attributmengen, voneinander unabhängige Relationenmengen)
- Kernrelationen

Es wird gezeigt, wie diese Metainformationen ermittelt werden können und wie sie eingesetzt werden können.

7.1 Einordnung von Metainformationen in die Diskussion unbekannter Integritätsbedingungen

Abbildung 7.1 zeigt, wie die Metainformationen bei der gezielten Diskussion von unbekannten Integritätsbedingungen eingesetzt werden können.

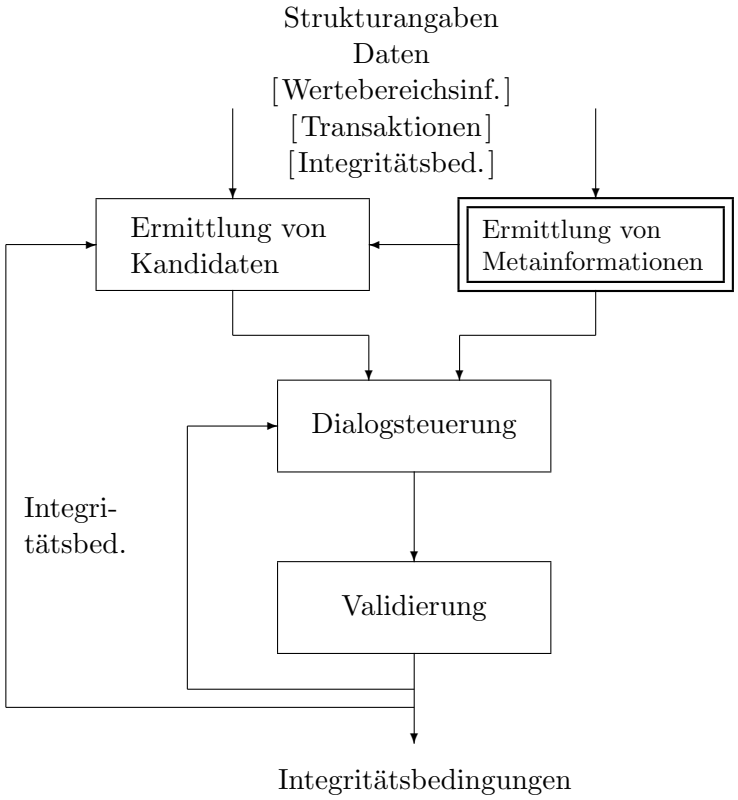


Abbildung 7.1: Einordnung der Metainformationen in die Semantikakquisition

Die Metainformationen werden verwendet, um den Suchraum zur Erfragung von Integritätsbedingungen sinnvoll zu sortieren und so einzuschränken, daß möglichst keine oder wenige Informationen verloren gehen.
Das Metawissen wird deshalb in den Heuristikregeln zur *Ermittlung von Kandidaten* für Integritätsbedingungen ausgewertet. Es wird ebenfalls in der *Dialogsteuerung*, in der die Reihenfolge der Erfragung der abgeschätzten Kandidaten festgelegt wird, angewendet.

7.2 Literaturübersicht

In diesem Abschnitt folgt eine Übersicht, welche bekannten Arbeiten es zur Ermittlung ähnlicher Metainformationen gibt. Diese Arbeiten beziehen sich meist auf ein anderes Anwendungsgebiet, weisen aber Ähnlichkeiten zu dem im Anschluß hier beschriebenen Zugang auf.

Es gibt zahlreiche Methoden zur *Clusterung* und vielfältige Anwendungen der Clusterung von Datenbanken.

Nach [Sch91] ist die Clusteranalyse wie folgt definiert: Clusteranalyse ist ein Sammelbegriff für eine Vielzahl mathematisch-statistischer Verfahren, bei denen eine Menge SO von n Objekten in eine geeignete Anzahl möglichst homogen zusammengesetzter Teilmengen $C_1, C_2, .. \subseteq SO$ (Cluster) zerlegt werden kann, deren Elemente jeweils ähnliche Eigenschaften aufweisen. Dabei sind die Eigenschaften der Objekte durch Daten vorgegeben, z.B. durch p-dimensionale Merkmalsvektoren $X_1..X_p$, durch Relationen, qualitative Beschreibungen oder eine Matrix paarweiser Ähnlichkeiten oder Distanzen vorgegeben. Es gibt disjunkte, überschneidende und hierarchische Cluster, dementsprechend spricht man von partitionierender, nichtdisjunkter oder hierarchischer Klassifikation.

Clusterung von verteilten Datenbanken. Bei *verteilten Datenbanken* erfolgt eine Clusterung der Datenbank auf logischer Ebene. Hierbei ist es das Ziel der Clusterung, *effiziente Zugriffe* auf der verteilten Datenbank durchführen zu können. Eine Methode zur Bestimmung einer vertikalen Partitionierung für verteilte Datenbanken wurde von Hoffer entwickelt und u.a. in [Haf93] und [Zha93] weiterentwickelt und angewendet.
Kriterien zur Clusterung sind in beiden Fällen ausschließlich die Operationen, die zum Ausführen von Transaktionen notwendig sind, die Häufigkeit der Transaktionen, sowie ein Kostenmodell, das die Kosten der einzelnen Operationen angibt. Die Clusterung erfolgt mit dem Ziel, die *Summe der Kosten aller Transaktionen minimal* zu halten.

Clusterung von physischen Datenbanken. Die Ermittlung von Clustern in Datenbanken wird auch in Datenbank-Management-Systemen bei der *Übersetzung in physische Datenbanken* verwendet. Dabei soll nach [Glu95] eine Clusterung der Datenbanken auf physischer Ebene mit dem Ziel erfolgen, *effiziente Zugriffe* auf der Datenbank zu erreichen. Objekte, auf die nacheinander zugegriffen wird, sollen geclustert vorliegen, um die Anzahl der Seitenzugriffe zu minimieren. In [Glu95] werden verschiedene Möglichkeiten der Clusterung angegeben (Mengen, Listen, hierarchische Strukturen), die Unterteilung der Objekte in die Cluster muß manuell erfolgen.

Abgrenzung von diesen Clusterverfahren. In der hier beschriebenen Anwendung – der Semantikakquisition – werden andere Kriterien zur Clusterbildung benötigt, da *inhaltliche Zusammenhänge* bestimmt werden sollen, um verschiedene Entwurfsaufgaben sinnvoll zu untergliedern. Aufgrund dessen müssen auch andere Kriterien zur Bestimmung der Cluster herangezogen werden.
Die vertikale Partitionierung aufgrund der Transaktionen und des Kostenmodells wäre also nur eine sehr schlechte Methode zur Bestimmung von Clustern auf dem konzeptuellen Modell, da man zum einen keine vollständigen Angaben über die Transaktionen und kein Kostenmodell hat, zum anderen sucht man nach inhaltlichen Einheiten. Auf diese hat z.B. die Häufigkeit von Transaktionen keinen Einfluß.

Bei der Suche nach inhaltlichen Zusammenhängen muß man auch zulassen, daß bestimmte Komponenten der Datenbank in mehrere Cluster aufgenommen werden. Es werden also *überschneidende Cluster* gesucht, da diese eine natürlichere Repräsentation inhaltlicher Zusammenhänge sind. Die *Abstandsfunktion* muß den *inhaltlichen Zusammenhang* zwischen Attributen bzw.

Relationen-Schemata wiedergeben.

Im folgenden werden einige Veröffentlichungen angegeben, in denen ebenfalls inhaltliche Zusammenhänge gesucht und Cluster daraus abgeleitet werden.

Akoka, Comyn-Wattiau: Automatische Clusterung von semantischen Modellen. Clusteralgorithmen basieren auf einer gut gewählten Abstandsfunktion zwischen den Objekten. In dem Artikel [ACW94] werden Abstandsfunktionen von Elementen im Entity-Relationship Modell und Objektdiagrammen diskutiert. Ein neuer (automatischer) Clusteralgorithmus wird vorgestellt.

Eine optimale Aufteilung in Cluster wird erreicht, durch

- Maximieren der Summe der Abstände zwischen den Clustern und
- Minimieren der Summe der Abstände innerhalb der Cluster

Diese Eigenschaft wird in folgenden Clusteralgorithmus zugrunde gelegt:

Gegeben sind n Objekte, sowie eine Abstandfunktion, die in einer Matrix $(n * n)$ angetragen ist. Weiterhin die Anzahl der gewünschten Cluster k. *Gesucht* ist eine optimale Aufteilung der Objekte in k Cluster.

Die *Grundidee des Algorithmus* ist folgende:
Zu Beginn werden alle Objekte in das 1. Cluster gelegt. Es werden nacheinander weitere Cluster j gebildet, bis die gewünschte Anzahl von Clustern $(j = k)$ erreicht ist. Zur Bildung eines Clusters j wird das Objekt i aus den Clustern genommen, das den größten Abstand von seinem Cluster hat. Dieses wird in das Cluster j übernommen. Anschließend wird untersucht, ob es ein Objekt i gibt, das enger mit einem Cluster p zusammenhängt als mit seinem eigenen Cluster q. In diesem Fall erfolgt eine Übernahme des Objektes i aus dem Cluster q in p.

Der Algorithmus ist sehr effizient, er liegt in der Ordnung $O(k * n)$.

Voraussetzung für den Algorithmus ist eine *Abstandsfunktion* für *Entity-Relationship Modelle*. Es werden drei verschiedene Abstandsfunktionen für diese Aufgabe vorgestellt:

1. *Optischer Abstand*
   Es wird die Länge des kürzesten Pfades zwischen zwei Knoten als Abstand genommen.

2. *Hierarchischer Abstand*
   Vorhandene Kardinalitäten werden verwendet, um den Abstand zwischen zwei Elementen zu bestimmen, dabei binden 1:n- Kardinalitäten stärker als n:m-Kardinalitäten.

3. *Semantischer Abstand*
   Hierbei werden strukturelle und semantische Merkmale wie schwache Entities und binäre oder höhere Relationships in die Ermittlung des Abstandes einbezogen.

Für *objektorientierte Modelle* werden ebenfalls Abstandsmaße vorgestellt, diese basieren auf den Messages, die zwischen den Objekten ausgetauscht werden.

Die Abstandsfunktionen werden jeweils mit einem Beispiel unterlegt, dabei werden verschiedene Beispiele verwendet. Ein Vergleich der Abstandsfunktionen ist dadurch nicht möglich. Sinnvoll wäre auch eine Kombination der verschiedenen Möglichkeiten zur Abstandsberechnung.

Im Ausblick wird eine Erweiterung der Methode auf überlappende Cluster als offenes Problem genannt.

Heiland/Kruck: Bewertung und Verdichtung von Entity-Relationship Modellen. In [HeK93] wird die Verdichtung von Entity-Relationship Diagrammen, die Clusterbildung und die Zuordnung von Clustern zu Oberbegriffen beschrieben.

Es werden verschiedene Maße zur Bewertung von Entity-Relationship Diagrammen vorgestellt:

- *Maß für die Ordnungsgüte*
  Verhältnis der 1:n Kardinalitäten zu allen Kardinalitäten
- *Maß für die Komplexität*
  Anzahl der Entities und Anzahl der Relationships ohne "is-a"-Beziehungen
- *Maß für die Spezialisierung*
  Verhältnis der "is-a"-Beziehungen zu allen Beziehungen

Zur Verdichtung von Entity-Relationship Diagrammen wird folgendes Vorgehen gewählt:

1. Es werden *Opferentities* aufgrund ihres *Verknüpfungsgrades* bestimmt. Dieser ergibt sich aus der Anzahl direkter Nachbarn ohne Spezialisierungen und Randentities. Randentities sind Entities, die als Beziehungen nur Spezialisierungen haben.
   Opferentity wird das Entity mit dem geringsten Verknüpfungsgrad.
2. Zu dem Opferentity wird ein *Verdichtungsentity* bestimmt. Dieses ist ein direkter Nachbar des Opferentities mit der *maximalen Bindungsstärke*. Die Bindungsstärke ergibt sich aus Semantik der Beziehungen und Kardinalität.
3. Es erfolgt die Verknüpfung von Opfer- und Verdichtungsentity. Dazu werden verschiedene Fälle angegeben.

Dieses Verfahren wird bis zum gewünschten Verdichtungsgrad inkrementell fortgeführt.
Ein Beispiel für die Methode wurde gegeben.

Campbell: Bilden von abstrakteren Schemata für ein konzeptuelles Schema. In diesem Artikel wird das gleiche Ziel wie in [HeK93] verfolgt. Große und komplexe konzeptuelle Schemata können unübersichtlich werden. Deshalb wird in [CHP96] eine Methode beschrieben, um durch Abstraktion eine schrittweise Zusammenfassung und Verallgemeinerung gegebener konzeptueller Schemata zu erreichen.

Diese Ableitung abstrakterer Schemata basiert auf der Ermittlung von Haupt-Objekttypen (main object types) für ein Schema. Haupt-Objekttypen sind nicht formal beschreibbar, die Einschätzung der Bedeutung von Objekttypen ist teilweise sogar subjektiv. Es wird jedoch ein Merkmal zur Ermittlung dieser verwendet, das aussagekräftig und meßbar ist. Als Haupt-Objekttypen werden solche Objekttypen ermittelt, in denen die größte Anzahl verschiedener Daten auftritt. Unter Verwendung dieser Haupt-Objekttypen wird die nächsthöhere (abstraktere) Ebene gebildet. Ein Beispiel für dieses Vorgehen wurde angegeben.

Castano/De Antonellis/Fugini/Pernici: Clusterung von Datenbanken. Bei der Clusterung von Datenbanken werden sinnvolle Teildatenbanken einer gegebenen Datenbank ermittelt. Ein Verfahren dazu wurde in [CaA94], [CAF95] und [CaA96] vorgestellt. Es werden zwei Eigenschaften ausgewertet.

1. *Affinity*
   Diese liegt zwischen 0 und 1 und wird aus einem speziell angelegten Synonymwörterbuch entnommen, in dem für alle Begriffspaare ein Ähnlichkeitswert eingetragen ist.
2. *Closeness*
   Die Art und Anzahl der Verbindungen zwischen Elementen wird betrachtet.

Der Zusammenhang zwischen zwei Elementen wird als gewichtete Summe dieser beiden Eigenschaften bestimmt.

Auf diesem paarweisen Zusammenhang basierend wird die Clusterung bestimmt. Dazu wird zunächst jedes Element in ein Cluster gelegt. Es werden solange jeweils die beiden ähnlichsten Cluster vereinigt, bis die gewünschte Anzahl von Clustern erreicht ist.

Castano/De Antonellis/Fugini/Pernici: Indexierung von Datenbanken. Für die automatische Ausführung vieler Methoden ist es erforderlich, die wichtigsten Elemente in den Datenbanken zu kennen. In [CAZ92], [CaA94], [CAF95] und [CaA96] wird eine Methode vorgestellt, mit der diese automatisch aus konzeptuellen Schemata (Entity-Relationship Diagrammen) abgeleitet werden. Dabei werden Heuristiken eingesetzt, die folgende Eigenschaften ausnutzen. Es werden die Elemente

- mit den meisten Verbindungen und
- mit den meisten Eigenschaften ausgewählt.

Diese Merkmale können anhand struktureller Angaben (im Entity-Relationship Modell: Attribute und Pfade) ermittelt werden.

Durch Addition dieser Eigenschaften wird eine Bedeutung für jedes Element ermittelt. Es wird der Mittelwert über den so abgeschätzten Bedeutungen gebildet, jedes Element, dessen Wert über dem Mittelwert liegt, wird als *Schema Descriptor (SD)* bezeichnet.

Castano/De Antonellis: Clusterung um Schema Descriptoren. In [CaA94] wird vorgestellt, wie die *Clusterung* von Datenbanken mit der *Indexierung* kombiniert werden kann. Dazu werden zunächst die *Schema Descriptoren (SD)* ermittelt, diese werden jeweils in ein Cluster gelegt. Diese Cluster werden mit einem Clusterungsalgorithmus um die anderen Entities der Datenbank erweitert.

Die Ermittlung von Schemadescriptoren zur Clusterung von Datenbanken wird in diesem Kapitel aufgegriffen. Es werden in diesem Kapitel zahlreiche weitere Heuristiken für diese Aufgabe angegeben. Weiterhin werden durch zahlreiche Heuristiken überschneidende Cluster aufgestellt. Auf diesen basierend werden unabhängige Einheiten der Datenbank bestimmt.

7.3 Cluster

Im Datenbank-Entwurf ist es für viele Aufgaben nützlich, wenn man weiß, welche Teile des Datenbank-Schema inhaltlich eng zusammenhängen. *Entwurfsaufgaben* lassen sich durch diese Informationen *in sinnvolle Teilaufgaben untergliedern*. Solche inhaltlich zusammenhängenden Teile werden als *Cluster* bezeichnet.

Zur Ermittlung von inhaltlichen Clustern werden strukturelle Merkmale, Integritätsbedingungen, vorhandene Transaktionen und Informationen über den Verlauf der Entwurfes (soweit bekannt) herangezogen.
Dabei sollen überschneidende Cluster bestimmt werden, da diese inhaltliche Zusammenhänge besser wiedergeben als disjunkte Cluster. Ein Entity/Relationship kann dabei in mehreren Clustern auftreten, es kann auch Entities/Relationships geben, die in keinem Cluster vorkommen.

Cluster auf konzeptueller oder logischer Ebene können sowohl innerhalb eines Relationen-Schemas (Attribut-Cluster) als auch innerhalb eines Datenbank-Schemas (Relationen-Cluster) ermittelt werden.

7.4 Attribut-Cluster

Als Attribut-Cluster werden Attribute innerhalb eines Relationen-Schemas bezeichnet, zwischen denen ein enger inhaltlicher Zusammenhang besteht.

7.4.1 Heuristiken zur Suche nach Attribut-Clustern

Voraussetzung für die Ermittlung korrekter Attribut-Cluster sind vollständige Informationen über die Integritätsbedingungen einer Datenbank. Inhaltlich zusammenhängende Teile können nicht exakt bestimmt werden, wenn keine vollständigen Informationen über die Integritätsbedingungen bekannt sind. Vollständige Informationen über die Semantik sind jedoch während der Semantikakquisition nicht bekannt, sie sollen durch das Mittel Attribut-Cluster gezielter ermittelt werden.

Um diese wechselseitigen Abhängigkeiten zu durchbrechen, werden die *Attribut-Cluster* durch verschiedene Heuristikregeln *abgeschätzt*. Strukturelle und (unvollständige) semantische Informationen, sowie bekannte Transaktionen können zu einer Abschätzung, welche Attribute eng zusammenhängen, herangezogen werden. Auch der zeitliche Ablauf des Entwurfsprozesses kann Hinweise auf zusammenhängende Teile liefern. Im folgenden werden alle auswertbaren Merkmale zusammengetragen, die Hinweise auf Attribute mit engem inhaltlichen Zusammenhang liefern.

Auswertung von Integritätsbedingungen

Die Clusterbildung wird zur Unterstützung der Semantikakquisition eingesetzt, man kann also nicht davon ausgehen, daß vollständige Informationen über Integritätsbedingungen bekannt sind. Es können jedoch bereits einige Integritätsbedingungen bekannt sein. Diese lassen sich bei der Clusterbildung auswerten.

$H_{AC}1$. Gibt es *funktionale Abhängigkeiten* mit einer *Attributmenge von mehr als einem Attribut auf der linken Seite*, so stellt die Attributmenge der linken Seite eine inhaltliche Einheit dar.
Beispiel: Aus einer geltenden funktionalen Abhängigkeit

$$AB \longrightarrow C$$

läßt sich ableiten, daß die Attribute AB ein Cluster bilden.

$H_{AC}2$ Funktionale Abhängigkeiten geben Zusammenhänge zwischen den Attribute der linken und rechten Seite an. Diese Attribute stehen in inhaltlichem Zusammenhang, bilden also ein Attribut-Cluster.
Beispiel: Ist die funktionale Abhängigkeit

$$\mathrm{AB} \longrightarrow \mathrm{C}$$

bekannt, so kann man aufgrund dessen ableiten, daß die Attribute ABC ein Attribut-Cluster bilden.

Auswertung des Entwurfsprozesses

Kann der Entwurfsprozeß beobachtet werden oder läßt sich der Entwurfsprozeß nachvollziehen, so kann man daraus ableiten, welche Attribute in Zusammenhang stehen.

$H_{AC}3$. Attribute, die in *zeitlichem Zusammenhang* eingegeben wurden, gehören oft auch inhaltlich zusammen.

$H_{AC}4$ Gibt es *natürlichsprachige Beschreibungen* zur Datenbank, so kann man annehmen, daß die Attribute, die in einem Satz gemeinsam genannt werden, oft in einem engen Zusammenhang stehen. Das gilt besonders für Attribute, die in der natürlichen Sprache in *Aufzählungen* gemeinsam genannt werden.

Auswertung von strukturellen Merkmalen

Aus den strukturellen Informationen der Datenbank lassen sich Rückschlüsse über zusammengehörige Attribute ziehen.

$H_{AC}5$. Aus der *Reihenfolge der Attribute in einem Relationen-Schema* lassen sich Schlußfolgerungen über die zusammengehörigen Attribute ziehen. Stehen Attribute nebeneinander in dem Relationen-Schema, so besteht zwischen diesen Attributen mit größerer Wahrscheinlichkeit ein Zusammenhang als zwischen Attributen, die in dem Relationen-Schema weit auseinander liegen. Diese Heuristik basiert ähnlich wie die Heuristik $H_{AC}3$ auf der Überlegung, daß der Datenbank-Entwerfer Attribute, die zusammengehören, auch zusammen eingibt.

$H_{AC}6$ Treten in einem Relationen-Schema Attribute mit *ähnlichen Attributnamen* auf, d.h. Attribute mit gleichen Teilstrings in den Attributnamen, so deutet das auf einen Zusammenhang dieser Attribute und damit auf ein Cluster hin.

$H_{AC}7$ Durch *Verschachtelung von Attributen*, die in NF^2- Datenmodellen unterstützt wird, *verbindet* ein Entwerfer *atomare Attribute* in dem Relationen-Schema. Man kann die Verschachtelung von Attributen als strukturelle Heuristik für den Zusammenhang von Attributen werten. Die so zusammengefaßten Attribute werden als Cluster betrachtet.

Explizite Angaben des Benutzers

$H_{AC}8$ Man kann einen Benutzer *explizit nach zusammengehörigen Attributen* fragen. Dabei ist es sinnvoll, Mehrfachnennungen eines Attributes in verschiedenen Clustern zuzulassen, da überschneidende Attribut-Cluster gesucht werden.

Transaktionen

$H_{AC}9$ In dieser Heuristik soll der Zusammenhang von Attributen *aufgrund von bekannten Transaktionen* bestimmt werden.
Diese Idee liegt auch der vertikalen Partitionierung bei verteilten Datenbanken zugrunde, dort erfolgt eine vertikale Aufteilung von Attributen aufgrund der Transaktionen und eines Kostenmodells.
In dieser Anwendung muß die Auswertung von Transaktionen in anderer Form erfolgen, da man hier nach inhaltlichen Zusammenhängen auf konzeptioneller Ebene sucht und nicht effiziente Anfragen auf der Datenbank erreichen will. Deshalb wird die Transaktionshäufigkeit nicht berücksichtigt, es ist auch nicht sinnvoll, ein Kostenmodell heranzuziehen. Man kann also durch Auswertung von Transaktionen nur qualitative Aussagen und keine quantitativen über inhaltliche Zusammenhänge ableiten.

Zur Ermittlung der inhaltlich zusammenhängenden Attribute wird eine Matrix gebildet, in der die Transaktionen als Zeilen und die Attribute als Spalten auftreten. In der Matrix wird an der Stelle M_{ki} eingetragen, ob die Transaktion T_k das Attribut A_i umfaßt:

	A_1	A_2	...	A_n
T_1	0	0	...	1
T_2	1	1	...	1
...	...	...	...	...
T_m	1	0	...	0

Aus dieser Matrix wird durch eine Abstandsfunktion der Abstand zwischen zwei Attributen A_i und A_j ermittelt:

$$\text{Abstand}(A_iA_j) := \sum_{k=1}^{m} |M_{ki} - M_{kj}|$$

Die Attribute stehen durch die Transaktionen in engem inhaltlichen Zusammenhang, wenn der auf diese Weise ermittelte Abstand möglichst klein ist.

Nicht in jedem Fall sind alle in den Heuristikregeln zur Ermittlung von Attribut-Clustern ausgewerteten Informationen vorhanden, sodaß in der Regel nur ein Teil der Heuristiken angewendet werden kann.

7.4.2 Wichtung der Heuristiken

Die anwendbaren Heuristiken müssen *gewichtet* werden, um nicht nur zusammenhängende Einheiten zu ermitteln, sondern auch eine Wertung zu haben, wie eng der inhaltliche Zusammenhang innerhalb dieser Einheiten zu sein scheint. Dabei sollen besonders aussagekräftige

Heuristiken höher gewichtet werden als die anderen Heuristikregeln. Die folgende Übersicht zeigt, wie zuverlässig die Ergebnisse der einzelnen Heuristiken für die meisten Anwendungen sind:

besonders aussagekräftig	$H_{AC}1$, $H_{AC}2$, $H_{AC}7$, $H_{AC}8$
aussagekräftig	$H_{AC}4$, $H_{AC}6$, $H_{AC}9$
nur in Zusammenhang mit anderen Heuristiken sinnvoll	$H_{AC}3$, $H_{AC}5$

Aus dieser Übersicht und der Information, welche Heuristikregeln anwendbar sind, ergeben sich die Gewichte für die einzelnen Heuristikregeln.

Man kann also durch *Auswertung aller einzelnen Heuristikregeln* eine *Menge von Attribut-Clustern* C, die jeweils durch eine Heuristikregeln gefunden werden, ermitteln. In der Menge C können *mehrfach gleiche Attributmengen* oder überschneidende Attributmengen auftreten. Die Heuristikregeln sollen *gegeneinander gewichtet* werden. Bei der Ermittlung der Gesamtplausibilitäten von Clustern $c_i, c_j \in C$ mit den Wichtungen $Pl(c_i)$ und $Pl(c_j)$ sind folgende Schritte auszuführen:

1. *Eliminieren von gleichen Attributmengen und Anpassen der Plausibilitäten:*
   Es werden alle identischen Cluster $c_1, c_2, .., c_n$ ermittelt, die Plausibilität von c_1 wird wie folgt verändert:
   $Pl(c_1) := Pl(c_1) + Pl(c_2) + .. + Pl(c_n)$

   Die Cluster $c_2..c_n$ werden anschließend gelöscht.

2. *Auswertung von Teilmengenbeziehungen:*
   Für jedes Cluster c_i wird untersucht, ob Cluster c_j mit der Eigenschaft $c_i \subset c_j$ existieren. Ist das der Fall, so wird die Plausibilität von c_i wie folgt verändert:
   $Pl(c_i) := Pl(c_i) + Pl(c_j) * \frac{|c_i|}{|c_j|}$

3. *Auswerten gemeinsamer Attributsequenzen:*
   Für alle Paare von Clustern c_i, c_j mit $c_i \neq c_j$ wird untersucht, ob es eine Attributmenge m mit $m \subset c_i, m \subset c_j$ und $|m| > 1$ gibt.

   Ist das der Fall, so wird die Attributmenge m als eigenes Cluster c_k abgespeichert und die Plausibilität $Pl(c_k)$ bestimmt.

   1. Fall: c_k ist bereits als Cluster in C vorhanden
      $Pl(c_k) := Pl(c_k) + Pl(c_i) * \frac{|c_k|}{|c_i|} + Pl(c_j) * \frac{|c_k|}{|c_j|}$

   2. Fall: c_k ist noch nicht als Cluster in C vorhanden
      Aufnahme von c_k in C
      $Pl(c_k) := Pl(c_i) * \frac{|c_k|}{|c_i|} + Pl(c_j) * \frac{|c_k|}{|c_j|}$

Durch die vorgestellten Heuristiken erhält man eine Menge von Attribut-Clustern, sowie eine Wichtung, die abschätzt, wie plausibel diese ermittelten Cluster nach Auswertung aller angegebenen Heuristikregeln sind. Die Berechnung der Gesamtplausibilitäten ist eine Abschätzung und keine exakte Berechnung, es können dadurch auch Plausibilitäten auftreten, die größer als 1 sind, diese werden anschließend auf 1 gesetzt.

Das Vorgehen bei der Ermittlung von Attributclustern und deren Wichtung soll im folgenden Abschnitt anhand eines Beispiels erläutert werden.

7.4.3 Beispiel für die Ermittlung von Attribut-Clustern

Gegeben sei folgendes Relationen-Schema:

Artikelnr	Einkaufspreis	Verkaufspreis	Gewinnspanne	Mehrwertsteuer	Rabatt

Gesucht werden Attribut-Cluster des Relationen-Schemas. Dazu sollen folgende funktionale Abhängigkeiten bekannt sein:

Einkaufspreis Verkaufspreis $\longrightarrow$ Gewinnspanne
Verkaufspreis $\longrightarrow$ Mehrwertsteuer
Verkaufspreis $\longrightarrow$ Rabatt

Durch die Heuristiken $H_{AC}1, H_{ACl}2$ und $H_{AC}6$, die die funktionalen Abhängigkeiten und die Attributbezeichnungen auswerten, werden Hinweise auf Attribut-Cluster abgeleitet:

Heuristikregel	gefundenes Attribut-Cluster	Wichtung
$H_{AC}1$	Einkaufspreis Verkaufspreis Gewinnspanne	0.4
	Verkaufspreis Mehrwertsteuer	0.4
	Verkaufspreis Rabatt	0.4
$H_{AC}2$	Einkaufspreis Verkaufspreis	0.4
$H_{AC}6$	Einkaufspreis Verkaufspreis	0.2

Diese ergeben folgende Gesamtwichtung:

gefundenes Cluster	Gesamtwichtung
Einkaufspreis Verkaufspreis	0.866
Einkaufspreis Verkaufspreis Gewinnspanne	0.4
Verkaufspreis Mehrwertsteuer	0.4
Verkaufspreis Rabatt	0.4

7.4.4 Verwendung von Attribut-Clustern

Die ermittelten Cluster lassen an vielen Stellen beim Entwurf oder Wiederentwurf von Datenbanken verwenden.

Sie können *Entwurfsaufgaben sinnvoll unterteilen* und die Reihenfolge der Entwurfsaufgaben für den Benutzer nachvollziehbar machen. Attribut-Cluster sind besonders bei Relationen-Schemata mit vielen Attributen hilfreich.

Bei der *Semantikakquisition* werden Integritätsbedingungen innerhalb von Attribut-Clustern immer im Zusammenhang diskutiert, sodaß der Dialog für einen Benutzer nicht sprunghaft erscheint. Die Attribut-Cluster werden also verwendet, um die *Erfragungsreihenfolge* festzulegen. Attribut-Cluster können auch verwendet werden, um Kandidaten für Integritätsbedingungen zu

bestimmen. Wurde ein Attribut-Cluster ermittelt, so wird für die Attribute des Clusters untersucht, ob zwischen diesen bestimmte Integritätsbedingungen gelten. So wird für ein ermitteltes Attribut-Cluster überprüft, ob alle Attribute des Clusters einen *Schlüssel* bilden. *Funktionale Abhängigkeiten* werden eher innerhalb der Cluster vermutet als zwischen Attributen, die nicht gemeinsam zu einem Cluster gehören.

Inhaltliche Einheiten sind *Voraussetzung* zur Bestimmung von *voneinander unabhängigen Einheiten*, diese werden im Abschnitt 7.6 beschrieben.

7.5 Relationen-Cluster

Im vorherigen Abschnitt wurde gezeigt, wie Attribut-Cluster als sinnvolle inhaltliche Einheiten innerhalb eines Relationen-Schemas ermittelt werden. Man möchte für viele Entwurfsaufgaben ebenfalls eine sinnvolle Unterteilung des Datenbank-Schemas erreichen. Dazu werden *inhaltlich zusammengehörige Relationen-Schemata* in sogenannte *Relationen-Cluster* zusammengefaßt. Dieses sind überschneidende Cluster, Relationen-Schemata können in mehreren Clustern auftreten, es kann auch Relationen-Schemata geben, die in keinem Cluster auftreten. Auf diese Weise werden inhaltliche Zusammenhänge der Relationen-Schemata in einem Datenbank-Schema am besten repräsentiert.

In diesem Abschnitt wird gezeigt, wie Relationen-Cluster gefunden werden können.

7.5.1 Suche nach Relationen-Clustern

Ähnlich wie für Attribut-Cluster können zur Suche nach Relationen-Clustern viele verschiedene Informationen ausgewertet werden:

Auswertung von semantischen Merkmalen

$H_{RC}1$ Ist ein Datenbank-Entwurf im *konzeptionellen Modell* bekannt, so können *vorhandene Pfade* im Entwurf Hinweise auf Einheiten in der Datenbank geben. Knoten, zwischen denen direkte Verbindungen über Pfade existieren, haben einen stärkeren Zusammenhang als Knoten ohne Verbindung oder nur mit einer Verbindung über lange Wege.

Dabei geben bekannte Kardinalitäten die Bindungen zwischen den Knoten an, eine intuitiv nachvollziehbare Möglichkeit dazu wurde in [HeK93] vorgeschlagen. Dort werden für Kardinalitäten folgende Bindungsstärken angegeben ($\mathsf{card}(R,E) = (1,1)$ - 3, $\mathsf{card}(R,E) = (0,1)$ - 2, $\mathsf{card}(R,E) = (1,n)$ - 1, $\mathsf{card}(R,E) = (0,n)$ - 0). In [ACW94] werden diese ähnlich, aber ohne konkrete Bewertung, festgelegt. Diese empirisch erstellte Bewertung kann verwendet werden, um den Zusammenhang zwischen Entities und Relationships zu bestimmen.

$H_{RC}2$ Kennt man Datenbanken nur im *logischen Modell*, so können zur Ermittlung von Relationen-Clustern keine Pfadinformationen verwendet werden, man kann die gleichen Informationen aus vorhandenen Integritätsbedingungen ableiten. Existieren zwischen Teilen eines Datenbank-Schemas geltende *funktionale Abhängigkeiten* oder *Inklusionsabhängigkeiten*, so deutet das auf ein Cluster in dem Datenbank-Schema hin.

Beobachtung des Entwurfsprozesses

$H_{RC}3$ Durch Auswertung des Entwurfsprozesses kann man ableiten, welche *Entities/Relationships* in Zusammenhang stehen. Werden Knoten *gemeinsam eingegeben*, so gehören diese oft auch inhaltlich zusammen.
Besonders, wenn der Entwurfsprozeß durch Anwendung einzelner Entwurfsschritte ausgeführt wurde (vorgestellt in [BCN92], erweitert und systematisiert in [Ape94]), kann man ableiten, welche Knoten zusammengehören. Knoten, die bei der Verfeinerung eines Entities/Relationships gemeinsam ergänzt wurden, gehören in der Regel inhaltlich eng zusammen.

$H_{RC}4$ Gibt es natürlichsprachige Beschreibungen der Datenbank, so kann man daraus eng zusammengehörige Teile ableiten. Existieren in den Beschreibungen Relationen-Schemata, deren Bezeichnungen in einem Satz gemeinsam auftreten, so weist das auf einen Zusammenhang dieser Relationen-Schemata hin. Gibt es sogar Relationen-Schemata, die in Aufzählungen durch Konnektoren verbunden sind, so gehören diese Relationen-Schemata in der Regel inhaltlich sehr eng zusammen.

Auswertung von strukturellen Merkmalen

$H_{RC}5$ Auch die *graphische Nähe* von *Knoten* in einem Entity-Relationship Diagramm kann Hinweise auf zusammengehörige Knoten geben. Dieses Merkmal ist nur auswertbar, wenn das graphische Layout nicht automatisch vorgenommen wurde, sondern durch den Entwerfer explizit erfolgte. Im Falle einer automatischen Plazierung der Knoten sind die Ergebnisse dieser Heuristik ähnlich der Heuristik $H_{RC}1$, da die automatische Plazierung aufgrund der vorhandenen Pfadinformationen erfolgt.

$H_{RC}6$ *Gleiche Teilstrings* in *Relationennamen* können ebenfalls ein Hinweis auf eine ähnliche Bedeutung von Relationen-Schemata im Entwurf sein, sodaß zwischen diesen Relationen-Schemata ein inhaltlicher Zusammenhang besteht.

Explizite Angaben des Benutzers

$H_{RC}7$ Eine *explizite Zusammenfassung* von Relationen-Schemata einer Datenbank *durch den Benutzer* ist eine Aufgabe, die für viele Benutzer wesentlich leichter ist als die Spezifikation von Integritätsbedingungen. Es ist dabei sinnvoll, Mehrfachnennungen eines Relationen-Schemas in verschiedenen Clustern zu unterstützen.

Auswertung von Transaktionen

$H_{RC}8$ Durch *Auswertung von Transaktionen* kann man Relationen-Schemata finden, auf die durch viele Transaktionen gemeinsam zugegriffen wird.
Diese Methode wurde in analoger Form zur Suche nach Attribut-Clustern verwendet. Man stellt hier eine Matrix auf, in der zu allen Transaktionen eingetragen wird, auf welche Relationen-Schemata sie zugreifen.

	R_1	R_2	...	R_n
T_1	1	1	...	0
T_2	1	0	...	0
...	...	...	...	...
T_m	0	1	...	1

Aus dieser Matrix werden wie in der Heuristikregel $H_{AC}9$ durch eine Abstandsfunktion ähnliche Spalten ermittelt. Relationen-Schemata mit geringem Abstand in dieser Übersicht stehen durch die Transaktionen in engem inhaltlichen Zusammenhang.

Diese Heuristiken sind in der Regel nicht vollständig anwendbar, da nicht alle ausgewerteten Informationen in jedem Fall vorhanden sind. Einige Heuristiken lassen sich nur auswerten, sofern ein konzeptionelles Modell vorhanden ist, andere nur, wenn Transaktionen bekannt sind, mit der Datenbank also bereits gearbeitet wurde oder die Transaktionen bereits zum Entwurf spezifiziert wurden.

7.5.2 Wichtung von Relationen-Clustern

Die Heuristiken müssen gewichtet werden, um nicht nur Cluster zu ermitteln, sondern auch eine Wertung zu haben, wie eng der inhaltliche Zusammenhang innerhalb dieser Cluster zu sein scheint. Dabei soll die Summe der Gewichte 1 sein, in Abhängigkeit von der Menge der auswertbaren Heuristiken werden die einzelnen Gewichte festgelegt. Die folgende Übersicht zeigt, wie zuverlässig die Ergebnisse der einzelnen Heuristiken sind:

besonders aussagekräftig	$H_{RC}2$, $H_{RC}7$
aussagekräftig	$H_{RC}1$, $H_{RC}4$, $H_{RC}6$, $H_{RC}8$
nur in Zusammenhang mit anderen Heuristiken sinnvoll	$H_{RC}3$, $H_{RC}5$

Mit diesen Methoden kann man eine Menge von Relationen-Clustern in einer Datenbank und eine Wichtung zu diesen ableiten. Die Ermittlung eines Gesamtplausibilitätsfaktors zu den ermittelten Clustern erfolgt analog der Wichtung von Attribut-Clustern (siehe Abschnitt 7.4.1).

7.5.3 Verwendung von Relationen-Clustern

Die ermittelten Clusters lassen sich ebenfalls an vielen Stellen beim Entwurf oder Wiederentwurf von Datenbanken verwenden. Alle *Entwurfsaufgaben* lassen sich *sinnvoll in Teilaufgaben untergliedern*, wenn man inhaltlich zusammenhängende Teile gefunden hat. Viele Entwurfsaufgaben können für die einzelnen Cluster getrennt ausgeführt werden.

Bei der *Semantikakquisition* können Cluster verwendet werden, um eine sinnvolle Gestaltung der Dialoge zu erreichen. Die *Dialogreihenfolge* kann dadurch so festgelegt werden, daß der Dialog für einen Benutzer nachvollziehbar ist, dazu werden alle Fragen zu den Integritätsbedingungen innerhalb von Relationen-Clustern im Zusammenhang geklärt. Dadurch wird die Sprunghaftigkeit der Fragen verhindert.

Die *Suche nach Analoga*, die zur Ermittlung von Inklusions- und Exklusionsabhängigkeiten, sowie Kardinalitäten notwendig sind, erfolgt in großen Datenbank-Schemata gezielt innerhalb der Relationen-Cluster.

Relationen-Cluster werden ebenfalls eingesetzt, um eine Wiederverwendung von Datenbanken zu unterstützen. Dabei werden inhaltliche Einheiten in einer Datenbank ermittelt, um die Wiederverwendung von Datenbank-Schemata gezielt anwenden zu können. Das Vorgehen wurde ausführlich in Kapitel 6 beschrieben.

Ermittelte Cluster sind auch *Voraussetzung* zur Bestimmung von *voneinander unabhängigen Einheiten*, wie diese gefunden werden können und wozu sie verwendet werden können, wird im nächsten Abschnitt beschrieben.

7.6 Unabhängige Einheiten

In großen Datenbank-Schemata ist es nicht möglich, alle konstruierbaren Integritätsbedingungen mit dem Benutzer zu diskutieren. Man muß deshalb den Suchraum der zu erfragenden Integritätsbedingungen einschränken. Das soll möglichst sinnvoll passieren, d.h. es sollen möglichst wenige geltende Integritätsbedingungen dadurch ausgeschlossen werden. Man versucht deshalb, Einheiten zu finden, die voneinander unabhängig sind, d.h. zwischen denen keine geltenden Integritätsbedingungen existieren. Findet man solche Einheiten, so läßt sich die Semantikakquisition hier einschränken.

7.6.1 Bestimmung unabhängiger Einheiten

Unabhängige Attribut-Einheiten sind Attributmengen, zwischen denen keine funktionalen Abhängigkeiten gelten.

Definition 7.1 *Sei R ein Relationen-Schema, U die Attributmenge von R, X und Y seien Teilmengen von U. Die Attributmengen X und Y sind* **unabhängige Attributmengen** *von R, wenn folgende Bedingung gilt:*
$\nexists\ X' \subseteq X$ und $Y' \subseteq Y$ mit $X' \longrightarrow Y'$ oder $Y' \longrightarrow X'$.

Unabhängige Relationen-Einheiten sind Mengen von Relationen-Schemata zwischen denen keine funktionalen Abhängigkeiten oder Inklusionsabhängigkeiten gelten.

Definition 7.2 *Gibt es innerhalb eines Datenbank-Schemas D zwei Mengen von Relationen-Schemata R und S, so heißen diese* **unabhängige Relationen-Schemata** *von D, wenn folgende Bedingung gilt:*
$\nexists\ X$ - Attributmenge in R und Y - Attributmenge in S, $X' \subseteq X$, $Y' \subseteq Y$, zwischen denen funktionale Abhängigkeiten ($X' \longrightarrow Y'$ oder $Y' \longrightarrow X'$) oder Inklusionsabhängigkeiten ($X' \subseteq Y'$ oder $Y' \subseteq X'$) gelten.

Nach Definition ergibt sich, daß man solche unabhängigen Einheiten nur dann finden kann, wenn man *vollständige Informationen* über die *funktionalen Abhängigkeiten und Inklusionsabhängigkeiten* der Relationen- und Datenbank-Schemata verfügt. Da man diese während der Semantikakquisition nicht kennt, ist diese Voraussetzung nicht gegeben. Man kann aber versuchen abzuschätzen, welche Einheiten voneinander unabhängig sind. Folgendes Vorgehen wird dazu gewählt:

Mit den Heuristikregeln zur Ermittlung von Clustern werden *Attribut-Cluster* oder *Relationen-Cluster* sowie eine *Wichtung*, wie plausibel die ermittelten Cluster sind, gefunden. Diese Information soll verwendet werden, um *voneinander unabhängige Einheiten* zu ermitteln. Dabei wird nach folgender Methode vorgegangen:

1. Die ermittelten *Attribut-Cluster bzw. Relationen-Cluster* werden entsprechend der ermittelten Plausibilität *sortiert*.

2. Zwischen den ermittelten Clustern erfolgt eine paarweise *Untersuchung auf Unabhängigkeit aufgrund der bereits bekannten Integritätsbedingungen*, d.h. es wird für jeweils zwei Cluster C_1 und C_2 getestet, ob deren Unabhängigkeit bereits durch die bekannten Integritätsbedingungen verletzt ist. Dabei wird mit den Clustern mit der größten Plausibilität begonnen.

   Es werden folgende Untersuchungen durchgeführt:

   (a) Für zwei Attribut-Cluster wird untersucht, ob zwischen den Attributen des Clusters keine bereits bekannten funktionalen Abhängigkeiten gelten:
   ($\nexists\, X' \subseteq C_1$ und $Y' \subseteq C_2$ mit $X' \longrightarrow Y', Y' \longrightarrow X'$).

   (b) Für zwei Relationen-Cluster wird untersucht, ob zwischen den Relationen-Schemata keine bereits bekannten funktionalen Abhängigkeiten und Inklusionsabhängigkeiten gelten:
   $\nexists\, R_1 \in C_1$ mit X - Attribute in R_1 und $\nexists\, R_2 \in C_2$ mit Y - Attribute in R_2, $X' \subseteq X$, $Y' \subseteq Y$, für die eine funktionale Abhängigkeit $X' \longrightarrow Y'$, $Y' \longrightarrow X'$ oder eine Inklusionsabhängigkeit $X' \subseteq Y'$, $Y' \subseteq X'$ gilt.

3. Ist durch die bereits vorhandenen Integritätsbedingungen die Unabhängigkeit der Cluster nicht verletzt, so erfolgt eine weitere *stichprobenweise Untersuchung*, ob die *Cluster voneinander unabhängig* sind.
   Dazu werden einige besonders wichtige mögliche Integritätsbedingungen zwischen den Einheiten herangezogen.

   (a) für zwei Attribut-Cluster
   Die unären funktionalen Abhängigkeiten zwischen zwei Attribut-Clustern C_1 und C_2 werden untersucht und mit dem Benutzer diskutiert, sofern sie noch nicht ableitbar waren und bereits im Algorithmusschritt 2 ausgewertet wurden.

   (b) für zwei Relationen-Cluster
   Zu den Schlüsseln der Relationen-Schemata innerhalb von C_1 werden mögliche Fremdschlüssel in C_2 gesucht. Über vorhandenen Fremdschlüsseln werden Inklusionsabhängigkeiten erfragt.

   Wenn diese Stichproben auf Unabhängigkeit deuten (also keine funktionalen Abhängigkeiten bei Attribut-Clustern bzw. keine funktionalen Abhängigkeiten und keine Inklusionsabhängigkeiten bei Relationen-Clustern gelten), so wird angenommen, daß die ermittelten Cluster voneinander unabhängig sind.

7.6.2 Verwendung unabhängiger Einheiten

Die aus den Attribut- und Relationen-Clustern abgeleiteten unabhängigen Einheiten werden verwendet, um sie zur *sinnvollen Beschränkung des Suchraumes* für die Akquisition von Integritätsbedingungen einzusetzen. Ist keine vollständige Erfragung der unbekannten Integritätsbedingungen aufgrund der Größe der Datenbank möglich, so möchte man die Menge der untersuchten Integritätsbedingungen so einschränken, daß möglichst wenige geltende Integritätsbedingungen von der Untersuchung ausgeschlossen werden.

Werden voneinander unabhängige Einheiten abgeschätzt, so erfolgt keine weitere Suche nach geltenden Integritätsbedingungen zwischen diesen Einheiten oder die Suche nach solchen Integritätsbedingungen erfolgt erst nach der Erfragung aller anderen unbekannten Integritätsbedingungen des Datenbank-Schemas. Man kann also die Abschätzung der voneinander unabhängigen Einheiten verwenden, um den *Suchraum* zur Semantikakquisition *einzuschränken* bzw. zu *sortieren*. Dabei werden die unabhängigen Attribut-Cluster verwendet, um die *Anzahl der zu untersuchenden funktionalen Abhängigkeiten* einzuschränken. Mit Hilfe der ermittelten unabhängigen Relationen-Cluster läßt sich die *Menge der zu untersuchenden Inklusionsabhängigkeiten* reduzieren. Funktionale Abhängigkeiten zwischen verschiedenen Relationen-Schemata werden in der vorliegenden Arbeit nicht untersucht, diese könnten mit der gleichen Methode eingeschränkt werden.

7.7 Kernrelationen

In jedem Datenbank-Schema gibt es besonders wichtige und weniger wichtige Relationen-Schemata. Die wichtigsten Relationen-Schemata sollen im folgenden als *Kernrelationen* bezeichnet werden. Intuitiv ist dieser Begriff klar, diese Erläuterung ist aber nicht exakt, es läßt sich keine formale Definition für die Bedeutung eines Relationen-Schemas innerhalb eines Datenbank-Schemas geben.

7.7.1 Suche nach Kernrelationen

Es gibt einige Hinweise, die ausgewertet werden können, um Kernrelation sowohl im Entity-Relationship Modell als auch im relationalen Modell zu ermitteln:

H_K1 Die Knoten im Entity-Relationship Modell, zu denen die meisten Pfade führen und von denen die meisten Pfade wegführen, sind wahrscheinlich die Knoten mit der größten Bedeutung, also die Kernrelationen der Datenbank.

H_K2 Entities oder Relationships bzw. Relationen-Schemata mit vielen Attributen enthalten mehr Informationen als Knoten mit wenigen Attributen, haben deshalb meist eine größere Bedeutung in der Datenbank.

H_K3 Entities sind eher Kernrelationen als Relationships.

H_K4 Kennt man eine Datenbank nur im relationalen Modell, so kann man keine Pfadbeziehungen auswerten. Man kann die gleiche Eigenschaft aus der bekannten Semantik ableiten. Die Relationen-Schemata, die bei vielen Inklusionsabhängigkeiten auf der rechten Seite auftreten, sind die Kernrelationen der Datenbank.

H_K5 Sind natürlichsprachige Beschreibungen einer Datenbank bekannt, so kann man ableiten, daß die Relationen-Schemata Kernrelationen der Datenbank sind, deren Bezeichnungen am häufigsten vom Datenbank-Entwerfer verwendet werden.

H_K6 Die Aussagen, mit denen die natürlichsprachigen Beschreibungen über die Datenbank begonnen werden, kennzeichnen oft die wichtigsten Objekte des Datenbank-Schemas.

H_K7 Die gleichen Merkmale kann man auch ohne Existenz natürlichsprachiger Beschreibungen auswerten. Ist die zeitliche Entwurfsreihenfolge eines Datenbank-Schemas bekannt, so kann man ableiten, daß die Relationen-Schemata, mit denen der Datenbank-Entwurf begonnen wurde, wichtiger sind als die Relationen-Schemata, die später ergänzt wurden.

H_K8 Sind top-down-Entwürfe nachvollziehbar, so kann angenommen werden, daß die Entities oder Relationships, mit denen der Entwurf begonnen wurde und die anschließend weiter untergliedert werden, die Kernrelationen des Datenbank-Schemas kennzeichnen. Finden sich diese Bezeichnungen in dem detaillierten Datenbank-Schema wieder, so kann man sagen, daß die zugehörigen Relationen-Schemata die Kernrelationen sind.

H_K9 Sind Transaktionen bekannt, so kann man ableiten, daß die Relationen-Schemata die größte Bedeutung in dem Datenbank-Schema haben, über denen die meisten Transaktionen durchgeführt werden.

7.7.2 Wichtung der Heuristiken zur Bestimmung von Kernrelationen

Einige der angegebenen Heuristiken zur Bestimmung von Kernrelationen sind plausibler als andere. Die folgende Übersicht gibt an, wie aussagekräftig die einzelnen Heuristiken sind:

besonders aussagekräftig	H_K1, H_K4, H_K8
aussagekräftig	H_K3, H_K5, H_K6, H_K7, H_K9
nur in Zusammenhang mit anderen Heuristiken sinnvoll	H_K2

Entsprechend dieser Übersicht werden die Gewichte der Heuristikregeln festgelegt, wobei aussagekräftigere Heuristiken höher gewichtet werden als die anderen Heuristiken.
Die folgende Formel gibt an, wie diese Heuristiken in eine Abschätzung der Plausibilitäten von Kernrelationen eingehen können:

$$Pl(R \text{ ist Kernrelation}) := \sum_{i=1}^{9} w_i * H_Ki(R)$$

$H_Ki(A) \in [0..1]$ liefert das Ergebnis der Heuristikregel H_Ki
für das Relationen-Schema R
$w_i \in [0,1]$ und $w_1 + .. + w_9 = 1$

Durch diese Abschätzung erhält man für jedes Relationen-Schema eine Abschätzung im Bereich 0..1, die angibt, wie groß die Bedeutung des Relationen-Schemas innerhalb des Datenbank-Schemas ist.

7.7.3 Verwendung von Kernrelationen

Die ermittelten Kernrelationen werden ebenso wie die Cluster und die unabhängigen Einheiten für die *Festlegung der Dialogreihenfolge* bei der Semantikakquisition verwendet. Bei der Erfragung von Kandidaten für Inklusions- und Exklusionsabhängigkeiten und Feststellung der Kardinalitäten wird mit diesen Kernrelationen begonnen.

Gibt es in dem Datenbank-Schema Teile ohne Verbindung zu anderen Teilen und man versucht, Analoga oder darauf basierend Inklusionsabhängigkeiten und Fremdschlüsselbeziehungen zu finden, so wird zuerst untersucht, ob diese zu den Kernrelationen bestehen.
Die Kernrelationen sind also auch ein Mittel zur *Festlegung der Suchreihenfolge* nach Analoga.

Kernrelationen können verwendet werden, um automatisch zu erkennen, welche *Inhalte* ein Datenbank-Schema hat, man kann diese heranziehen, um automatisch *Stichwörter* zu einem Datenbank-Schema zu generieren. Dieses kann bei der Wiederverwendung von Datenbanken verwendet werden, um gezielter Datenbankteile zu ermitteln, die wiederverwendet werden können.

Bei der in Kapitel 6 vorgestellten Wiederverwendung erfolgt eine Untergliederung eines Datenbank-Schemas in sinnvolle Teile, dazu werden Kernrelationen ermittelt, um die Hauptinhalte herauszufinden, um diese Kernrelationen erfolgt eine Clusterbildung. Dabei wird die Ermittlung von Relationen-Clustern eingesetzt.

Kapitel 8

Bestimmung der Erfragungsreihenfolge von Integritätsbedingungen

In den vorherigen Kapiteln wurde gezeigt, wie plausible Kandidaten für Integritätsbedingungen aus verschiedenen Angaben abgeleitet werden können. Für die noch unbekannten Integritätsbedingungen wurde dazu eine Abschätzung vorgestellt, die feststellt, wie wahrscheinlich diese unbekannten Integritätsbedingungen geltende Integritätsbedingungen sind. Die Anzahl der unbekannten Integritätsbedingungen kann sehr groß sein und damit kann auch die *Anzahl der plausiblen Kandidaten für Integritätsbedingungen zu groß* sein, um diese der Reihe nach zu erfragen. Jede zu erfragende unbekannte Integritätsbedingung führt zu einem Dialogschritt mit dem Benutzer, es gilt also, die Anzahl der Dialogschritte zu reduzieren, um einen sehr langwierigen Dialog mit dem Benutzer zu vermeiden. Deshalb muß eine *effiziente Erfragungsreihenfolge* gefunden werden. Weiterhin muß der Dialog zur Erfragung unbekannter Integritätsbedingungen für den Benutzer *nachvollziehbar* sein, um zu sichern, daß der Benutzer diesen akzeptiert und keine falschen Antworten durch die Sprunghaftigkeit und fehlende Nachvollziehbarkeit des Dialoges gegeben werden. Eine weitere Anforderung an Dialoge ist, daß die *wichtigsten Integritätsbedingungen* im Dialog zuerst erfragt werden sollen, damit diese, wenn nur eine unvollständige Erfragung möglich ist, für die nachfolgenden Anwendungen vorhanden sind.
Aufgrund dieser drei Kriterien muß eine Reihenfolge des Dialoges zur Validierung der unbekannten Integritätsbedingungen festgelegt werden, eine Möglichkeit dazu wird in diesem Kapitel gezeigt. Die Dialogsteuerung verwendet die Abschätzungen der Integritätsbedingungen und die ermittelten Metainformationen, um eine optimale Reihenfolge des Dialoges nach den angegebenen drei Kriterien festzulegen.

Abbildung 8.1 zeigt, wie die Dialogsteuerung sich in die Erfragung von Integritätsbedingungen eingliedert.

Dieses Kapitel ist wie folgt gegliedert. Zunächst wird in Abschnitt 8.1 angegeben, wie groß die Anzahl der unbekannten Integritätsbedingungen eines Relationen-Schemas bzw. Datenbank-Schemas sein kann. Anschließend wird in Abschnitt 8.2 gezeigt, wie Integritätsbedingungen aus anderen Integritätsbedingungen ableitbar sind. Diese Eigenschaft wird verwendet, um eine Dialogsteuerung zu erreichen, die das Kriterium Effizienz erfüllt. Anschließend folgt in Abschnitt 8.3 eine Dialogsteuerung, die neben der Effizienz auch die Nachvollziehbarkeit und Erfragung der wichtigsten Integritätsbedingungen berücksichtigt. In Abschnitt 8.4 erfolgt eine Bewertung der angegebenen Dialogsteuerung, es wird dabei angegeben, von welchen Größen die Anzahl der notwendigen Dialogschritte abhängig ist.

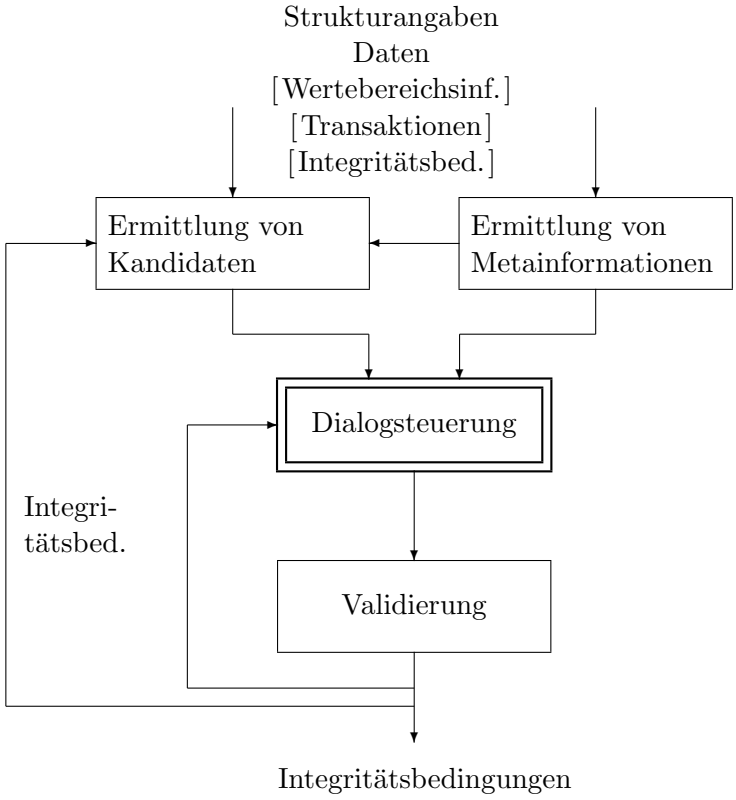


Abbildung 8.1: Einordnung der Dialogsteuerung in die Semantikakquisition

8.1 Komplexitätsprobleme

Die Anzahl unbekannter Integritätsbedingungen eines Relationen-Schemas oder Datenbank-Schemas kann sehr groß sein.
Die Anzahl möglicher *funktionaler Abhängigkeiten* eines Relation-Schemas beträgt $2^n * 2^n$, wobei n die Anzahl der Attribute des Relationen-Schemas ist. Die Anzahl der rechtsminimalen nichttrivialen funktionalen Abhängigkeiten (auf die die Akquisition beschränkt werden kann) beträgt $2^n * \frac{n}{2}$, liegt also ebenfalls in der Ordnung $O(2^n)$. Auch die *Anzahl der voneinander unabhängigen funktionalen Abhängigkeiten* liegt in der Ordnung $O(2^n)$.
Die Anzahl möglicher *Schlüssel* eines Relationen-Schemas beträgt 2^n.
Die Anzahl der möglichen *unären Inklusions- und Exklusionsabhängigkeiten* ist in der Ordnung m^2, wobei m hier die Anzahl der Attribute eines Datenbank-Schemas ist.

Bei der vollständigen Semantikakquisition müssen alle unbekannten Integritätsbedingungen untersucht werden. Man kann leicht ersehen, daß eine vollständige Semantikakquisition schon für kleine Relationen-Schemata (Attributanzahl>6) zu einem sehr langen Dialog mit dem Benutzer führen würden, wenn alle konstruierbaren Integritätsbedingungen der Reihe nach diskutiert werden. Für größere Relationen-Schemata oder größere Datenbanken ist dieses Vorgehen gar nicht möglich. Die *große Anzahl* möglicher lokaler und globaler Integritätsbedingungen ist besonders deshalb *kritisch*, weil diese nicht eine entsprechende Anzahl von Berechnungsschritten, sondern eine entsprechende Anzahl von *notwendigen Dialogschritte* darstellt.

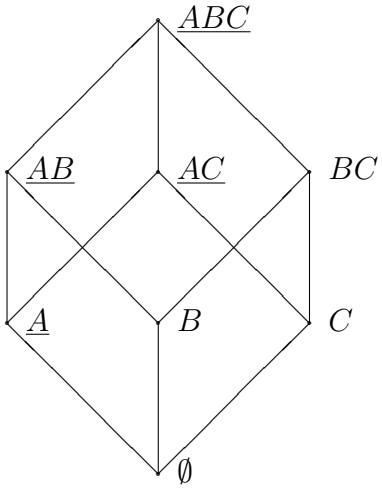
8.2 Effiziente Erfragungsreihenfolge

Die Anzahl der Dialogschritte zur Untersuchung von unbekannten Integritätsbedingungen entspricht im ungünstigsten Fall der Anzahl der unbekannten Integritätsbedingungen. Es ist deshalb nicht möglich, alle Integritätsbedingungen im Dialog mit dem Benutzer der Reihe nach zu untersuchen und zu erfragen.
Integritätsbedingungen sind aus anderen Integritätsbedingungen ableitbar, deshalb hat die *Reihenfolge der Erfragung* von Integritätsbedingungen Auswirkungen auf die *Anzahl der notwendigen Dialogschritte*. Eine effiziente Reihenfolge der Erfragung kann die Anzahl der erforderlichen Dialogschritte minimieren.
Abschnitt 8.2.1 zeigt die Ableitung von Integritätsbedingungen aus anderen Integritätsbedingungen, in Abschnitt 8.2.2 wird gezeigt, wie man die Anzahl der aus einem Constraint ableitbaren weiteren Constraints berechnen kann. Eine Vereinfachung dieser Berechnung wird in Abschnitt 8.2.3 gegeben.

8.2.1 Ableitung von Constraints aus anderen Constraints

Aus der in Kapitel 2 gezeigten Axiomatisierung von Integritätsbedingungen im Kapitel 2 ergibt sich, daß Constraints aus anderen Constraints abgeleitet werden können. Dieses soll hier für ein Beispiel gezeigt werden. Es wird ein Relationen-Schema mit nur drei Attributen $R = (A, B, C)$ gewählt, da dieses graphisch anschaulich darstellbar ist. Es soll die Ableitung von Schlüsseln aus anderen Schlüsseln erläutert werden. Abbildung 8.2 zeigt eine graphische Darstellung aller Attributmengen des angegebenen Relationen-Schemas.

Ist eine Attributmenge des Relationen-Schemas Schlüssel, so läßt sich ableiten, daß auch alle Obermengen dieser Attributmenge Schlüssel sind.

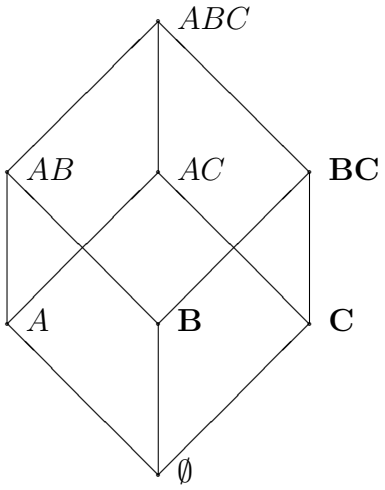


Aus einem Schlüssel A können folgende weitere Schlüssel abgeleitet werden

(A - Schlüssel) $\Rightarrow$ (AB - Schlüssel)
(A - Schlüssel) $\Rightarrow$ (AC - Schlüssel)
(A - Schlüssel) $\Rightarrow$ (ABC - Schlüssel)

Abbildung 8.2: Ableitung von Schlüsseln aus anderen Schlüsseln

Auch aus negierten Schlüsseln lassen sich weitere Constraints ableiten, Abbildung 8.3 zeigt das ebenfalls an einem Beispiel.



Aus einem negierten Schlüssel BC können folgende weitere negierte Schlüssel abgeleitet werden

(BC - negierter Schlüssel) $\Rightarrow$ (B - negierter Schlüssel)

(BC - negierter Schlüssel) $\Rightarrow$ (C - negierter Schlüssel)

Abbildung 8.3: Ableitung von negierten Schlüsseln aus anderen negierten Schlüsseln

Es gibt also innerhalb eines Relationen-Schemas oder Datenbank-Schemas Constraints, aus denen sich weitere Constraints ableiten lassen. Diese Eigenschaft kann man ausnutzen, um die Erfragung der unbekannten Constraints zu effektivieren.

8.2.2 Berechnung des zu erwartenden Informationsgewinns durch Constraints

Bei der effizienten Erfragung sollen zuerst solche Constraints erfragt werden, aus denen die meisten anderen, noch unbekannten Constraints abgeleitet werden können. Diese *Anzahl der aus einem Constraint ableitbaren, noch unbekannten Constraints (Anz)* wird wie folgt berechnet:

$$Anz(\sigma) := |\{\theta | (\Sigma \nvdash \theta) \wedge (\Sigma \cup \sigma \vdash \theta)\}|$$

wobei Σ die Menge der bekannten Integritätsbedingungen ist.

Ist eine Integritätsbedingung noch unbekannt, so weiß man nicht, ob diese noch unbekannte Integritätsbedingung bei der Validierung als geltende oder negierte Integritätsbedingungen festgelegt wird. Will man abschätzen, wieviele Constraints sich aus einer unbekannten Integritätsbedingung σ ableiten lassen, so muß deshalb sowohl die Anzahl der aus σ ableitbaren als auch die Anzahl der aus $\neg\sigma$ ableitbaren Constraints berücksichtigt werden.
Der *Informationsgewinn (IG)* durch eine Integritätsbedingung σ (die mittlere Anzahl ableitbarer Constraints) kann dann nach folgender Formel abgeschätzt werden:

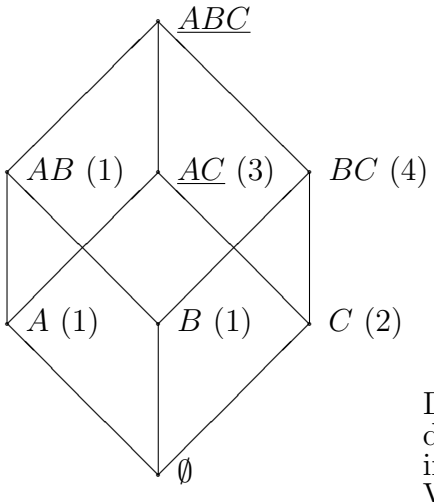
$$IG(\sigma) := \frac{Anz(\sigma) + Anz(\neg\sigma)}{2}$$

Beispiel 8.1: Gegeben sei ein Relationen-Schema $R = (A, B, C)$, in dem noch keine Constraints bekannt sind.

Aus einem zu untersuchenden Schlüssel A lassen sich – falls A als Schlüssel validiert wird – drei unbekannte Constraints ableiten (A - Schlüssel, AB - Schlüssel, AC - Schlüssel).
Aus dem negierten Schlüssel A läßt sich nur ein unbekanntes Constraint ableiten (A - negierter Schlüssel).
Der Informationsgewinn des Schlüsselkandidaten A ist also nach obiger Formel 2.

Hat ein Constraint den Informationsgewinn 0, so ist dieses Constraint bereits bekannt (aus der Menge der bekannten Constraints ableitbar). Ist der Informationsgewinn größer oder gleich 1, so ist das Constraint noch unbekannt, es muß also untersucht werden.

Eine Möglichkeit der Dialogsteuerung, bei der jeweils das Constraint mit dem größten Informationsgewinn als nächstes diskutiert wird, wird in Abbildung 8.4 anhand eines Beispiels gezeigt.



Relationen-Schema $R = (A, B, C)$,

man weiß, daß $\emptyset$ negierter Schlüssel und ABC Schlüssel ist

Validierungsschritte
(1) - Validierung von AB : Erg. - neg. Schl
(2) - Validierung von C : Erg. - neg. Schl
(3) - Validierung von AC : Erg. - Schl
(4) - Validierung von BC : Erg. - neg. Schl

Die unterstrichenen Attributmengen sind Schlüssel, die anderen Attributmengen negierte Schlüssel, in Klammern ist angegeben, aufgrund welches Validierungsschrittes diese erkannt wurden.

Abbildung 8.4: Effiziente Erfragung von Schlüsseln

Kennt man keine weiteren Aussagen über die Semantik des Datenbank-Schemas, so liefert diese Abschätzung für den Informationsgewinn die besten erreichbaren Ergebnisse.

Mit den Methoden zur Ableitung von Kandidaten für Integritätsbedingungen in Kapitel 5 und 6 wurden jedoch auch *Plausibilitäten* für die Kandidaten für Constraints ermittelt. Diese können bei der effizienten Erfragung ausgenutzt werden, da durch *Einbeziehung der Plausibilitäten* der *Informationsgewinn genauer abgeschätzt* werden kann.

$$IG(\sigma) := Pl(\sigma) * Anz(\sigma) + Pl(\neg\sigma) * Anz(\neg\sigma)$$

wobei $Pl(\sigma)$ - Plausibilitätsabschätzung für eine Integritätsbedingung σ
(in Kapitel 5 und 6 ausführlich erläutert)

Auf diese Weise werden die vorgeschlagenen Heuristikregeln in die Berechnung des Informationsgewinns einbezogen. Durch Ausnutzen dieser Abschätzung kann die effizienteste Erfragungsreihenfolge zur Validierung unbekannter Constraints bestimmt werden, indem jeweils das

Constraint mit dem höchsten Informationsgewinn als nächstes erfragt wird.

Die Berechnung des Informationsgewinns ist jedoch sehr aufwendig, da dieser nach jeder Änderung der Menge der bekannten Constraints Σ_P und Σ_N für alle noch unbekannten Constraints neu berechnet werden muß.
Man kann allerdings auch ermitteln, welche Änderungen sich in der Menge der einzelnen Informationsgewinn-Werte durch neu hinzukommende Constraints ergeben und damit nur die sich ändernden Informationsgewinn-Werte neu berechnen. Dadurch wird die Berechnung der Informationsgewinn-Werte effizienter. Eine weitere Möglichkeit der Vereinfachung folgt im nächsten Abschnitt.

8.2.3 Vereinfachte Methode zur Ermittlung von Constraints mit großem Informationsgewinn

Für große Datenbank-Schemata mit vielen unbekannten Constraints ist die Berechnung des Informationsgewinns für jedes unbekannte Constraint *zu aufwendig*. Es kann jedoch eine *Vereinfachung* des Verfahrens genutzt werden. Anstelle der im vorigen Kapitel vorgestellten Abschätzung des Informationsgewinnes läßt sich aussagen, aus welchen *Constraints* generell die *meisten Informationen* abgeleitet werden können.

Lokale Integritätsbedingungen. Bei den *innerhalb eines Relationen-Schemas geltenden Integritätsbedingungen* sind aus folgenden Constraints in der Regel viele weitere Constraints ableitbar. Es ist weiterhin angegeben, aufgrund welcher geltenden Beziehung aus diesen Constraints die meisten Informationen über die Semantik abgeleitet werden können:

Sei R ein Relationen-Schema, X, Y, Z Attributmengen von R:

- Schlüssel mit wenigen Attributen
  (X ist Schlüssel) $\Rightarrow$ (XY ist Schlüssel)
- negierte Schlüssel mit vielen Attributen
  (XY ist negierter Schlüssel) $\Rightarrow$ (X ist negierter Schlüssel)
- funktionale Abhängigkeiten mit wenigen Attributen auf der linken Seite
  $(X \longrightarrow Y) \Rightarrow (XZ \longrightarrow Y)$
- negierte funktionale Abhängigkeiten mit vielen Attributen auf der linken Seite
  $(XZ \not\longrightarrow Y) \Rightarrow (X \not\longrightarrow Y)$

Globale Integritätsbedingungen. Auch für die *Integritätsbedingungen, die über mehreren Relationen-Schemata definiert sind,* läßt sich angeben, aus welchen Constraints die meisten Informationen abgeleitet werden können:

Seien R und S Relationen-Schemata eines Datenbank-Schemas D, W, X Attributsequenzen von R und Y, Z Attributsequenzen von S, die Attributsequenzen W und Y, sowie X und Z sind gleichlang:

- Inklusionsabhängigkeiten über großen Attributsequenzen
  $(WX \subseteq YZ) \Rightarrow (W \subseteq Y)$
- negierte Inklusionsabhängigkeiten über kleinen Attributsequenzen
  $(W \not\subseteq Y) \Rightarrow (WX \not\subseteq YZ)$
- Exklusionsabhängigkeiten über kleinen Attributsequenzen
  $(W \mid\mid Y) \Rightarrow (WX \mid\mid YZ)$
- negierte Exklusionsabhängigkeiten über großen Attributsequenzen
  $(WX \nparallel YZ) \Rightarrow (W \nparallel Y)$

Aus den Constraints, aus denen viele weitere Constraints ableitbar sind, werden *solche Kandidaten* ausgesucht, die nach Auswertung der Heuristikregeln *sehr plausibel* erscheinen. Es ist sinnvoll, die Validierung mit diesen Constraints zu beginnen, da diese generell den höchsten Informationsgewinn (*IG*) haben. Bei der Berechnung des Informationsgewinnes wird die Plausibilität durch die in Kapitel 5 und 6 erläuterten Heuristiken abgeschätzt, die Anzahl ableitbarer Constraints (*Anz*) wird durch die Annahmen in den beiden obigen Aufzählungen abgeschätzt.

Man beginnt also die Validierung mit plausiblen Kandidaten für Schlüssel, die wenige Attribute enthalten, und plausiblen Kandidaten für negierte Schlüssel, die viele Attribute enthalten usw., um bei der Erfragung das Kriterien Effizienz zu erreichen.

8.3 Effiziente und benutzerfreundliche Dialogreihenfolge

Da die Validierung von Integritätsbedingungen im Dialog mit dem Benutzer durchgeführt wird, kann nicht nur das *Kriterium der Effizienz*, das im vorigen Abschnitt erläutert wurde, zugrunde gelegt werden, um die Reihenfolge des Dialoges festzulegen. Es müssen bei dem Dialog mit dem Benutzer noch zwei weitere Bedingungen beachtet werden.
Der *Dialog* muß für den Benutzer *verständlich und nachvollziehbar gestaltet* werden.
Weiterhin soll die Erfragung mit den *wichtigsten Integritätsbedingungen* beginnen, also mit den Integritätsbedingungen, die größte Bedeutung in den Anwendungen, die Integritätsbedingungen auswerten, haben. Dadurch will man erreichen, daß in dem Fall, in dem keine vollständige Erfragung möglich ist, die bedeutendsten Integritätsbedingungen ermittelt werden.

Die Anforderungen, die sich aus diesen beiden weiteren Kriterien ergeben, werden in den Abschnitten 8.3.1 und 8.3.2 aufgezählt, anschließend folgt in Abschnitt 8.3.3 ein Algorithmus zur Dialogsteuerung, der alle drei Kriterien zur Dialogsteuerung berücksichtigt.

8.3.1 Anforderungen an die Mensch-Maschine-Kommunikation

Dialoge mit einem Benutzer müssen die menschliche Psychologie berücksichtigen. Einige allgemeine Forderungen sind folgende (vgl. auch [Düs96], [Lew96]):

- *Verständlichkeit*
  verständliche Fragen, Erklärungen zu den Fragen, Erläuterung von Antwortmöglichkeiten

- *Abwechslungsreichtum*
  keine monotonen Fragestellungen, keine sich wiederholenden Fragen

- *inhaltliche Strukturiertheit*
  umfassendes Behandeln inhaltlich zusammenhängender Teile, Nachvollziehbarkeit der Fragenreihenfolge, plausibel erscheinende Dialogreihenfolge

- *Flexibilität*
  Einstellung auf den Benutzer, Zulassen der Möglichkeit, Fragen nicht zu beantworten

- *spezielle Angepaßtheit an den Entwerfer*
  Optimal ist eine Dialogsteuerung, die aus den Antworten des Entwerfers Schlußfolgerungen zieht, die den weiteren Dialogverlauf beeinflussen.

Eine Steuerung des Dialoges aufgrund dieser Charakteristika ist notwendig.

Die Validierung von Integritätsbedingungen wird im Dialog mit dem Entwerfer anhand von Beispieldiskussionen erfolgen (Kapitel 9). Damit kann man der Forderung nach *Verständlichkeit* der Fragen genügen. Es ist dabei möglich, Fragen unbeantwortet zu lassen, also wird auch das Kriterium der *Flexibilität* berücksichtigt.

Die Kriterien inhaltliche Strukturiertheit und spezielle Angepaßtheit an den Entwerfer müssen durch die Reihenfolge des Dialoges erfüllt werden. Um das Kriterium *Strukturiertheit* zu erreichen, erfolgt die Diskussion von Schlüsseln und funktionalen Abhängigkeiten für die einzelnen Relationen-Schemata nacheinander. Zusammengehörende Informationen sollen dabei auch im Zusammenhang diskutiert werden. In Relationen-Schemata mit vielen Attributen werden deshalb Attribut-Cluster (vorgestellt in Abschnitt 7.4) ermittelt, die Erfragung von Integritätsbedingungen innerhalb eines Clusters erfolgt im Zusammenhang.
Bei der Untersuchung von Inklusions- und Exklusionsabhängigkeiten, sowie Kardinalitäten sollen ebenfalls große Sprünge zwischen den Datenbankteilen vermieden werden. In großen Datenbank-Schemata mit vielen Relationen-Schemata erfolgt deshalb die Untersuchung von Integritätsbedingungen zuerst in ermittelten Relationen-Cluster der Datenbank. In Abschnitt 7.5 wurde gezeigt, wie diese ermittelt werden.

Eine Strukturiertheit kann dadurch erreicht werden, daß bestimmte Integritätsbedingungen im Zusammenhang diskutiert werden. In einem Dialog mit dem Benutzer kann das so realisiert werden, daß nach dem Diskutieren bestimmter Integritätsbedingungen Nachfragen zur Präzisierung der Integritätsbedingungen oder zum Finden ähnlicher Integritätsbedingungen gestellt werden. Sinnvolle Nachfragen dazu werden im folgenden aufgezählt.

Nachfragen zur Präzisierung von Integritätsbedingungen

Einige erkannte Integritätsbedingungen können präzisiert werden. Die folgende Übersicht zeigt, um welche Integritätsbedingungen es sich dabei handelt und welche Nachfragen zur Präzisierung erfolgen müssen.

negierte Inklusionsabhängigkeit $X \not\subseteq Y$	Es wird untersucht, ob zwischen den Attributsequenzen X und Y eine *Exklusionsabhängigkeit* $X \mid\mid Y$ gilt.
negierte Exklusionsabhängigkeit $X \not\parallel Y$	Zwischen den Attributsequenzen X und Y wird überprüft, ob eine *Inklusionsabhängigkeit* $X \subseteq Y$ oder $Y \subseteq X$ gilt.
Kardinalität	Anhand von Beispieldiskussionen können nur die Kardinalitäten $\mathsf{card}(R, S) = (0, 1), (1, 1), (0, n)$ oder $(1, n)$ gefunden werden. Für diese *Kardinalitäten* kann der *genaue Wert* erfragt werden. Dieses geschieht durch eine direkt formulierte Frage nach den Kardinalitäten: −Wie oft kann ein Wert aus S höchstens in R auftreten ? −Wie oft muß ein Wert aus S mindestens in R auftreten ?

Diese Nachfragen zu den Integritätsbedingungen sollten direkt im Anschluß an die Diskussion der Integritätsbedingungen erfolgen. Dadurch wird eine *gewisse Kontinuität des Dialoges* erreicht.

Nachfragen zur Erweiterung von Integritätsbedingungen

Es kann möglich sein, daß weitere Integritätsbedingungen in einem Datenbank-Schema gelten, die mit einer gerade abgeleiteten Integritätsbedingung in engem Zusammenhang stehen. Diese sollen ebenfalls im Anschluß diskutiert werden. Die folgende Übersicht zeigt, nach welchen ermittelten Integritätsbedingungen welche weiteren Integritätsbedingungen diskutiert werden können:

funktionale Abhängigkeit ($X \longrightarrow Y$)	Es wird untersucht, ob X *Schlüssel* der Relation ist.
	Es wird überprüft, ob weitere *funktionale Abhängigkeiten* mit der gleichen linken Seite existieren ($X \longrightarrow Z$).
negierte funktionale Abhängigkeit ($X \not\longrightarrow Y$)	Weitere *negierte funktionale Abhängigkeiten* mit der gleichen linken Seite ($X \not\longrightarrow Z$) werden diskutiert.
Inklusionsabhängigkeit ($R.X \subseteq S.Y$)	Weitere Kandidaten für *Inklusionsabhängigkeiten* zwischen den Relationen-Schemata R und S werden gesucht und gegebenenfalls erfragt.
Inklusionsabhängigkeiten $R.X \subseteq T.Z$ *und* $S.Y \subseteq T.Z$	Es wird überprüft, ob zwischen $R.X$ und $S.Y$ eine *Inklusionsabhängigkeit* ($R.X \subseteq S.Y$) oder ($S.Y \subseteq R.X$) gilt.
	Es wird überprüft, ob zwischen $R.X$ und $S.Y$ eine *Exklusionsabhängigkeit* ($R.X \mid\mid S.Y$) gilt.
Inklusionsabhängigkeit $R.X \subseteq S.Y$ X *Schlüssel* in R	Es muß untersucht werden, ob Y *Schlüssel* in S ist.
Inklusionsabhängigkeit $R.X \subseteq S.Y$ Y *Schlüssel* in S	Es wird untersucht, ob X *Schlüssel* in R ist.
	Die *Kardinalität* $\mathsf{card}(R, S)$ muß bestimmt werden.
Exklusionsabhängigkeit ($R.X \mid\mid S.Y$)	Es erfolgt die Suche nach einem "Überbegriff" T von R und S, entsprechende möglicherweise geltende *Inklusionsabhängigkeiten* ($R.X \subseteq T.Z$) und ($S.Y \subseteq T.Z$) werden gesucht.
	Es werden weitere *Exklusionsabhängigkeiten* zwischen den Relationen R und S gesucht und gegebenenfalls erfragt.

Auch diese Nachfragen zur Erweiterung der Integritätsbedingungen bewirken eine *Strukturierung des Dialoges*, da die Integritätsbedingungen, die einander ähnlich sind oder die Integritätsbedingungen, die im Zusammenhang stehen, auch zusammenhängend geklärt werden.

8.3.2 Auswahl der bedeutendsten Integritätsbedingungen

Bei der Semantikakquisition werden Informationen über geltende Integritätsbedingungen (Σ_P) und negierte Integritätsbedingungen (Σ_N) ermittelt. Ist die Semantikakquisition nicht vollständig möglich, so möchte man zumindest die Integritätsbedingungen aus Σ_P mit den *stärksten Auswirkungen in den Anwendungen* - (einige davon sind in Kapitel 1.2.5 aufgezählt) - ermitteln.

Es ist dabei günstig, die Diskussion von Integritätsbedingungen mit *Kandidaten für geltende Integritätsbedingungen* zu beginnen. Zum einen wird die *Benutzerakzeptanz* auf diese Weise erhöht, da die Fragen zu diesen unbekannten Integritätsbedingungen sinnvoller erscheinen, andererseits ist es gerade bei *unvollständiger Erfragung* hilfreich, wenn man einige geltende Integritätsbedingungen finden konnte.

Um die wichtigsten Integritätsbedingungen eines Datenbank-Schemas zu finden, müßte man also betrachten, welche Integritätsbedingungen welche Auswirkungen auf die in Abschnitt 1.2.5 aufgezählten Anwendungen wie Normalisierung, Übersetzung ins physische Schema, Optimierung, Konsistenzüberwachung, Viewintegration, Wiederverwendung von Datenbanken, usw. haben. Eine generelle Aussage zur Bedeutung der Integritätsbedingungen ist aus folgenden Gründen nicht einfach zu machen:

- Da *sehr viele verschiedene Anwendungen* der Semantik existieren, kann man *nicht für alle* untersuchen, welche Integritätsbedingungen welche Auswirkungen haben. Allgemeingültige Aussagen über wichtige Integritätsbedingungen und eine entsprechend darauf aufbauende Steuerung und Einschränkung der Semantikakquisition ist also nicht möglich.

- In großen Datenbank-Schemata ist die Anzahl der unbekannten Integritätsbedingungen sehr groß. Da man nicht weiß, welche der unbekannten Integritätsbedingungen gelten, müßte also für alle untersucht werden, welche Auswirkungen sie auf eine Anwendung haben. Deshalb ist auch *für ein Anwendungsgebiet der Semantik* die Untersuchung der Bedeutung von Integritätsbedingungen *zu komplex*.

- Kann man trotz dieser Probleme feststellen, welche Integritätsbedingungen große Änderungen bewirken würden, so wird häufig der Fall auftreten, daß gerade solche Integritätsbedingungen die meisten Änderungen bewirken würden, die am unwahrscheinlichsten sind (in der Abschätzung in Kapitel 5 und 6 am wenigsten plausibel). Es ist jedoch nicht sinnvoll, vorwiegend Integritätsbedingungen zu untersuchen, die wahrscheinlich nicht gelten.

Eine begründete Aufstellung wichtiger und weniger wichtiger Integritätsbedingungen ist also nur in Bezug auf eine oder wenige Anwendungen möglich und auch dann sehr aufwendig. Man kann also auch hier nur einige *allgemeine Vermutungen* angeben, welche Integritätsbedingungen in einem Datenbank-Schema die größte Bedeutung haben und damit bei unvollständiger Validierung untersucht werden sollen. Diese werden im folgenden aufgezählt, die Angaben werden dabei empirisch begründet.

Schlüssel. Existiert ein Primärschlüssel eines Relationen-Schemas, so sind weitere Schlüssel, die mehr Attribute umfassen (oder mehr Speicherbedarf für die Attribute benötigen), in den Anwendungen weniger wichtig. Diese weiteren Schlüssel dienen der Konsistenzüberwachung. Bei Übersetzungen des Datenbank-Schemas wird jedoch der Schlüssel mit den wenigsten Attributen oder geringstem Speicherplatzbedarf als Fremdschlüssel verwendet.

Funktionale Abhängigkeiten. Nach [BGM85] und [BoG86] sind funktionale Abhängigkeiten mit mehr als vier oder fünf Attributen auf der linken Seite unwahrscheinlich. Das entspricht auch der Vorstellung, daß die meisten Zusammenhänge in der realen Welt relativ überschaubar in den Datenbanken dargestellt sind beziehungsweise daß Betriebsmodelle relativ überschaubar organisiert sind. Das widerspiegelt sich in der natürlichen Sprache, dort werden meist nur wenige Sachverhalte über Konnektoren verbunden.
Treten solche funktionalen Abhängigkeiten mit sehr vielen Attributen auf der linken Seite jedoch auf, so wird hier häufig aus Effizienzgründen auf eine Normalisierung verzichtet, da bei der Normalisierung (zum Erreichen der dritten Normalform) die Attribute der linken Seite der funktionalen Abhängigkeit in zwei Relationen-Schemata redundant gespeichert werden würden. Deshalb kann man in Relationen-Schemata, in denen keine vollständige Untersuchung der Semantik erreichbar ist, auf die Erfragung möglicher funktionaler Abhängigkeiten mit sehr vielen Attributen auf der linken Seite verzichten.

Inklusionsabhängigkeiten. Der Suchraum zur Ermittlung von *Inklusionsabhängigkeiten*, die über *mehreren Attributen* definiert sind, ist durch folgende Beziehung eingeschränkt:

$$(R.X_1..R.X_n) \subseteq (S.Y_1..S.Y_n) \Rightarrow (R.X_1 \subseteq S.Y_1 \wedge .. \wedge R.X_n \subseteq S.Y_n)$$

Voraussetzung für die Gültigkeit von Inklusionsabhängigkeiten, die über mehreren Attributen definiert sind, ist also, daß die entsprechenden unären Inklusionsabhängigkeiten erfüllt sind. Inklusionsabhängigkeiten über mehreren Attributen müssen also nur über den geltenden unären Inklusionsabhängigkeiten gesucht werden. Deshalb ist die Ermittlung unärer Inklusionsabhängigkeiten besonders wichtig.

Die meisten auftretenden Inklusionsabhängigkeiten sind nach [MaR92] *schlüsselbasiert*, können also im Entity-Relationship Modell strukturell dargestellt werden. Auftretende Inklusionsabhängigkeiten, die nicht schlüsselbasiert sind, werden zur Konsistenzprüfung verwendet, sie können aber auch Umstrukturierungen des Datenbank-Schemas bewirken. Diese Inklusionsabhängigkeiten sind deshalb besonders wichtig. Man kann also die Suche nach Inklusionsabhängigkeiten in großen Datenbank-Schemata nicht auf schlüsselbasierte Inklusionsabhängigkeiten beschränken. Eine Untersuchung der Analoga $R.X \approx S.Y$, bei denen eine Attributsequenz Schlüssel ist, reicht also nicht aus.

Treten zwischen Relationen-Schemata *mehrere Inklusionsabhängigkeiten* auf, so ist diejenige, die über den *wenigsten Attribute* (oder den Attributen mit dem geringsten Speicherplatzbedarf) definiert ist, am wichtigsten, denn diese werden für Joins bei Anfragen auf der Datenbank verwendet. Deshalb beginnt die Diskussion von Inklusionsabhängigkeiten bei unären Inklusionabhängigkeiten. Wurde zwischen zwei Relationen-Schemata eine Inklusionsabhängigkeit gefunden, so kann (wenn keine vollständige Semantikakquisition möglich ist) die Suche nach weiteren Inklusionsabhängigkeiten abgebrochen werden oder erst nach Überprüfung der Inklusionsabhängigkeiten zwischen anderen Relationen-Schemata erfolgen.

Exklusionsabhängigkeiten. Bestehen in einem Datenbank-Schema Exklusionsabhängigkeiten zwischen zwei Relationen-Schemata, so wird dadurch angegeben, daß keine Objekte in beiden Relationen-Schemata auftreten. In diesem Fall existiert meist ein gemeinsamer "Überbegriff" für die beiden Relationen-Schemata, also ein Relationen-Schema, zu dem Inklusionsabhängigkeiten bestehen.
Die Suche nach Exklusionsabhängigkeiten kann eingeschränkt werden, indem nur solche Kandidaten $R.X \,||\, S.Y$ erfragt werden, für die *geltende Inklusionsabhängigkeiten* $R.X \subseteq T.Z$ und $S.Y \subseteq T.Z$ bekannt sind.

Kardinalitäten. Voraussetzung zur Bestimmung von Kardinalitäten ist die Existenz von Fremdschlüsselbeziehungen. Damit ist der Suchraumes zur Ermittlung von Kardinalitäten bereits durch die *Schlüssel* und *Inklusionsabhängigkeiten* stark eingeschränkt.

Zur Auswertung der *Bedeutung von Integritätsbedingungen* wird ein Faktor benötigt, der diese abschätzt. Diese Berechnung ist sehr empirisch aufgestellt, sie ist hier nur angegeben, da die vorherigen Aussagen zu inkonkret für eine Verwendung in einem Algorithmus sind.

Schlüssel: $Bed(\text{Schlüssel } X) := \frac{1}{|X|} * \frac{1}{|S|+1}$

$|X|$ - Attributanzahl des Schlüsselkandidaten
S - Anzahl der bereits existierenden Schlüssel des Relationen-Schemas

funktionale Abhängigkeiten: $Bed(X \longrightarrow Y) := \frac{1}{|X|}$

$|X|$ - Attributanzahl der linken Seite des Kandidaten für eine funktionale Abhängigkeit

Inklusionsabhängigkeiten: $Bed(R.X \subseteq S.Y) := \frac{1}{|R.X|} * \frac{1}{|I|+1}$

$|R.X|$ - Attributanzahl des Kandidaten für eine Inklusionsabhängigkeit
I - weitere Inklusionsabhängigkeiten zwischen den Relationen-Schemata R und S

Exklusionsabhängigkeiten werden nur über bekannten Inklusionsabhängigkeiten gesucht, Kardinalitäten nur über bekannten Schlüsseln und Inklusionsabhängigkeiten. Damit sind diese Arten von Integritätsbedingungen bereits so stark eingeschränkt und die Kandidaten sind so plausibel, daß alle diese Kandidaten untersucht und erfragt werden.

8.3.3 Algorithmus zur Steuerung des Dialoges

In den vorherigen Abschnitten wurden als Anforderungen an der Dialog die Kriterien *Effizienz, Benutzerfreundlichkeit und Erfragung der wichtigsten Integritätsbedingungen* genannt und

erläutert. Eine Methode zur Gestaltung effizienter Dialoge wurde bereits gezeigt sowie eine Erläuterung der anderen beiden Kriterien vorgenommen. In diesem Abschnitt soll eine Dialogsteuerung vorgestellt werden, die diese drei (sich teilweise widersprechenden) Kriterien berücksichtigt.

Die Auswahl und Validierung von Kandidaten erfolgt nach folgendem Algorithmus:

```
PROCEDURE umfassende_Validierung(σ: Constraint; VAR Σ: Integritätsbedingungen);
BEGIN
    ⟨ Validierung und Abspeicherung von σ (Σ := Σ ∪ σ) ⟩
    IF Präzisierung σ' möglich THEN
        umfassende_Validierung(σ', Σ)
    END;
    WHILE Erweiterung σ'' möglich THEN
        umfassende_Validierung(σ'', Σ)
    END;
END umfassende_Validierung;


BEGIN
    Σ := Menge der bekannten Integritätsbedingungen;
    REPEAT
        ⟨ Auswahl des Kandidaten σ mit Σ ⊬ σ
          mit IG(σ) * Bed(σ) ist maximal ⟩
        umfassende_Validierung(σ, Σ)
    UNTIL (∄ σ mit Σ ⊬ σ);
END.
```

Algorithmus 8.1: Allgemeines Vorgehen zur Bestimmung der Validierungsreihenfolge

Eine mit dem Benutzer zu diskutierende Integritätsbedingung wird nach den Kriterien 1. maximale Anzahl der ableitbaren Integritätsbedingungen (IG) und 2. Bedeutung der Integritätsbedingungen (Bed) ausgewählt. Es wird die Integritätsbedingung gewählt, bei der das Produkt dieser beiden Werte maximal ist.

Alle Nachfragen zur Präzisierung einer Integritätsbedingung σ und zur Erweiterung einer Integritätsbedingung σ werden unmittelbar im Anschluß an die Diskussion der Integritätsbedingung σ gestellt.

8.4 Bewertung der effizienten Erfragung

Das vorgestellte Verfahren zur Bestimmung der Validierungsreihenfolge genügt sowohl dem Anspruch der *Benutzerfreundlichkeit*, der *Effizienzforderung*, als auch der *Erfragung der wichtigsten Integritätsbedingungen*.

Durch das Stellen von Nachfragen zu einer Integritätsbedingung werden inhaltlich zusammenhängende Integritätsbedingungen nacheinander diskutiert. Die Fragen erscheinen in logischen Zusammenhängen. Dadurch werden die Fragen nachvollziehbar, die Beantwortung ist für den Benutzer leichter.

Ist ein Fragenkomplex beendet, so erfolgt die Auswahl einer zu diskutierenden Integritätsbedingung aufgrund des abgeschätzten Informationsgewinns (*IG*) und der Bedeutung (*Bed*) der zu erfragenden Integritätsbedingungen für die Anwendungen. Damit wird die Semantikakquisition effizient. Durch die Auswahl von zu diskutierenden Integritätsbedingungen nach ihrer Bedeutung versucht man, bei der unvollständigen Semantikakquisition soviele wichtige Informationen über geltende Integritätsbedingungen wie möglich in den ersten Dialogschritten zu finden.

Die Dialogreihenfolge wird durch den angegebenen Algorithmus festgelegt, sie bestimmt die Reihenfolge der Validierung von Integritätsbedingungen. Die Validierung der einzelnen Integritätsbedingungen wird im nächsten Abschnitt erläutert.

Trotz des Einsatzes verschiedener Heuristiken und einer möglichst effizienten Erfragung ist es nicht immer möglich, alle unbekannten Integritätsbedingungen eines Datenbank-Schemas zu untersuchen. Bei *großen Relationen-Schemata* und *großen Datenbank-Schemata* kann die Semantikakquisition auch mit der vorgestellten Methode nur *unvollständig* durchgeführt werden.

Dabei lassen sich keine exakten Angaben machen, wieviele Dialogschritte in welchen Datenbanken notwendig sind, um die Semantik vollständig zu erfassen. Die *Anzahl der notwendigen Schritte* zur Validierung hängt von folgenden Merkmalen ab:

- Anzahl der Relationen-Schemata eines Datenbank-Schemas
- Anzahl der Attribute jedes Relationen-Schemas
- Anzahl der Attribute mit gleichen Längen und Typen in dem Datenbank-Schema, diese Größe hat Auswirkungen auf die Anzahl der zu untersuchenden Inklusions- und Exklusionsabhängigkeiten
- Anzahl und Art der aus den Daten ableitbaren Integritätsbedingungen
- Anzahl und Art der aus der Datenbank-Statistik ableitbaren Integritätsbedingungen
- Anzahl und Art der explizit eingegebenen (bzw. bereits bekannten) Integritätsbedingungen
- Anzahl und Art der in dem Datenbank-Schema geltenden Integritätsbedingungen
- Zutreffen der Plausibilitätsabschätzungen auf die geltenden Integritätsbedingungen der Datenbank-Schemas

Die *Grenzen* dafür, ab welcher Größe *keine vollständige Untersuchung* mehr durchgeführt werden kann, lassen sich deshalb nicht exakt angeben.

Aus den Angaben, von welchen Größen die Anzahl der Dialogschritte abhängig ist, läßt sich ableiten, auf welche Weise man versuchen kann, die Effizienz durch den Einsatz der Dialogsteuerung zu erhöhen. Die ersten sieben aufgezählten Einflüsse lassen sich nicht durch die Dialogsteuerung verändern.

Ausnutzen kann man bei der Dialogsteuerung nur die Eigenschaft:

- Zutreffen der Plausibilitätsabschätzungen auf die geltenden Integritätsbedingungen der Datenbank-Schemas

Man kann also versuchen, durch eine Vorabschätzung der unbekannten Integritätsbedingungen eine gezielte Erfragung von Integritätsbedingungen vorzunehmen. Von dieser Größe hängt auch die Anzahl der notwendigen Dialogschritte ab. Trifft diese Größe zu, liefern also die Heuristikregeln richtige Abschätzungen für die Integritätsbedingungen, so erhöht sich die Effizienz der Dialogsteuerung. Damit sieht man auch, daß es nicht möglich ist, anzugeben, wie effizient die vorgestellte Dialogsteuerung ist, da diese auf der Abschätzung der Plausibilitäten für unbekannte Integritätsbedingungen basiert. Ohne umfangreiche Untersuchungen zum Zutreffen der Heuristikregeln läßt sich also nicht angeben, wie effizient die Erfragung der Integritätsbedingungen mit der vorgestellten Erfragungsreihenfolge ist bzw. wieviel effizienter dieses Vorgehen gegenüber der herkömmlichen Erfragung aller Integritätsbedingungen ist. In Kapitel 5 wurde bereits angegeben, daß das Zutreffen von Heuristikregeln nicht allgemeingültig festzustellen ist.

Man kann also nur vermuten, da die Heuristiken empirisch nachvollziehbar sind, daß ihr Einsatz die Semantikakquisition wesentlich effizienter macht und geltende Integritätsbedingungen dadurch schneller gefunden werden.

Kapitel 9

Validierung von Integritätsbedingungen

In den Kapiteln 5 und 6 wurde gezeigt, wie sich Kandidaten für Integritätsbedingungen ableiten lassen. Eine Festlegung, in welcher Reihenfolge die Diskussion von Integritätsbedingungen erfolgen soll, wurde in Kapitel 8 gezeigt. Die Diskussion der Kandidaten mit dem Benutzer ist Inhalt dieses Kapitels. Dabei soll die Entscheidung über die Gültigkeit von vorgeschlagenen Integritätsbedingungen für den Benutzer in *nicht formaler, einfacher und verständlicher Weise* erfolgen. Abbildung 9.1 zeigt, wie sich die Validierung in die Semantikakquisition einordnet.

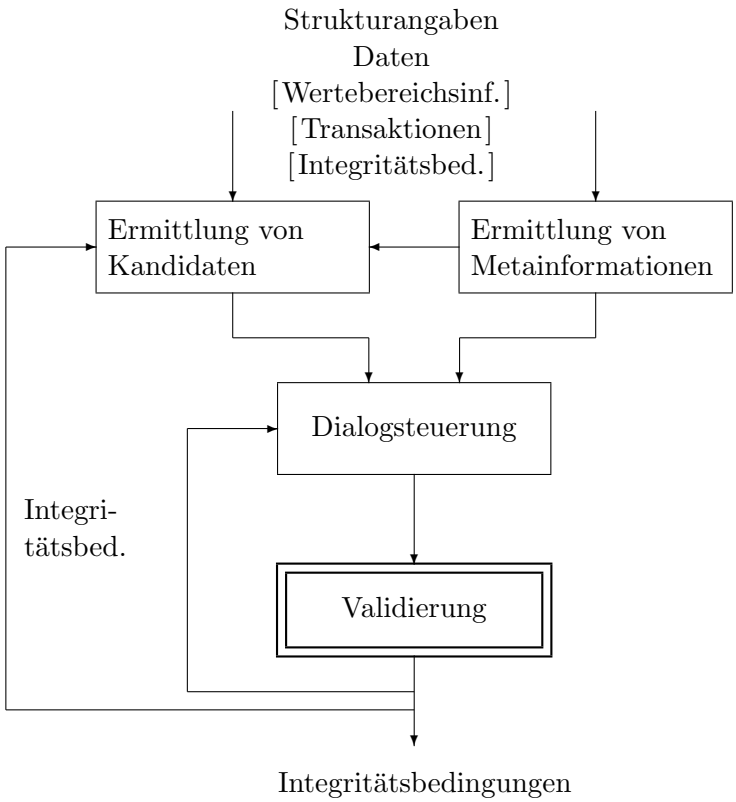


Abbildung 9.1: Einordnung der Validierung in die Diskussion von Integritätsbedingungen

Bei der Validierung werden Kandidaten für geltende Integritätsbedingungen mit dem Benutzer diskutiert. Diese können sowohl bestätigt als auch abgelehnt werden, sie werden dementsprechend als geltende oder negierte Integritätsbedingungen gespeichert. Ebenso können im Dialog mit dem Benutzer Kandidaten für negierte Integritätsbedingungen diskutiert und entsprechend der Validierung als geltende oder negierte Integritätsbedingungen gespeichert werden. Abbildung 9.2 zeigt, welche Änderungen in den Mengen der Integritätsbedingungen und in den Kandidatenmengen aufgrund der Validierung erfolgen.

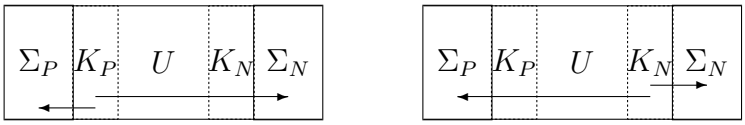


Abbildung 9.2: Veränderung der Mengen der Integritätsbedingungen und der Kandidaten aufgrund der Validierung

Ein bekannter Ansatz zur einfachen und verständlichen Validierung von Kandidaten sind *Armstrongdatenbanken*. Diese wurden kurz in Abschnitt 1.4 erläutert, es wurden auch die dabei auftretenden Probleme gezeigt. Da Armstrongdatenbanken sehr groß werden können und ihre Gültigkeit in diesem Fall nur schwer entscheidbar ist, werden hier keine Armstrongdatenbanken für die Validierung von unbekannten Integritätsbedingungen verwendet. Es wird für jede einzelne unbekannte Integritätsbedingung eine eigene Beispieldatenbank generiert und die Gültigkeit dieser über eine pseudonatürlichsprachige Frage diskutiert.

Bei der Generierung von Beispieldatenbanken ist zu beachten, daß *negierte Integritätsbedingungen* direkt in Beispieldatenbanken darstellbar sind und sich direkt über die Gültigkeit eines Beispiels erfragen lassen. *Geltende Integritätsbedingungen* sind nicht im Beispiel darstellbar, man kann aber die Negation dieser im Beispiel darstellen und die Gültigkeit des Beispiels erfragen. Kann so ein generierter Fall nicht auftreten, so muß die entsprechende Integritätsbedingung gelten. Bei der Erfragung geltende Integritätsbedingungen wird also erfragt, ob das *Gegenteil* der entsprechenden Integritätsbedingung *nicht gelten* kann.
Geltende und negierte Integritätsbedingungen werden also jeweils auf die gleiche Weise erfragt, aus den Antworten ergibt sich, welche Integritätsbedingung jeweils gilt.

Zur Generierung von Beispieldatenbanken werden die *vorhandenen realen Daten* verwendet. Diese sind für den Benutzer anschaulicher als Datenbanken mit abstrakten Daten. Die *pseudonatürlichsprachige Erfragung* der dargestellten Integritätsbedingungen ist so allgemein formuliert, daß sie für alle Datenbankinhalte sinnvoll erscheint.

Eine mögliche Validierung für funktionale Abhängigkeiten, Schlüssel, Inklusions- und Exklusionsabhängigkeiten, sowie Kardinalitäten wird in diesem Kapitel demonstriert.

9.1 Funktionale Abhängigkeiten

Aus der Definition der negierten funktionalen Abhängigkeit (siehe Kapitel 2) ergibt sich, daß Beispielrelationen nur aus zwei Tupeln bestehen müssen, um eine negierte funktionale

Abhängigkeit darzustellen. Mit ihnen soll herausgefunden werden, ob eine funktionale oder negierte funktionale Abhängigkeit zwischen bestimmten Attributmengen besteht. Es erscheint günstig, auch nur zwei Tupel zu generieren, um den Blick des Benutzers auf das Wesentliche zu lenken.

In dem generierten Beispiel müssen ein oder mehrere Attribute (X) den gleichen Wert haben, andere Attribute (Y) mit verschiedenen Werten belegt sein. Das Beispiel repräsentiert dann die negierten funktionalen Abhängigkeiten $(X \not\rightarrow Y_1) \wedge \ .. \ \wedge (X \not\rightarrow Y_n), \forall\, Y_i \in Y$.
Dieses Beispiel wird durch eine pseudonatürlichsprachige Frage ergänzt, die die negierte funktionale Abhängigkeit umschreibt.

Abbildung 9.3 zeigt die Erfragung einer funktionalen bzw. negierten funktionalen Abhängigkeit.

Personen:

Personennr	Name	Vorname	Wohnort	PLZ	Straße	Nummer	Vorwahl	Telefonnr
			==Rostock==				0381	3237426
			==Rostock==				030	38327362

Können zwei Einträge auftreten, die in — Wohnort — den gleichen Wert und
in — Vorwahl und Telefonnr — verschiedene Werte haben (j/n/u) ? _

Abbildung 9.3: Erfragung von funktionalen und negierten funktionalen Abhängigkeiten (1)

Bei Annahme des Beispiels gelten die dargestellten negierten funktionalen Abhängigkeiten, bei Ablehnung des Beispiels muß mindestens eine der funktionalen Abhängigkeiten $X \longrightarrow Y_i, Y_i \in Y$ gelten. Für das Beispiel in Abbildung 9.3 sind folgende Integritätsbedingungen aus der Beantwortung der Frage ableitbar:

$$
\begin{array}{lllll}
\text{bei Annahme:} & (\text{Wohnort} & \not\rightarrow & \text{Vorwahl}) & \wedge \\
 & (\text{Wohnort} & \not\rightarrow & \text{Telefonnr}) & \\
\text{bei Ablehnung:} & (\text{Wohnort} & \longrightarrow & \text{Vorwahl}) & \vee \\
 & (\text{Wohnort} & \longrightarrow & \text{Telefonnr}) &
\end{array}
$$

Wird das Beispiel abgelehnt, so sind die Ergebnisse nicht eindeutig. Deshalb ist eine Nachfrage erforderlich. Diese kann folgendermaßen aussehen:

- Welche Attribute sind von — Wohnort — abhängig ? _

Wird eine rechtsminimale negierte funktionale Abhängigkeit generiert und erfragt, so kann man sowohl bei Annahme als auch Ablehnung des Beispiels eindeutige Informationen über Integritätsbedingungen ableiten. Abbildung 9.4 zeigt dieses:

Personen:

Personennr	Name	Vorname	Wohnort	PLZ	Straße	Nummer	Vorwahl	Telefonnr
			==Rostock==				0381	
			==Rostock==				030	

Können zwei Einträge auftreten, die in — Wohnort — den gleichen Wert und
in — Vorwahl — verschiedene Werte haben (j/n/u) ? _

Abbildung 9.4: Erfragung von funktionalen und negierten funktionalen Abhängigkeiten (2)

Dieses Beispiel liefert die folgenden Informationen:

bei Annahme: Wohnort $\not\longrightarrow$ Vorwahl
bei Ablehnung: Wohnort $\longrightarrow$ Vorwahl

Um möglichst wenige Nachfragen stellen zu müssen, werden Kandidaten für funktionale Abhängigkeiten (die durch die Methoden in Kapitel 5 abgeschätzt wurden) mit nur einem Attribut auf der rechten Seite erfragt. Kandidaten für negierte funktionale Abhängigkeiten können mit mehreren Attributen auf der rechten Seite generiert und diskutiert werden.

Der Beispielzugang ist auch für die Erfragung von Integritätsbedingungen in NF^2- *Relationen* möglich. Die Erfragung von funktionalen und negierten funktionalen Abhängigkeiten zwischen verschachtelten Attributen mit Hilfe von generierten Beispielen erfolgt analog, in Abbildung 9.5 wird eine Studentendatei zur Erläuterung verwendet.

Studenten:

Stud_nr	Name	Vorname	Adresse				Hauptfächer	Nebenfächer
			Wohnort	PLZ	Straße	Nr	Hauptfach	Nebenfach
			==Rostock==	18055				
			==Rostock==	18057				

Können zwei Einträge auftreten, die in — Adresse.Wohnort — den gleichen Wert und
in — Adresse.PLZ — verschiedene Werte haben (j/n/u) ? _

Abbildung 9.5: Erfragung von funktionalen und negierten funktionalen Abhängigkeiten (3)

Kann dieses generierte Beispiel zutreffen, so existiert eine negierte funktionale Abhängigkeit innerhalb des verschachtelten Attributes, kann so ein Fall nicht auftreten, so muß dort eine funktionale Abhängigkeit existieren.

9.2 Schlüssel

Schlüssel sind für viele Benutzer einfacher zu bestimmen oder zu bestätigen als funktionale Abhängigkeiten. Man könnte diese also direkt erfragen.

Es ist aber auch möglich, Schlüssel anhand eines Beispiels zu erfragen. Dazu wird eine Beispielrelation mit zwei Tupeln generiert, in der die Attribute des Schlüsselkandidaten mit gleichen Werten belegt sind. Abbildung 9.6 zeigt dieses:

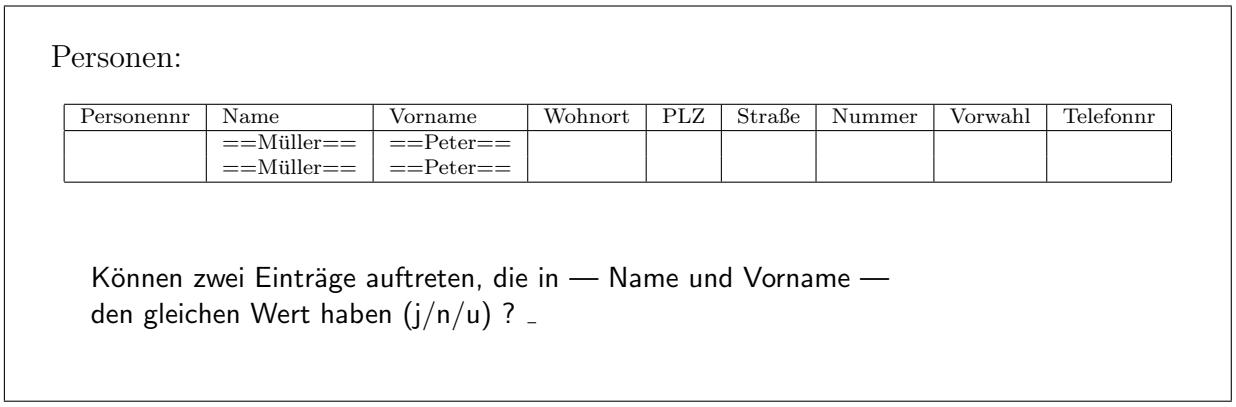


Abbildung 9.6: Erfragung von Schlüsseln bzw. negierten Schlüsseln

Aus der Antwort des Benutzers läßt sich ableiten, ob dieser dargestellte negierte Schlüssel (Name, Vorname) gilt oder in der Relation ein entsprechender Schlüssel bestehend aus den Attributen Name und Vorname gelten muß.

9.3 Inklusionsabhängigkeiten

Die Darstellung einer negierten Inklusionsabhängigkeit anhand von Beispieldaten ist nicht so einfach möglich, wie die Darstellung einer negierten funktionalen Abhängigkeit oder eines negierten Schlüssels. Negierte Inklusionsabhängigkeiten sind gegenüber Tupelergänzungen nicht invariant. In einer Beispieldatenbank können negierte Inklusionsabhängigkeiten gelten, die durch Ergänzen von Tupeln wieder ungültig werden. Die Ableitung negierter Inklusionsabhängigkeiten ist deshalb nur aus abgeschlossenen (vollständigen) realen Datenbanken möglich, solche müßten auch zur Erfragung generiert werden.

Man kann jedoch Beispielrelationen generieren, die der Benutzer als in sich abgeschlossen ansehen muß. Dazu werden zwei Beispielrelationen r und s erzeugt, bei denen X mit einigen Einträgen belegt ist und in Y nur ein Eintrag steht, der in X nicht vorkommt.

Die im Beispiel dargestellte negierte Inklusionsabhängigkeit wird durch eine generierte pseudonatürlichsprachige Frage, die eine negierte Inklusionsabhängigkeit umschreibt, erfragt. Diese Frage ist im Gegensatz zum Beispiel eindeutig, das Beispiel wird aber verwendet, um diese Frage zu veranschaulichen. Abbildung 9.7 zeigt die Erfragung von Inklusionsabhängigkeiten und negierten Inklusionsabhängigkeiten.

Wird diese Frage bejaht, so existiert eine negierte Inklusionsabhängigkeit (Mitarbeiter.Name $\not\subseteq$ Personen.Name), im anderen Fall eine Inklusionsabhängigkeit (Mitarbeiter.Name $\subseteq$ Personen.Name).

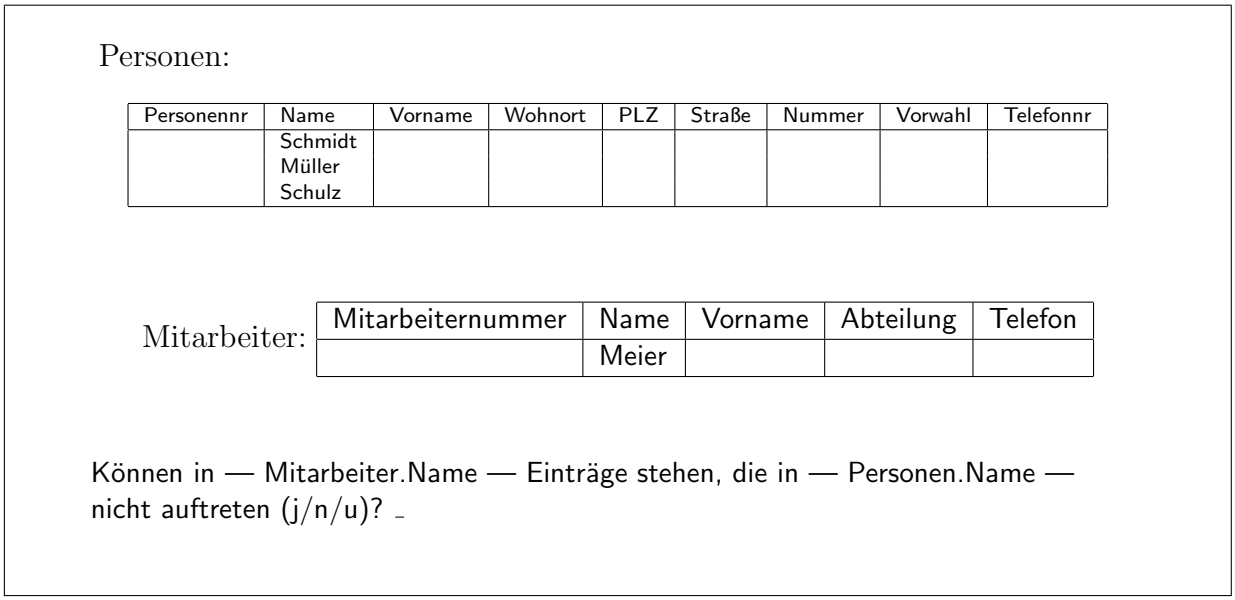


Abbildung 9.7: Erfragung von geltenden und negierten Inklusionsabhängigkeiten

9.4 Exklusionsabhängigkeiten

Die Erfragung einer Exklusionsabhängigkeit $R.X \mid\mid S.Y$ kann einfach durch ein Beispiel dargestellt werden.

Es werden dazu zwei Beispielrelationen r und s generiert, in diesen muß jeweils nur ein Tupel stehen. In $R.X$ und $S.Y$ muß dabei der gleiche Wert auftreten. Abbildung 9.8 zeigt die Erfragung einer Exklusionsabhängigkeit bzw. einer negierten Exklusionsabhängigkeit.

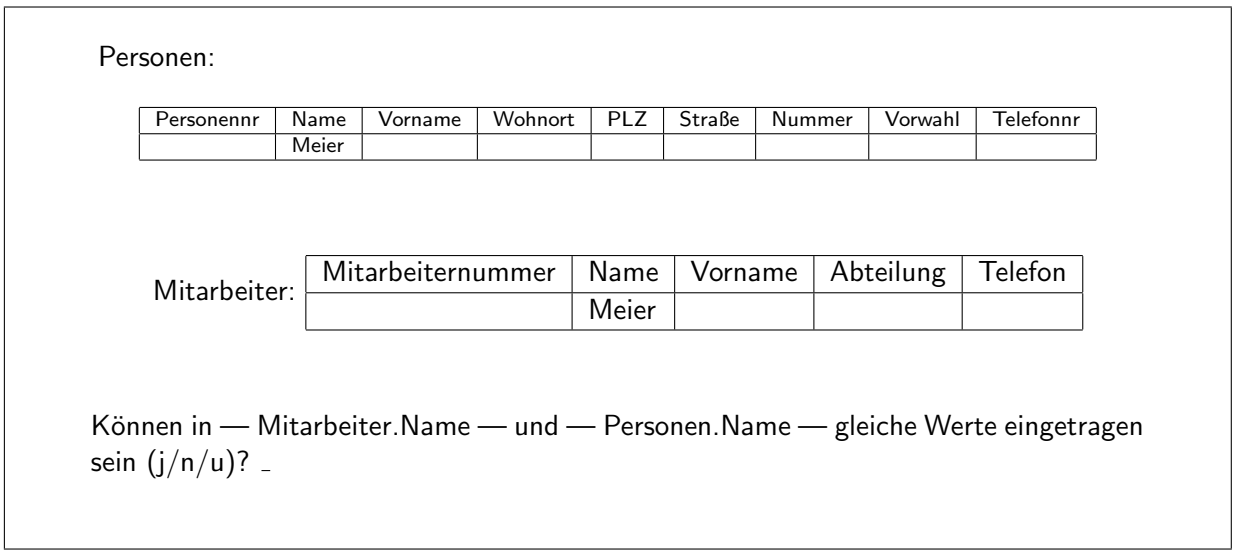


Abbildung 9.8: Erfragung von geltenden und negierten Exklusionsabhängigkeiten

Wird diese Frage bejaht, so gilt die dargestellte negierte Exklusionsabhängigkeit (Mitarbeiter.Name $\not\mid\mid$ Personen.Name), im anderen Fall gilt eine Exklusionsabhängigkeit $R.X \mid\mid S.Y$

(Mitarbeiter.Name || Personen.Name).

9.5 Kardinalitäten

Zur Erfragung von Kardinalitäten zwischen zwei Relationen-Schemata sind Fremdschlüsselbeziehungen notwendig. Dieses soll am Beispiel erläutert werden. Dazu werden die folgenden Relationen-Schemata verwendet:

Studenten:

Matrikelnummer	Name	Vorname	Fachbereich	Semester

Vorlesungsbesuch:

Studentennummer	Fach	Semester

Das Attribut Matrikelnummer ist Schlüssel in dem Relationen-Schema Studenten und das Attribut Studentennummer ist Fremdschlüssel in dem Relationen-Schema Vorlesungsbesuch, es gilt die Inklusionsabhängigkeit Vorlesungsbesuch.Studentennummer $\subseteq$ Studenten.Matrikelnummer. Aufgrund dieses Fremdschlüssels wird die Kardinalität zwischen den beiden Relationen-Schemata bestimmt.

Die Erfragung von Kardinalitäten ist durch zwei Sätze möglich, in denen die Information umschrieben wird. Diese erfragen den Minimumwert und den Maximumwert der Kardinalitäten.

> Wie oft kann ein Eintrag aus — Studenten — maximal
> in — Vorlesungsbelegung — auftreten ? _

> Wie oft muß ein Eintrag aus — Studenten — mindestens
> in — Vorlesungsbelegung — auftreten ? _

Es ist auch durch zwei Beispieldiskussionen möglich, die Kardinalitäten zu bestimmen. Der Maximumwert der Kardinalitäten kann durch ein generiertes Beispiel ermittelt werden, in dem die erste Relation r nur aus einem Tupel besteht. In diesem Tupel ist nur der Wert des Schlüssels belegt. Dieser Wert kommt in der anderen generierten Relation s mehrfach als Fremdschlüssel vor. Dargestellt wird dadurch die Kardinalität $\mathsf{card}(R, S) \neq (\_, 1)$. Durch die verwendete strenge Semantik der Kardinalitäten und die Einschränkung auf die Werte 1 und n für den Maximumwert der Kardinalitäten bedeutet diese Kardinalität: $\mathsf{card}(R, S) = (\_, n)$. Abbildung 9.9 zeigt die Erfragung des Maximumwertes der Kardinalitäten.

Aus der Beantwortung läßt sich ableiten, daß folgende Kardinalitäten gelten:

Bejahung	card(Vorlesungsbelegung, Student) $= (\_, n)$
Verneinung	card(Vorlesungsbelegung, Student) $= (\_, 1)$

Der Minimumwert der Kardinalitäten kann durch ein Beispiel erfragt werden, in dem eine Relation r mit einem Tupel generiert wird, dessen Eintrag bei dem Schlüssel in der Relation s nicht als Fremdschlüssel vorkommt. Wie Inklusionsabhängigkeiten sind auch die Minimumwerte der Kardinalitäten *invariant gegenüber Tupelergänzungen*. Die Darstellung dieser erfolgt in

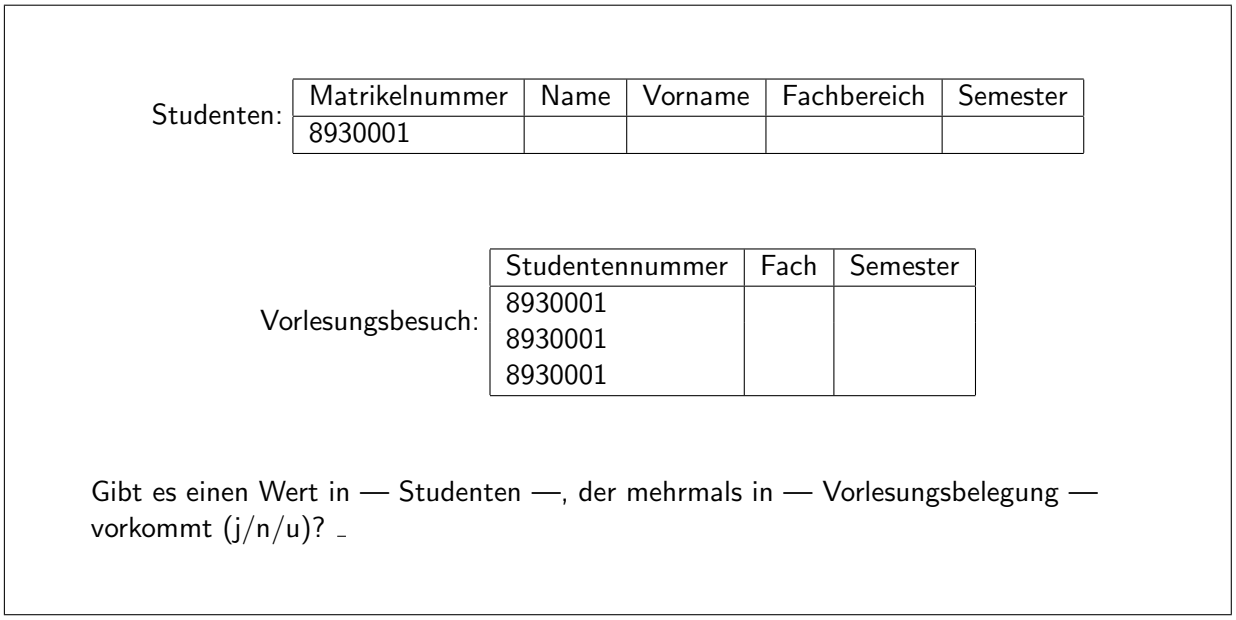


Abbildung 9.9: Erfragung von Maximum-Kardinalitäten

zwei Relationen, die der Benutzer als abgeschlossen ansehen muß, dazu erscheint eine pseudo-natürlichsprachige Frage, die das verdeutlicht.

Das generierte Beispiel stellt die Kardinalität $\mathsf{card}(R, S) \neq (1, \_)$ dar. Durch die verwendete strenge Semantik der Kardinalitäten und die Beschränkung auf die Werte 0 und 1 für den Minimumwert der Kardinalitäten bedeutet das: $\mathsf{card}(R, S) = (0, \_)$. In Abbildung 9.10 wird die Erfragung des Minimumwertes der Kardinalitäten dargestellt.

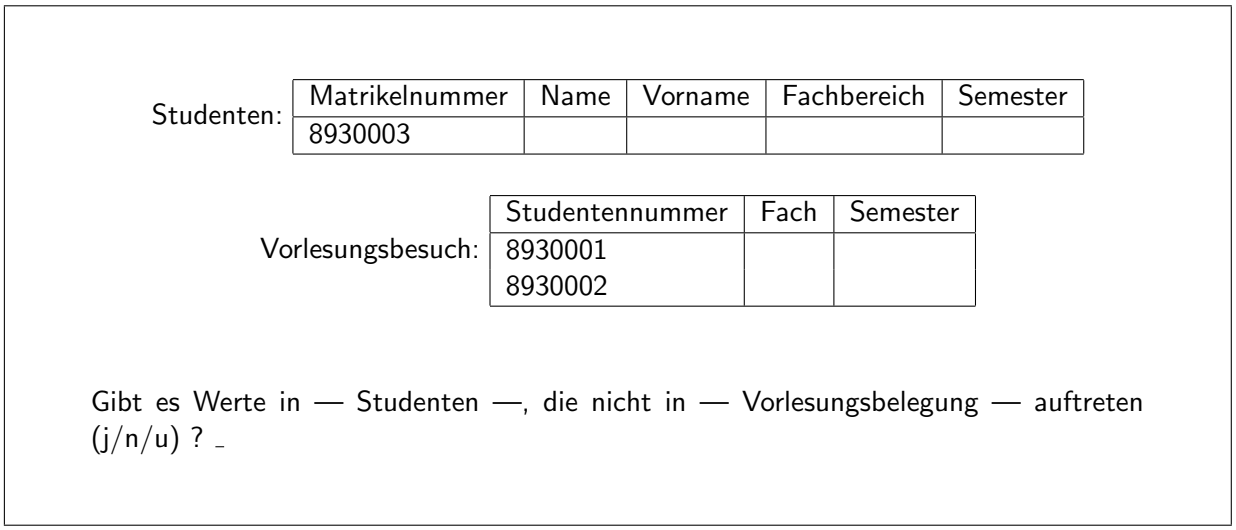


Abbildung 9.10: Erfragung von Minimum-Kardinalitäten

Aus der Beantwortung läßt sich ableiten:

Bejahung	$\mathsf{card}(\text{Vorlesungsbelegung, Student}) = (0, \_)$
Verneinung	$\mathsf{card}(\text{Vorlesungsbelegung, Student}) = (1, \_)$

Aus der Beantwortung der beiden Fragen läßt sich die Kardinalität zwischen zwei Relationen-Schemata ableiten. Die folgende Übersicht zeigt die resultierenden Kardinalitäten:

	Frage 1 $\mathsf{card}(R,S) = (\_, n)$ möglich?		
		ja	nein
Frage 2	ja	$\mathsf{card}(R,S) = (0,n)$	$\mathsf{card}(R,S) = (0,1)$
$\mathsf{card}(R,S) = (0,\_)$ möglich?	nein	$\mathsf{card}(R,S) = (1,n)$	$\mathsf{card}(R,S) = (1,1)$

Man kann also durch die beiden Fragen die Kardinalität ermitteln.

Die in diesem Kapitel gezeigten Möglichkeiten zur Validierung erfüllen den Anspruch der *Verständlichkeit*. Sie erfragen Integritätsbedingungen nicht formal, sondern anhand von Beispielen. Dadurch ermöglichen sie auch Benutzern, die keine formalen Integritätsbedingungen angeben können, Angaben zur Semantik zu machen, aus denen formale Integritätsbedingungen abgeleitet werden können.

Kapitel 10

Übernahme von Integritätsbedingungen in veränderte Datenbank-Schemata

Der Datenbank-Entwurf und die Forward-Engineering-Phase im Re-Engineering von Datenbanken sind *iterative Prozesse*. Es ist möglich, daß in einem Datenbank-Schema bereits Integritätsbedingungen bekannt waren und dieses Datenbank-Schema anschließend verändert werden. Teilweise sind die Änderungen nur geringfügig, z.B. kann ein Ergänzen von Attributen erfolgen, weil Unvollständigkeiten im Datenbank-Schema gefunden wurden.

Bei Änderung eines Datenbank-Schemas möchte man die *Integritätsbedingungen*, die trotz der Änderung weiterhin gültig sind, für dieses veränderte Datenbank-Schema *übernehmen*. Weiterhin möchte man aus Integritätsbedingungen, die aufgrund der Änderung des Datenbank-Schemas nicht übernommen werden können, *sinnvolle Kandidaten* für Integritätsbedingungen ableiten, die möglicherweise in dem veränderten Datenbank-Schema gelten.

Dazu muß zunächst betrachtet werden, welche *Änderungen eines Datenbank-Schemas* berücksichtigt werden müssen.
Die *Attribute* eines Relationen-Schemas können verändert werden, dabei werden *Attributnamen oder Wertebereiche der Attribute* (Typ, Länge eines Attributes oder Anzahl der verschiedenen Werte, die ein Attribut annimmt) geändert.
Ein *Relationen-Schema* ändert sich, wenn der *Name des Relationen-Schemas* verändert wird. Weiterhin erfolgt eine Änderung des Relationen-Schemas, wenn *zugehörige Attribute* verändert werden. Man kann jedoch die Annahme treffen, daß das Ergänzen oder Löschen einzelner Attribute keine Änderung der Bedeutung des Relationen-Schemas nach sich zieht. Deshalb wird ein Relationen-Schema nur dann als verändert betrachtet, wenn sich Attribute ändern, die als Schlüssel ausgewiesen sind oder wenn mehr als ein Drittel der Attribute geändert, gelöscht oder ergänzt werden.
Es wird angenommen, daß die Bedeutung von Attributen und Relationen-Schemata, in denen keine Änderung der Bezeichnungen, Wertebereiche, zugehörigen Attribute, usw. erfolgte, unverändert bleibt und damit auch die bekannten Integritätsbedingungen über diesen Attributen und Relationen-Schemas weiterhin gelten.

In diesem Kapitel wird gezeigt, welche Integritätsbedingungen nach *Ergänzung, Löschen und Verändern von Attributen und Relationen-Schemata* übernommen werden können und welche Kandidaten für Integritätsbedingungen in diesen Fällen abgeleitet werden können.

Zur Übernahme von Integritätsbedingungen nach Änderungen müssen das Datenbank-Schema vor der Änderung (altes Datenbank-Schema), die geltenden Integritätsbedingungen auf diesem Datenbank-Schema und das Datenbank-Schema nach der Änderung (neues Datenbank-Schema) bekannt sein. Aus diesen Informationen lassen sich Integritätsbedingungen des neuen Datenbank-Schemas und Kandidaten für Integritätsbedingungen auf dem neuen Datenbank-Schema ableiten.

Die Übernahme von Integritätsbedingungen und Kandidaten wird in den Abschnitten 10.1 bis 10.6 vorgestellt und jeweils begründet. Diese Übernahmemöglichkeiten werden in Abschnitt 10.7 noch einmal in einer Tabelle zusammengefaßt. Anschließend folgt in Abschnitt 10.8 ein Beispiel, in dem die Übernahme von Integritätsbedingungen nach Verändern des Datenbank-Schemas demonstriert wird.

10.1 Ergänzen von Attributen

Wird in einem Relationen-Schema R ein Attribut A ergänzt, so werden bekannte Integritätsbedingungen des Relationen-Schemas wie folgt behandelt:

- **Schlüssel.**
  Bekannte Schlüssel des Relationen-Schemas R können nicht übernommen werden, da man nicht weiß, ob die Schlüssel auch das neu hinzugekommene Attribut A identifizieren. Die Schlüssel werden aber als Kandidaten in das veränderte Relationen-Schema übernommen und eine erneute Validierung dieser Schlüssel wird versucht.

- **Negierte Schlüssel.**
  Bekannte negierte Schlüssel eines Relationen-Schemas können übernommen werden, wenn Attribute im Relationen-Schema ergänzt werden. Da negierte Schlüssel nicht alle Attribute des Relationen-Schemas identifizieren und deshalb gilt: $X \not\rightarrow Y_1$ oder $X \not\rightarrow Y_2$ .. oder $X \not\rightarrow Y_n$, Y_i - Attribute von R, weiß man, daß diese negierten funktionalen Abhängigkeiten auch nach Ergänzen eines Attributes erhalten bleiben.

- **Funktionale und negierte funktionale Abhängigkeiten.**
  Zusätzliche Attribute beeinflussen die Geltung von funktionalen und negierten funktionalen Abhängigkeiten nicht, diese können also ohne Validierung übernommen werden.

- **Inklusionsabhängigkeiten, negierte Inklusionsabhängigkeiten, Exklusionsabhängigkeiten, negierte Exklusionsabhängigkeiten.**
  Zusätzliche Attribute eines Relationen-Schemas R beeinflussen die Geltung von Inklusions- bzw. Exklusionsabhängigkeiten zwischen zwei Relationen-Schemata R und S nicht, diese Integritätsbedingungen können also ohne Validierung übernommen werden.

- **Kardinalitäten.**
  Die Kardinalitäten zwischen zwei Relationen-Schemata bleiben durch das Ergänzen von Attributen unverändert, bekannte Kardinalitäten können also in das veränderte Datenbank-Schema übernommen werden.

Das Ergänzen von Attributen in Relationen-Schemata beeinflußt also nur das Gelten von Schlüsseln, diese müssen für die Relationen-Schemata, in denen Attribute ergänzt werden, neu validiert werden.

10.2 Löschen von Attributen

Beim Löschen eines Attributes A aus einem Relationen-Schema R sind für die Integritätsbedingungen eines Relationen-Schemas zwei Fälle zu betrachten.

Nicht über A definierte Integritätsbedingungen

Beim Löschen eines Attributes A aus einem Relationen-Schema R können folgende Integritätsbedingungen übernommen werden, sofern sie nicht über dem Attribut A definiert sind:

- **Schlüssel.**
  Bekannte Schlüssel X können ohne Validierung übernommen werden, sofern das gelöschte Attribut A nicht in X auftritt.
- **Negierte Schlüssel.**
  Bekannte negierte Schlüssel X sagen aus, daß mindestens eine negierte funktionale Abhängigkeit $X \not\rightarrow Y$, $\forall\ Y$ - Attribute in R gilt. Da diese negierte funktionale Abhängigkeit zu dem gelöschten Attribut A bestehen kann, muß der negierte Schlüssel erneut validiert werden. Es ist möglich, daß dieser negierte Schlüssel für das verkleinerte Relationen-Schema einen Schlüssel darstellt.
- **Funktionale und negierte funktionale Abhängigkeiten.**
  Funktionale und negierte funktionale Abhängigkeiten können ohne Validierung übernommen werden, sofern sie nicht auf dem gelöschten Attribut A definiert sind.
- **Inklusionsabhängigkeiten, negierte Inklusionsabhängigkeiten, Exklusionsabhängigkeiten, negierte Exklusionsabhängigkeiten.**
  Gelöschte Attribute eines Relationen-Schemas R beeinflussen die Geltung von Inklusions- bzw. Exklusionsabhängigkeiten zwischen zwei Relationen-Schemata R und S nicht, diese können also ohne Validierung übernommen werden.
- **Kardinalitäten.**
  Kardinalitäten sind semantische Eigenschaften, die zwischen zwei Relationen-Schemata definiert sind. Diese widerspiegeln sich auf den Fremdschlüsselbeziehungen zwischen den Relationen-Schemata. Werden Attribute, über denen die Kardinalitäten nicht definiert waren, gelöscht, so hat das keinen Einfluß auf die Kardinalitäten, diese können also übernommen werden.

Über A definierte Integritätsbedingungen

Die Integritätsbedingungen, die über dem gelöschten Attribut definiert waren, können generell *nicht übernommen* werden. Ausnahmen, in der eine Übernahme von Kandidaten versucht wird, sind folgende:

- **Schlüssel.**
  Ein Schlüssel X, in dem ein Attribut gelöscht wurde, kann nicht übernommen werden. Da jedoch ein Schlüssel im Relationen-Schema vorhanden sein muß, um die Tupel des Relationen-Schemas zu identifizieren, versucht man den Restschlüssel $X - A$ als Schlüsselkandidaten zu validieren.

- **Kardinalitäten.**
  Kardinalitäten sind semantische Eigenschaften, die zwischen zwei Relationen-Schemata definiert sind. Diese werden auf Fremdschlüsselbeziehungen zwischen den Relationen-Schemata dargestellt. Werden Attribute, über denen die Kardinalitäten definiert waren (also Schlüsselattribute oder Fremdschlüsselattribute eines Relationen-Schemas) gelöscht, so wird versucht, neue Fremdschlüsselbeziehungen zwischen diesen Relationen-Schemata zu finden, um so die bekannten Kardinalitäten über diesen zu validieren.
  Kardinalitäten werden also nach dem Löschen von Attributen in modifizierter Form als Kandidaten übernommen.

Die Übernahme von Integritätsbedingungen, die über gelöschten Attributen definiert waren, ist also generell nicht möglich. Es wird aber versucht, Schlüssel und Kardinalitäten als Kandidaten zu übernehmen.

10.3 Änderung von Attributen

Die Änderung eines Attributes A in einem Relationen-Schema R in ein Attribut A' kann als *Löschen* des Attributes A und *Ergänzen* des Attributes A' gesehen werden.

Dabei werden ebenso wie beim Löschen von Attributen zwei Fälle unterschieden.

Nicht über A definierte Integritätsbedingungen

Integritätsbedingungen, in denen das veränderte Attribut nicht vorkommt, können als geltende Integritätsbedingungen ohne Validierung oder als Kandidaten für Integritätsbedingungen, die validiert werden müssen, übernommen werden. Die folgende Übersicht zeigt, wie das erfolgen kann.

- **Schlüssel.**
  Bekannte Schlüssel müssen neu validiert werden, da man nicht weiß, ob diese Schlüssel auch das veränderte Attribut identifizieren. Sie werden also als Kandidaten übernommen.

- **Negierte Schlüssel.**
  Bekannte negierte Schlüssel X sagen aus, daß mindestens eine negierte funktionale Abhängigkeit $X \not\rightarrow Y,\ \forall\ Y$ - Attribute in R gilt. Da diese negierte funktionale Abhängigkeit zu dem geänderten Attribut A bestehen kann, muß der negierte Schlüssel erneut validiert werden. Dieser kann also nur als Kandidat für einen negierten Schlüssel übernommen werden.

- **Funktionale und negierte funktionale Abhängigkeiten.**
  Funktionale und negierte funktionale Abhängigkeiten können ohne Validierung übernommen werden.

- **Inklusionsabhängigkeiten, negierte Inklusionsabhängigkeiten, Exklusionsabhängigkeiten, negierte Exklusionsabhängigkeiten.**
  Geänderte Attribute eines Relationen-Schemas R beeinflussen das Gelten von Inklusions- und Exklusionsabhängigkeiten zwischen zwei Knoten R und S nicht, diese können also ohne Validierung übernommen werden.

- **Kardinalitäten.**
  Werden Attribute, über denen die Kardinalitäten nicht definiert waren, geändert, so hat das keinen Einfluß auf die Kardinalitäten, diese können also übernommen werden.

Über A definierte Integritätsbedingungen

Alle Integritätsbedingungen (Schlüssel, negierte Schlüssel, funktionale Abhängigkeiten, negierte funktionale Abhängigkeiten, Inklusionsabhängigkeiten, negierte Inklusionsabhängigkeiten, Exklusionsabhängigkeiten, negierte Exklusionsabhängigkeiten, Kardinalitäten), in denen das geänderte Attribut A vorkommt, können nicht übernommen werden. Da man aber einen Zusammenhang zwischen dem vorher vorhandenen Attribut A und dem geänderten Attribut A' vermutet, werden alle diese Integritätsbedingungen als Kandidaten für Integritätsbedingungen neu validiert.

- **Schlüssel.**
  Bekannte Schlüssel, in denen ein Attribut verändert wurde, müssen neu validiert werden, da man nicht weiß, ob die Schlüsseleigenschaft für diesen veränderten Schlüssel gewährleistet ist. Sie werden als Kandidaten übernommen.

- **Negierte Schlüssel.**
  Bekannte negierte Schlüssel in denen ein Attribut verändert wurde, müssen neu validiert werden, da man nicht weiß, ob das veränderte Attribut weiterhin die gleiche Bedeutung hat. Es erfolgt eine Übernahme der negierten Schlüssel als Kandidaten.

- **Funktionale und negierte funktionale Abhängigkeiten.**
  Eine funktionale bzw. negierte funktionale Abhängigkeit, in der ein Attribut verändert wurde, muß ebenfalls neu validiert werden.

- **Inklusionsabhängigkeiten, negierte Inklusionsabhängigkeiten, Exklusionsabhängigkeiten, negierte Exklusionsabhängigkeit.**
  Geänderte Attribute einer Inklusions- oder Exklusionsabhängigkeiten bewirken, daß diese Abhängigkeit ebenfalls nur als Kandidat übernommen werden kann und eine Validierung erfolgen muß.

- **Kardinalitäten.**
  Nach der Änderung von Schlüsselattributen oder Fremdschlüsselattributen, über denen Kardinalitäten definiert sind, erfolgt ebenfalls eine Übernahme der Kardinalitäten als Kandidaten und eine erneute Validierung.

Bei der Änderung von Attributen werden also die Integritätsbedingungen, in denen das geänderte Attribut A nicht enthalten ist, übernommen (Ausnahme: Schlüssel und negierte Schlüssel). Alle Integritätsbedingungen, in denen das geänderte Attribut A auftritt, können nur als Kandidaten übernommen werden.

10.4 Ergänzen von Relationen-Schemata

Beim Ergänzen von Relationen-Schemata in einem Datenbank-Schema ändern sich die bereits bekannten Integritätsbedingungen, die über den anderen Relationen-Schemata des Datenbank-Schemas definiert sind, nicht.

10.5 Löschen von Relationen-Schemata

Beim Löschen eines Relationen-Schemas R aus einem Datenbank-Schema sind zwei Fälle der Übernahme von Integritätsbedingungen zu unterscheiden.

Nicht über R definierte Integritätsbedingungen

Integritätsbedingungen, die über anderen Relationen-Schemata der Datenbank definiert sind, bleiben von dem Löschen unbeeinflußt, diese können also auch in das neue Datenbank-Schema als Integritätsbedingungen übernommen werden.

Über R definierte Integritätsbedingungen

Beim Löschen von Relationen-Schemata können keine Integritätsbedingungen, die über diesem Relationen-Schema definiert sind, übernommen werden.

10.6 Änderung von Relationen-Schemata

Beim Ändern eines Relationen-Schemas R in R' werden ebenfalls zwei Fälle der Übernahme von Integritätsbedingungen unterschieden.

Nicht über R definierte Integritätsbedingungen

Beim Ändern eines Relationen-Schemas ändern sich die Integritätsbedingungen, die über anderen Relationen-Schemata definiert sind, nicht.

Über R definierte Integritätsbedingungen

Bei der Übernahme von Integritätsbedingungen, die über einem geänderten Relationen-Schema definiert sind, ist zu beachten, daß nur solche Übernahmen erfolgen können, die beim Löschen und Ergänzen eines Relationen-Schemas erfolgen können. Damit können keine Integritätsbedingungen, die über den Relationen-Schemas R definiert sind, übernommen werden. Da man jedoch einen Zusammenhang zwischen dem alten Relationen-Schema R und dem neuen Relationen-Schema R' vermutet, werden alle bisher bekannten Integritätsbedingungen (Schlüssel, negierte Schlüssel, funktionale und negierte funktionale Abhängigkeiten, Inklusionsabhängigkeiten, negierte Inklusionsabhängigkeiten, Exklusionsabhängigkeiten, negierte Exklusionsabhängigkeiten, Kardinalitäten) als Kandidaten übernommen.

10.7 Zusammenfassung der Übernahmemöglichkeiten für Integritätsbedingungen

In diesem Abschnitt soll die Übernahme von Integritätsbedingungen nach Änderung eines Datenbank-Schemas anhand einer Tabelle zusammengefaßt werden.

Änderung der Datenbank		KEY	NKEY	FD	NFD	ID	NID	ED	NED	Kard
Ergänzen von Attributen		K	I	I	I	I	I	I	I	I
Löschen von Attributen	$A \not\varepsilon \sigma$	I	K	I	I	I	I	I	I	I
	$A \varepsilon \sigma$	K	-	-	-	-	-	-	-	K
Ändern von Attributen	$A \not\varepsilon \sigma$	K	K	I	I	I	I	I	I	I
	$A \varepsilon \sigma$	K	K	K	K	K	K	K	K	K
Ergänzen von Relationen-Schemata		I	I	I	I	I	I	I	I	I
Löschen von Relationen-Schemata	$R \not\varepsilon \sigma$	I	I	I	I	I	I	I	I	I
	$R \varepsilon \sigma$	-	-	-	-	-	-	-	-	-
Ändern von Relationen-Schemata	$R \not\varepsilon \sigma$	I	I	I	I	I	I	I	I	I
	$R \varepsilon \sigma$	K	K	K	K	K	K	K	K	K

$A\ \varepsilon\sigma$ - die Integritätsbedingung σ ist über dem Attribut A definiert
$A \not\varepsilon \sigma$ - die Integritätsbedingung σ ist nicht über dem Attribut A definiert
$R\ \varepsilon\sigma$ - die Integritätsbedingung σ ist über der Relationen-Schemata R definiert
$R \not\varepsilon \sigma$ - die Integritätsbedingung σ ist nicht über der Relationen-Schemata R definiert

Die Einträge in die Tabelle haben folgende Bedeutung:

- I bedeutet, daß eine Integritätsbedingung übernommen wird,
- K bedeutet, daß eine Integritätsbedingung als Kandidat übernommen wird und
- \- bedeutet, daß eine Integritätsbedingung nicht übernommen wird.

Für eine Integritätsbedingung können mehrere Änderungen der Datenbank relevant sein, die Einfluß auf die Übernahme der Integritätsbedingungen in die geänderte Datenbank haben. Dabei muß der ungünstigste Fall der Änderungen der Datenbank berücksichtigt werden.
Gilt z.B. eine funktionale Abhängigkeit $X \longrightarrow Y$ über einem Relationen-Schema R und es erfolgt eine Änderung des Relationen-Schemas R und das Löschen eines Attributes A des Relationen-Schemas R, mit $A \notin X$ und $A \notin Y$. Aufgrund des Löschens eines Attributes, über dem die funktionale Abhängigkeit nicht definiert ist, kann die funktionale Abhängigkeit übernommen werden (siehe Tabelle). Aufgrund der Änderung des Relationen-Schemas, über dem die funktionale Abhängigkeit definiert ist, kann nur die Übernahme als Kandidat erfolgen (siehe Tabelle). Deshalb ist nur die Übernahme der funktionalen Abhängigkeit als Kandidat möglich.

10.8 Beispiel

Die Übernahme von Integritätsbedingungen nach Änderung eines Datenbank-Schemas soll an einem umfangreichen Beispiel gezeigt werden. Dieses Beispiel beinhaltet das Ergänzen, Löschen und Ändern von Attributen, sowie das Ändern von Relationen-Schemata.

Folgendes Datenbank-Schema soll dabei zugrunde gelegt werden:

Datenbank-Schema Universität.
Studenten = (Name, Vorname, Studiennummer)
studiert = (Studentennummer, Fachbereich)
Professor = (Personalnummer, Name, Vorname, Titel, Adresse)
arbeitet = (Professor, Fachbereich, Lehrstuhlbezeichnung)
Fachbereich = (Bezeichnung, Adresse, Telefon)

Auf diesem Datenbank-Schema sind folgende Integritätsbedingungen bekannt:

Schlüssel.
Studiennummer Schlüssel von Studenten
Personalnummer Schlüssel von Professor
Studentennummer,Fachbereich Schlüssel von studiert
Professor Schlüssel von arbeitet
Bezeichnung Schlüssel von Fachbereich

Inklusionsabhängigkeiten.
studiert.Studentennummer $\subseteq$ Studenten.Studiennummer
studiert.Fachbereich $\subseteq$ Fachbereich.Bezeichnung
arbeitet.Professor $\subseteq$ Professor.Personalnummer
arbeitet.Fachbereich $\subseteq$ Fachbereich.Bezeichnung

Auf diesem Datenbank-Schema sollen folgende Änderungen erfolgen:

Ändern von Relationen-Schemata.
Das Relationen-Schema studiert wird in Hauptfach geändert.

Ergänzen von Attributen.
Das Attribut Geburtsdatum wird in dem Relationen-Schema Studenten ergänzt.
Das Attribut Vorwahl wird in dem Relationen-Schema Fachbereich ergänzt.
Das Attribut Fachsemester wird in dem Relationen-Schema Hauptfach ergänzt.

Löschen von Attributen.
In dem Relationen-Schema Professor wird das Attribut Adresse gelöscht.

Ändern von Attributen.
Das Attribut Telefon in dem Relationen-Schema Fachbereich wird in Rufnummer verändert.

Durch diese Änderungen ergibt sich folgendes Datenbank-Schema:

Datenbank-Schema Universität'.
Studenten = (Name, Vorname, Studiennummer, Geburtsdatum)
Hauptfach = (Studentennummer, Fachbereich, Fachsemester)
Professor = (Personalnummer, Name, Vorname, Titel)
arbeitet = (Professor, Fachbereich, Lehrstuhlbezeichnung)
Fachbereich = (Bezeichnung, Adresse, Vorwahl, Rufnummer)

Folgende Integritätsbedingungen können in dieses Datenbank-Schema übernommen werden:

Schlüssel.
Name, Vorname Schlüssel von Professor.
Professor Schlüssel von arbeitet.

Inklusionsabhängigkeiten.
arbeitet.Professor $\subseteq$ Professor.Personalnummer
arbeitet.Fachbereich $\subseteq$ Fachbereich.Bezeichnung

Begründung der Übernahme der Integritätsbedingungen. Die Schlüssel der Relationen-Schemata Professor und arbeitet können übernommen werden, da in diesen Relationen-Schemata keine Ergänzung oder Änderung von Attributen erfolgte. Es erfolgt auch keine Änderung der betreffenden Relationen-Schemata. In beiden Relationen-Schemata wurde ein Attribut gelöscht, das nicht im Schlüssel enthalten ist, das beeinträchtigt die Schlüsseleigenschaft nicht.
Die Inklusionsabhängigkeiten zwischen den Relationen-Schemata arbeitet und Professor, sowie arbeitet und Fachbereich können übernommen werden, da sich die Relationen-Schemata nicht verändert haben und die Attribute, über denen diese Inklusionsabhängigkeiten definiert waren, ebenfalls unverändert blieben.

Weiterhin können folgende Kandidaten für Integritätsbedingungen übernommen werden:

Kandidaten für Schlüssel.
Studiennummer Schlüssel von Studenten
Studentennummer,Fachbereich Schlüssel von Hauptfach
Bezeichnung Schlüssel von Fachbereich

Kandidaten für Inklusionsabhängigkeiten.
Hauptfach.Studentennummer $\subseteq$ Studenten.Studiennummer
Hauptfach.Fachbereich $\subseteq$ Fachbereich.Bezeichnung

Begründung der Übernahme der Kandidaten. In dem Relationen-Schema Studenten, Hauptfach und Fachbereich wurden Attribute ergänzt bzw. verändert, die Schlüssel dieser Relationen-Schemata müssen deshalb neu validiert werden. Sie werden deshalb als Kandidaten für Schlüssel in das veränderte Datenbank-Schema übernommen.
Das Relationen-Schema studiert wurde in Hauptfach geändert, deshalb müssen die Inklusionsabhängigkeiten des Relationen-Schemas studiert für das veränderte Relationen-Schema noch einmal validiert werden, es erfolgt auch hier eine Übernahme als Kandidaten.

10.9 Anwendung der Übernahme von Integritätsbedingungen

Die Übernahme von bekannten Integritätsbedingungen ist nach der Veränderung von Datenbank-Schemata notwendig, um eine mehrfache Eingabe von identischen Integritätsbedingungen zu vermeiden. Veränderungen von Datenbank-Schemata treten im Datenbank-Entwurf

und in der Forward-Engineering Phase des Re-Engineerings von Datenbanken häufig auf, da es sich hierbei um iterative Prozesse handelt. Durch eine Übernahme von bekannter Semantik (soweit das möglich ist), wird das *Prinzip der Redundanzfreiheit der notwendigen Angaben* vom Benutzer gewahrt.

Einige Integritätsbedingungen bleiben von den Änderungen des Datenbank-Schemas unbeeinflußt, diese können direkt übernommen werden.
Einige Integritätsbedingungen, die nicht übernommen werden können, werden als Kandidaten für Integritätsbedingungen übernommen. Für das veränderte Datenbank-Schema erfolgt eine Semantikakquisition, bei der diese Kandidaten für Integritätsbedingungen validiert werden.
Beim Löschen von Attributen und Relationen-Schemata können die zugehörigen Integritätsbedingungen nicht übernommen werden.

Diese Übernahme der Semantik erfolgt unter der Annahme, daß Attribute und Relationen-Schemata, bei denen keine Änderung der Bezeichnungen oder Wertebereiche auftritt, in ihrer Bedeutung unverändert bleiben. Ist diese Annahme falsch, interpretiert also ein Benutzer die Bedeutung des Datenbank-Schemas anders als das beim Entwurf erfolgte, so kann das die Übernahme falscher Integritätsbedingungen verursachen. Dieser Fall kann nicht erkannt werden, da dabei nur die Interpretation der Datenbank-Strukturen verändert wird, es aber keine äußeren Anzeichen dafür gibt. Diese Änderung der Interpretation kann zu jedem Zeitpunkt erfolgen. Als zusätzliche Absicherung für diesen Fall, empfiehlt sich die Überprüfung der abgeleiteten Integritätsbedingungen nach den in Kapitel 4 beschriebenen Methoden.

Kapitel 11

Gesamtalgorithmus zur Unterstützung der Semantikakquisition

In den bisherigen Kapiteln wurden verschiedene Methoden vorgestellt, mit denen *Integritätsbedingungen* und *Kandidaten für Integritätsbedingungen* hergeleitet werden können. Teilweise wurde bereits angegeben, wie die einzelnen vorgestellten Methoden sinnvoll miteinander kombiniert werden können.
In diesem Kapitel soll noch einmal zusammengefaßt werden, wie diese Methoden miteinander verbunden werden, um auf effiziente und benutzerfreundliche Weise die Integritätsbedingungen eines Datenbank-Schemas mit dem Benutzer zu diskutieren. Dabei wird begründet, welche Methoden vor welchen anderen Methoden angewendet werden sollten, um optimale Ergebnisse bei der Semantikakquisition zu erreichen.

Es wurden in dieser Arbeit Methoden vorgestellt, mit denen *Integritätsbedingungen abgeleitet* werden können, ohne daß Interaktionen mit dem Benutzer erforderlich sind. Mit einigen Methoden wurden *Kandidaten für Integritätsbedingungen abgeleitet*, die aber noch mit dem Benutzer diskutiert werden müssen. Weiterhin gab es Methoden, mit denen *Kandidaten für Integritätsbedingungen* mit dem Benutzer diskutiert und dadurch *validiert* werden.
Diese drei Gruppen von Methoden werden in Abschnitt 11.1 - 11.3 noch einmal kurz zusammengefaßt. In Abschnitt 11.4 folgt ein Vorschlag für einen Gesamtalgorithmus.

11.1 Ermittlung von Integritätsbedingungen ohne Interaktion mit dem Benutzer

Zu Beginn der Semantikakquisition sollten solche Aktionen ausgeführt werden, bei denen Integritätsbedingungen ohne Interaktion mit dem Benutzer abgeleitet werden können. Diese Methoden müssen vor der Diskussion unbekannter Integritätsbedingungen ausgeführt werden, damit Informationen, die aus anderen verfügbaren Informationen direkt ableitbar sind, nicht mit dem Benutzer diskutiert werden müssen. Im folgenden werden diese Methoden aufgezählt.

Anpassung der Semantik an veränderte Datenbanken. Da der Datenbank-Entwurf ein iterativer Prozeß ist und auch das Re-Engineering von Datenbanken einen iterativen Entwurfsschritt einschließt, kann es vorkommen, daß die Semantikakquisition für ein Datenbank-Schema

mehrfach erfolgt und daß das Datenbank-Schema in der Zwischenzeit verändert wurde.
Deshalb muß bei der Semantikakquisition für ein Datenbank-Schema überprüft werden, ob bereits Integritätsbedingungen für dieses Datenbank-Schema ermittelt wurden. Ist das der Fall und das Datenbank-Schema wurde nicht verändert, so werden bereits ermittelte Integritätsbedingungen übernommen. Wurde das Datenbank-Schema verändert, so müssen die bekannten Integritätsbedingungen an das veränderte Datenbank-Schema angepaßt werden. Das Vorgehen dazu wurde in **Kapitel 10** beschrieben, benötigt werden dabei das alte und das neue Datenbank-Schema, sowie die Integritätsbedingungen, die in dem alten Datenbank-Schema bekannt waren. Abgeleitet werden Integritätsbedingungen und Kandidaten für Integritätsbedingungen für das geänderte Datenbank-Schema.

Auswertung von Daten und Datenbank-Statistik. Vorhandene Daten und Datenbank-Statistik-Angaben können zur Ermittlung von Integritätsbedingungen ausgewertet werden. Aus diesen können *negierte Integritätsbedingungen* abgeleitet werden. Welche Integritätsbedingungen aus welchen Informationen resultieren, wurde in **Kapitel 3** gezeigt.

Auswertung von formalen Integritätsbedingungen. Im Datenbank-Entwurf und im Reverse-Engineering von Datenbanken können explizite Angaben zu formalen Integritätsbedingungen bekannt sein. Diese formalen Integritätsbedingungen sollen soweit möglich überprüft werden. In **Kapitel 4** wurde gezeigt, aus welchen Gründen formale Integritätsbedingungen bekannt sein können und wie die Überprüfung solcher Integritätsbedingungen erfolgen kann.
Dabei soll auch überprüft werden, ob die formal spezifizierten Integritätsbedingungen konform mit dem Daten und den Datenbank-Statistik-Informationen ist. Diese Überprüfung kann erst erfolgen, wenn sowohl die Daten und Datenbank-Statistik-Informationen vorhanden sind als auch die explizite Angaben zur Semantik bekannt sind. Die Diskussion von evt. auftretenden Konflikten mit dem Benutzer soll möglichst direkt im Anschluß an das Eingeben von expliziten Integritätsbedingungen geschehen, deshalb muß zuerst die Auswertung von Daten und Datenbank-Statistik-Informationen erfolgen, erst dann kann das Einlesen und Überprüfen von formalen Integritätsbedingungen durchgeführt werden.

Mit diesen Methoden werden einige geltende und negierte Integritätsbedingungen des Datenbank-Schemas ermittelt. In Abbildung 11.1 werden diese Integritätsbedingungen in der Menge aller konstruierbaren Integritätsbedingungen des Datenbank-Schemas dargestellt.

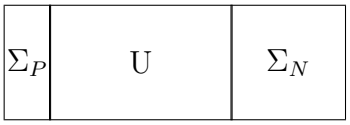


Abbildung 11.1: Integritätsbedingungen, die ohne Interaktionen mit dem Benutzer ermittelt werden können

11.2 Ermittlung von Kandidaten

Mit den in Abschnitt 11.1 angegebenen Methoden sind keine vollständigen Angaben zu den geltenden Integritätsbedingungen zu finden. Es wird nach Ausführung der Methoden, bei denen keine Diskussion der Integritätsbedingungen mit dem Benutzer erfolgt, in der Regel noch sehr viele unbekannte Integritätsbedingungen geben. Diese unbekannten Integritätsbedingungen sollen mit dem Benutzer diskutiert werden. Damit hierbei eine intelligente, nachvollziehbare und effiziente Vorgehensweise erreicht werden kann, ist ein Vorabschätzung der unbekannten Integritätsbedingungen erforderlich. Dazu werden plausible Kandidaten für Integritätsbedingungen ermittelt.

Es wurden in der Arbeit verschiedene Methoden vorgestellt, die sinnvolle Kandidaten für Integritätsbedingungen ermitteln. Diese werden im folgenden noch einmal kurz dargestellt.

Anpassung der Semantik an veränderte Datenbanken. Bei der Übernahme von bereits bekannten Integritätsbedingungen in veränderte Datenbanken (**Kapitel 10**) werden neben Integritätsbedingungen auch sinnvolle und plausible Kandidaten für Integritätsbedingungen übernommen.

Anwendung der Metainformationen. Es können mit Hilfe von Heuristiken plausible Kandidaten für Metainformationen abgeleitet werden. Es werden dafür inhaltliche Cluster, unabhängige Einheiten und Kernrelationen bestimmt. In **Kapitel 7** wird gezeigt, wie das erfolgen kann. Die Metainformationen können bei der Ableitung von Kandidaten durch Heuristiken und Festlegung der Dialogreihenfolge eingesetzt werden. Deshalb werden die Metainformationen vor dem Aufruf dieser Methoden ausgewertet.
Zur Bestimmung der Metainformationen werden bekannte Integritätsbedingungen ausgewertet, deshalb ist es nicht möglich, können diese Metainformationen nur einmal für die Semantikakquisition zu ermitteln. Gab es große Änderungen in der Menge der bekannten Integritätsbedingungen, so müssen auch die Metainformationen über das Datenbank-Schema neu bestimmt werden.

Ableitung von Kandidaten durch Heuristiken. Mit den in **Kapitel 5** gezeigten Heuristiken werden weitere plausible Kandidaten für Integritätsbedingungen abgeschätzt. Dazu werden strukturelle Eigenschaften des Datenbank-Schemas, Integritätsbedingungen, Beispieldaten, Transaktionen und Metainformationen verwendet.
Mit diesen Methode können Kandidaten für Integritätsbedingungen, sowie eine Plausibilität, mit der diese Kandidaten gelten, ermittelt werden.
Die Abschätzung der unbekannten Integritätsbedingungen wertet u.a. die Menge der bereits bekannten Integritätsbedingungen aus. Deshalb muß auch die Ermittlung von Kandidaten mehrfach während der Semantikakquisition vorgenommen werden. Nach großen Änderungen in der Menge der bekannten Integritätsbedingungen, muß die Abschätzungen der Kandidaten für Integritätsbedingungen neu vorgenommen werden.

Suche nach gleichen oder ähnlichen Datenbank-Schemata. Es wird nach Datenbankteilen gesucht, die ähnlich oder gleich dem Datenbank-Schema sind, für das die Integritätsbedingungen ermittelt werden sollen. Sind in ähnlichen Datenbank-Schemata bereits Integritäts-

bedingungen bekannt, so können diese als Kandidaten in das aktuelle Datenbank-Schema übernommen werden. In **Kapitel 6** wurde das Vorgehen dazu gezeigt.

Abbildung 11.2 zeigt, welche Informationen über die Semantik einer Datenbank nach Auswertung der in 11.1 und 11.2 beschriebenen Methoden vorliegen.

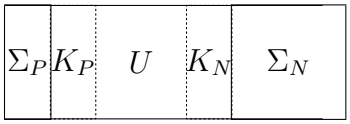


Abbildung 11.2: Semantikinformationen der Datenbank nach der Abschätzung von Kandidaten

Nach Ausführung dieser Methoden werden also neben den bereits bekannten Integritätsbedingungen sinnvolle und plausible Kandidaten für geltende und negierte Integritätsbedingungen in der Menge der noch unbekannten Integritätsbedingungen ermittelt.

11.3 Ermittlung von Integritätsbedingungen durch Diskussion mit dem Benutzer

Die ermittelten Kandidaten für Integritätsbedingungen werden mit dem Benutzer zur *Validierung* diskutiert. Es ist deshalb vor der Validierung erforderlich, die ableitbaren Kandidaten zu ermitteln. Abbildung 11.3 stellt dar, welche Auswirkungen die Validierung auf die Menge der Integritätsbedingungen hat.

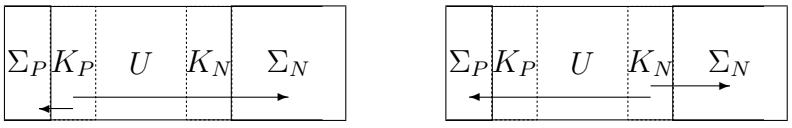


Abbildung 11.3: Validierung von Kandidaten für geltende und negierte Integritätsbedingungen

Kandidaten für geltende oder negierte Integritätsbedingungen werden diskutiert und dadurch entweder als geltende oder negierte Integritätsbedingungen erkannt.

Die Reihenfolge der Diskussion von Integritätsbedingungen soll bestimmte Eigenschaften erfüllen. Deshalb wird eine Methode zur Bestimmung der Dialogreihenfolge verwendet.

Festlegung der Dialogreihenfolge. Für die Menge der Kandidaten wird die Reihenfolge des Dialoges zur Validierung festgelegt. Im **Kapitel 8** wurde beschrieben, wie die Reihenfolge

des Dialoges nach den Kriterien Effizienz der Erfragung, Erfragung der wichtigsten Integritätsbedingungen und Nachvollziehbarkeit des Dialoges festgelegt wird. Aufgrund dieser Kriterien wird jeweils der nächste zu validierende Kandidat ausgewählt.

Validierung von Kandidaten. Zur Validierung der Kandidaten wird der Beispielzugang verwendet, der in **Kapitel 9** beschrieben wurde. Dabei wird ein ausgewählter Kandidat für eine Integritätsbedingung im Beispiel dargestellt und diskutiert. Im Ergebnis der Beispieldiskussion kann man eine geltende oder negierte Integritätsbedingung ableiten.

Im Ergebnis der Validierung der Kandidaten können alle unbekannten Integritätsbedingungen untersucht worden sein, in diesem Fall ist die Semantikakquisition *vollständig*, in Abbildung 11.4 wird das Ergebnis der Semantikakquisition für diesen Fall gezeigt.

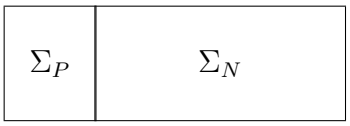


Abbildung 11.4: Vollständige Semantikakquisition

Die Validierung der Kandidaten kann auch unvollständig erfolgen, in großen Datenbank-Schemata ist eine vollständige Validierung von allen konstruierbaren Integritätsbedingungen meist nicht möglich. Abbildung 11.5 zeigt die Menge der Integritätsbedingungen eines Datenbank-Schemas für diesen Fall.

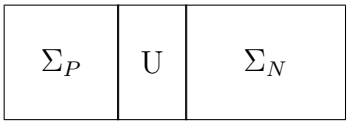


Abbildung 11.5: Unvollständige Semantikakquisition

11.4 Gesamtalgorithmus

Die Kombination der in diesem Kapitel zusammengefaßten Methoden soll noch einmal anhand einer Abbildung dargestellt werden.

Es werden zuerst die in Abschnitt 11.1 beschriebenen Methoden zur Ableitung von Integritätsbedingungen, die ohne Interaktion mit dem Benutzer erfolgen, angewendet. Die in Abschnitt 11.2 beschriebenen Methoden zur Ermittlung von Kandidaten werden angewendet, um die unbekannten Integritätsbedingungen abzuschätzen. In Abschnitt 11.3 wurden die Methoden zur Validierung der ermittelten Kandidaten gezeigt. Diese werden im Anschluß an die Kandidatenermittlung angewendet.

Abbildung 11.6 faßt noch einmal die beschriebene Verknüpfung der Methoden zur Semantikakquisition zusammen.

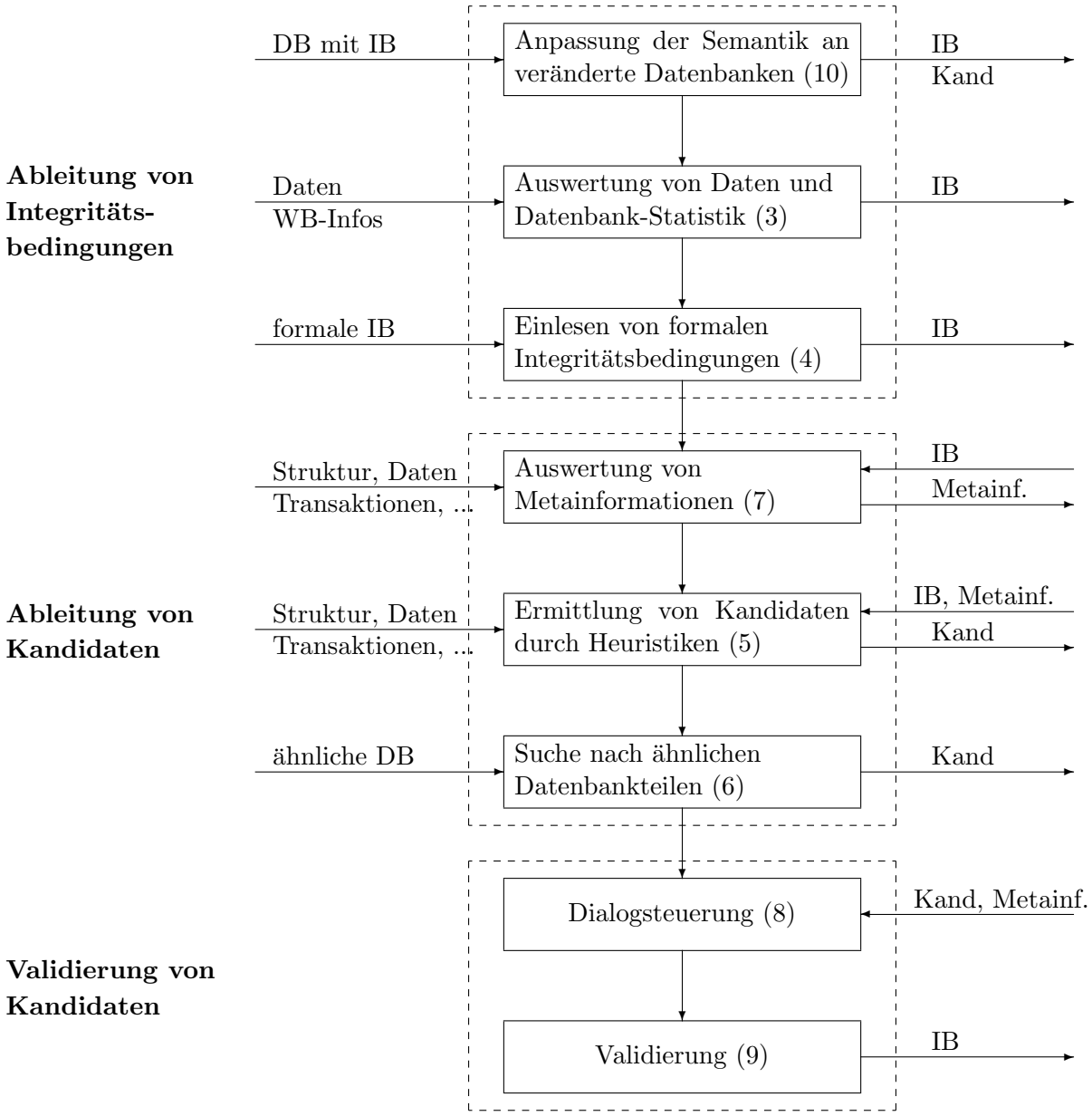


Abbildung 11.6: Verbindung der Methoden zur Semantikakquisition

Die Methoden, die auf unveränderlichen Größen wie der Struktur, den Daten usw. basieren müssen nur einmal durchgeführt werden. Die Verfahren, die Integritätsbedingungen oder Metadaten einbeziehen, müssen dabei mehrfach ausgeführt werden, diese müssen nach Änderungen der entsprechenden Mengen wiederholt werden.

Man kann ersehen, daß die Methoden zur Semantikakquisition modular aufgebaut sind, es können einzelnen Methoden weggelassen und andere hinzugefügt werden, sofern sie auf Integritätsbedingungen oder Kandidaten für Integritätsbedingungen basieren und diese Informationen auch als Ergebnis liefern.

Durch die vorgestellte Semantikakquisition werden Integritätsbedingungen abgeleitet. Allgemeine Eigenschaften der Semantikakquisition werden noch einmal im folgenden Kapitel zusammengefaßt.

Kapitel 12

Zusammenfassung — Möglichkeiten und Grenzen der informalen Semantikakquisition

In dieser Arbeit wurden Methoden vorgestellt, mit denen ein Entwerfer oder Datenbank-Administrator bei der Spezifikation der Semantik eines Datenbank-Schemas unterstützt wird. In diesem Abschnitt sollen noch einmal die Ergebnisse zusammengefaßt werden. Die allgemeinen Eigenschaften der vorgestellten Semantikakquisition werden zunächst noch einmal zusammengestellt (Abschnitt 12.1). In Abschnitt 12.2 wird gezeigt, wie die Unterstützung der Semantikakquisition für verschiedene Typen von Benutzern und für verschiedene Aufgaben aussehen kann. Sich ergebende Probleme werden in Abschnitt 12.3 genannt und mögliche Erweiterungen der Semantikakquisition werden in Abschnitt 12.4 vorgeschlagen.

12.1 Allgemeine Eigenschaften der Methoden zur Semantikakquisition

Im folgenden werden einige wichtige Eigenschaften der vorgestellten Methoden zur Semantikakquisition aufgezählt.

- *Informalität:*
  Die Semantikakquisition kann für den Benutzer vollständig informal erfolgen. Fragen zu den Integritätsbedingungen werden in Form von generierten Beispieldatenbanken gestellt. Dabei werden keine abstrakten Daten, sondern die vom Benutzer eingegebenen Daten herangezogen. Auf diese Weise wird die Semantikspezifikation zu einer Diskussion von Fällen in realitätsnahen Datenbanken. Unterstützt wird diese Beispieldiskussion durch generierte pseudonatürlichsprachige Sätze, die die zu erfragenden Integritätsbedingungen umschreiben und erläutern.
  Es ist möglich, formale Integritätsbedingungen einzugeben, dieses ist aber für das Semantikakquisitionstool nicht erforderlich. Die Semantikakquisition kann also vollständig informal erfolgen.

- *Keine Redundanz der Eingaben:*
  Alle Eingaben (Daten, Datenbank-Statistik-Angaben, Angaben über die Integritätsbedingungen) müssen nur einmal gemacht werden, alle Informationen werden nur einmal erfragt, zwischen den einzelnen Methoden erfolgt eine Weitergabe der Informationen.
  Vor der Erfragung einer Integritätsbedingung erfolgt eine Überprüfung, ob diese Information bereits aus anderen Integritätsbedingungen, Daten oder Datenbank-Statistik-Angaben ableitbar ist. Nur wenn das nicht der Fall ist, erfolgt eine Erfragung.

- *Sicherung der Konsistenz:*
  Für jede Änderung der Menge von Integritätsbedingungen erfolgt die Prüfung, ob die Menge der Integritätsbedingungen konfliktfrei, also konsistent ist. Ist das für eine ergänzte Integritätsbedingung nicht der Fall, so erfolgt vor der Abspeicherung der zu ergänzenden Integritätsbedingungen eine Klärung des Konfliktes. Dadurch können keine Widersprüche in der Menge der ermittelten und gespeicherten Integritätsbedingungen auftreten.

- *Effizienz:*
  Alle vorhandenen Informationen werden ausgenutzt, um Integritätsbedingungen abzuleiten. Durch *Auswertung von Daten* können viele Integritätsbedingungen ausgeschlossen werden. Ebenso können aus vorhandenen *Datenbank-Statistik-Angaben* Integritätsbedingungen abgeleitet werden. Dadurch wird die Menge der noch zu untersuchenden Integritätsbedingungen eingeschränkt.

  Die Effizienz des Verfahrens wird vorwiegend durch den Einsatz von *Heuristiken* erhöht. Dabei wird versucht, Hintergrundwissen über die Anwendung, das sich in dem Datenbank-Schema widerspiegelt, auszunutzen. Alle in dem Datenbank-Schema sichtbaren Merkmale werden in den Heuristikregeln ausgewertet.

  Die Erfragung der Integritätsbedingungen beginnt mit solchen Integritätsbedingungen, aus denen sich viele weitere Informationen über die Semantik ableiten lassen. Auf diese Weise verringert sich die Anzahl der nötigen Dialogschritte zur Erfragung der Semantik.

- *Erweiterbarkeit:*
  Die Methode ist offen für Erweiterungen, alle Einzelmethoden bauen auf der Menge von geltenden und negierte Integritätsbedingungen und auf der Menge von Kandidaten für geltende und negierte Integritätsbedingungen auf und liefern als Ergebnis eine Menge von Integritätsbedingungen oder eine Menge von Kandidaten für Integritätsbedingungen. Dadurch können weitere Methoden ergänzt werden, die andere Informationen auswerten.

  Die Heuristikregeln zur Abschätzung von unbekannten Integritätsbedingungen sind durch die voneinander unabhängige Wichtung ebenfalls problemlos um weitere Heuristiken erweiterbar.

- *Iteratives Vorgehen:*
  Eine iterative Semantikakquisition ist möglich, es kann nicht nur die Semantikakquisition für das gleiche Datenbank-Schema fortgesetzt werden. Es ist auch möglich, für strukturell geänderte Datenbank-Schemata einige bereits bekannte Integritätsbedingungen zu übernehmen, andere Integritätsbedingungen werden als Kandidaten für eine erneute Validierung in das veränderte Datenbank-Schema übernommen.

- *Flexibilität:*
  Die Methode kann mit unterschiedlichen Voraussetzungen umgehen, da die meisten Eingangsinformationen optional sind. Notwendig vorhanden sein müssen:

- strukturelle Angaben zur Datenbank (im relationalen Datenmodell), diese können auch durch Übersetzung aus konzeptuellen Datenmodellen entstanden sein
- Daten, mindestens zwei verschiedene Einträge für jedes Attribut, um daraus Beispielrelationen zur Diskussion der Integritätsbedingungen generieren zu können

Weiterhin können folgende Größen ausgewertet werden:

- formale Integritätsbedingungen
- Datenbank-Statistik-Angaben (Tupelanzahl, Anzahl der Werte für jedes Attribut)
- Transaktionen
- Verhaltensangaben
- Informationen über den Entwurfsprozeß (zeitliche Reihenfolge des Entwurfes, gemeinsam ausgeführte Entwurfsschritte)
- natürlichsprachige Beschreibungen zur Datenbank
- Datenbank-Schemata mit gleichen oder ähnlichen Inhalten

Gibt es in speziellen Datenbank-Schemata zusätzliche Informationen, aus denen Integritätsbedingungen ableitbar sind, dann wäre eine Erweiterung der Semantikakquisition möglich, die diese zusätzlichen Informationen auswertet und Kandidaten oder Integritätsbedingungen aus diesen ableitet.

Durch die große Anzahl optionaler Eingaben kann die Methode zur Semantikakquisition sehr flexibel eingesetzt werden. Eine Anpassung auf den Benutzer ist aufgrund der möglichen expliziten Eingaben realisierbar. Eine Anpassung auf die konkrete Anwendung kann ebenfalls dadurch erreicht werden, daß verschiedenen Ausgangsinformationen ausgewertet werden können.

In diesen Eigenschaften widerspiegeln sich die Vorteile der vorgestellten Semantikakquisition.

12.2 Unterstützung verschiedener Benutzertypen im Entwurf bzw. Reverse-Engineering von Datenbanken

Durch die vorgestellten Methoden ist eine unterschiedliche Unterstützung der Semantikakquisition je nach Anforderungen und Voraussetzungen möglich. Einige typische Fälle sollen in diesem Abschnitt erläutert werden.

12.2.1 Unterstützung ungeübter Entwerfer beim Datenbank-Entwurf

Bei der Semantikakquisition kann ein ungeübter Datenbank-Entwerfer in folgender Weise unterstützt werden.

Der Entwerfer macht Angaben zur Struktur der Datenbank, entweder erfolgt ein Entwurf im Entity-Relationship Modell, das in ein relationales Datenmodell übersetzt wird oder der Entwerfer gibt eine Datenbank im relationalen Datenmodell an.
Weiterhin muß der Entwerfer Beispieldaten angeben, er kann auch Angaben zur Datenbank-Statistik machen. Auch ein ungeübter Entwerfer kann evt. einige Integritätsbedingungen, z.B. Schlüssel eingeben. Für diese wird überprüft, ob sie in den angegebenen Beispieldaten gelten und konform zu den Datenbank-Statistik-Angaben sind. Aus den Daten und den Datenbank-Statistik-Angaben (sofern vorhanden) werden negierte Integritätsbedingungen abgeleitet.
Anschließend werden durch Heuristikregeln Kandidaten für plausible geltende und negierte Integritätsbedingungen ermittelt. Über eine Beispieldiskussion werden diese unbekannten Integritätsbedingungen diskutiert, dabei wird besonders versucht, geltende Integritätsbedingungen zu finden.

Die vorgestellte Methode zur Diskussion von Integritätsbedingungen dient auch zur Erklärung von Integritätsbedingungen. Es ist möglich, daß durch die Beispieldarstellung Integritätsbedingungen so erklärt werden, daß ein Benutzer diese versteht und die formale Spezifikation der Integritätsbedingungen dadurch erlernt.

12.2.2 Unterstützung geübter Entwerfer beim Datenbank-Entwurf

Geübte Datenbank-Entwerfer werden durch den gleichen Zugang in anderer Weise unterstützt.

Der Entwerfer gibt ebenfalls strukturelle Angaben entweder im Entity-Relationship Modell oder im relationalen Datenmodell an. Weiterhin muß der Entwerfer einige Beispieldaten angeben, er kann auch Angaben zur Datenbank-Statistik machen. Der Entwerfer kann alle ihm bekannten Integritätsbedingungen des Datenbank-Schemas spezifizieren. Für diese wird überprüft, ob sie konfliktfrei sind und konform mit den Beispieldaten und der Datenbank-Statistik sind. Die Algorithmen zur Überprüfung, ob eine Menge von Integritätsbedingungen konfliktfrei ist, sind zum Teil recht aufwendig. Auch ein geübter Benutzer kann deshalb nicht immer übersehen, ob eine Menge von Integritätsbedingungen widerspruchsfrei ist. Der vorgestellte Zugang liefert dabei Unterstützung.

Auch ein geübter Entwerfer kann Integritätsbedingungen vergessen, weil diese offensichtlich sind oder weil diese zu kompliziert sind, um sie formal anzugeben. Es wird deshalb versucht, weitere geltende Integritätsbedingungen, die der Entwerfer vergessen oder übersehen hat, in der Menge der unbekannten Integritätsbedingungen zu finden. Dazu erfolgt die Ermittlung plausibler Kandidaten für geltende Integritätsbedingungen durch die Heuristikregeln.
Ein Entwerfer kann auch dadurch unterstützt werden, daß Integritätsbedingungen aus ähnlichen Datenbank-Schemata, die der gleiche Entwerfer erstellt hat oder die für eine ähnliche Anwendung erstellt wurden, übernommen werden. Die unbekannten Integritätsbedingungen, die durch die Heuristikregeln abgeschätzt werden oder aus ähnlichen Datenbank-Schemata übernommen werden, werden mit dem Entwerfer über Beispieldatenbanken diskutiert. Diese Form ist für die Erfragung komplizierter Integritätsbedingungen am besten geeignet.

Der große Vorteil des Zuganges ist, daß in keinem Fall weniger Integritätsbedingungen gefunden werden als der Benutzer explizit spezifizieren kann. In vielen Fällen werden durch die Unterstützung weitere Integritätsbedingungen gefunden. Auch wenn keine zusätzlichen Integritätsbedingungen gefunden werden, hat man durch den Einsatz eines Tools den Vorteil, daß der Entwerfer weiß, daß die Menge der von ihm angegebenen Integritätsbedingungen konfliktfrei ist.

12.2.3 Unterstützung ungeübter Datenbank-Administratoren beim Reverse-Engineering von Datenbanken

Beim *Reverse-Engineering* sind oft große Datenmengen bekannt, die analysiert werden sollen, um aus diesen formale Integritätsbedingungen abzuleiten. Ein manuelles Vorgehen dabei ist sehr aufwendig, es bietet sich deshalb an, die Spezifikation von Integritätsbedingungen für das Reverse-Engineering zu unterstützen. Beim Einsatz eines Tools werden aus einer bestehenden Datenbank negierte Integritätsbedingungen abgeleitet. Es können Integritätsbedingungen des Datenbank-Schemas explizit bekannt sein. Diese sollen ausgewertet werden. Es ist weiterhin möglich, daß auch ein ungeübter Datenbank-Administrator einige explizite Angaben zur Semantik machen kann, diese sollen ebenfalls ausgewertet werden.

Noch unbekannte Integritätsbedingungen werden durch Heuristikregeln abgeschätzt, plausible Kandidaten für geltende Integritätsbedingungen werden anhand von generierten Beispielen mit dem Benutzer diskutiert. Durch den Beispielzugang wird auch ungeübten Benutzern die Bestätigung oder Ablehnung von vorgeschlagenen Integritätsbedingungen ermöglicht.

12.2.4 Unterstützung geübter Datenbank-Administratoren beim Reverse-Engineering von Datenbanken

Auch geübte Datenbank-Administratoren können beim *Reverse-Engineering* durch ein Tool unterstützt werden.
Die Auswertung der bekannten Datenmengen und die Ableitung von formalen Integritätsbedingungen aus diesen erfolgt dabei durch ein Tool.
Der Datenbank-Administrator kann weitere Integritätsbedingungen spezifizieren, es können auch einige Integritätsbedingungen des Datenbank-Schemas explizit bekannt sein. Diese sollen ausgewertet werden.
Es kann sein, daß durch diese Methoden einige geltende Integritätsbedingungen noch nicht ermittelt wurden, es wird deshalb durch die Heuristiken gezielt nach plausiblen Kandidaten für geltende Integritätsbedingungen gesucht, diese werden gegebenenfalls über eine Beispieldiskussion erfragt.
Durch dieses Vorgehen kann der Datenbank-Administrator feststellen, ob die von ihm angegebenen Integritätsbedingungen konfliktfrei und konform zu den bekannten Datenmengen sind. Eine manuelle Überprüfung dieser Eigenschaften ist aufgrund der großen Datenmengen, die bekannt sein können, sehr aufwendig, bzw. gar sind möglich. Weiterhin können durch das Verfahren Integritätsbedingungen ermittelt werden, die der Datenbank-Administrator aus unterschiedlichen Gründen vergessen hat zu spezifizieren.

Generell kann man deshalb sagen, daß die vorgestellte Methode sowohl ungeübte Benutzer, die keine Integritätsbedingungen explizit angeben können, als auch geübte Benutzer bei der Semantikakquisition im Entwurfsprozeß und Reverse-Engineering von Datenbanken in sinnvoller Weise unterstützt.

12.3 Probleme bei der informalen Semantikakquisition

Die Vorteile der vorgestellten informalen Semantikakquisition wurden im Abschnitt 12.1 genannt, es können dabei auch einige Probleme auftreten. Diese werden in diesem Abschnitt noch einmal zusammengefaßt.

Die angegebenen *Heuristiken können fehlschlagen.*
Der Erfolg der Heuristiken ist in starkem Maße von dem Aussehen der Datenbank (und damit von dem Entwerfer und von dem Anwendungsgebiet) abhängig.
Es ist nur das Hintergrundwissen auswertbar, daß sich direkt in der Datenbank (z.B. in bestimmten Bezeichnungen oder den Daten) widerspiegelt. Es wird immer Fälle geben, in denen für den Benutzer offensichtliche Integritätsbedingungen nicht gefunden werden können, da eine Speicherung und Auswertung des Weltwissens nicht möglich ist und sich für diese Fälle keine direkt auswertbaren Hinweise in der Datenbank finden.
Liefern die Heuristiken falsche Ergebnisse, so ermittelt man dadurch zwar keine falsche Integritätsbedingungen des Datenbank-Schemas, da die ermittelten Kandidaten validiert werden, die erwünschte Effizienzerhöhung bei der Diskussion von Integritätsbedingungen mit dem Benutzer wird jedoch nicht erreicht.

Bei *sehr großen Datenbank-Schemata* (Datenbanken mit vielen Attributen) kann auch mit der Semantikakquisition trotz Einsatz von Heuristiken und effizienter Erfragung *keine Vollständigkeit der Semantikakquisition* und damit auch *keine Vollständigkeit der gefundenen Integritätsbedingungen* gewährleistet werden. Es wurden verschiedenen Methoden gezeigt, die den Suchraum einschränken und die Suchreihenfolge festlegen. Dabei wurde versucht, sowenig Informationen über geltende Integritätsbedingungen wie möglich zu verlieren. Da diese Suchraumeinschränkung und Festlegen der Suchreihenfolge auf vagen Informationen (den angegebenen Heuristikregeln und den durch Heuristiken ermittelten Metainformationen) beruht, kann keine Gewähr geliefert werden, daß bei einer unvollständigen Überprüfung der unbekannten Integritätsbedingungen alle geltenden Integritätsbedingungen gefunden werden.

12.4 Erweiterungen der Methoden zur Semantikakquisition

Die Heuristikregeln zur Abschätzung unbekannter Integritätsbedingungen sind besonders wichtig für eine effiziente und möglichst vollständige Semantikakquisition. Die Einschränkung und Umsortierung des Suchraumes basiert überwiegend auf diesen Informationen. In kleinen Datenbank-Schemata werden die plausiblen Kandidaten zuerst erfragt. In sehr großen Datenbank-Schemata werden nur solche unbekannten Integritätsbedingungen erfragt, die durch die Heuristikregeln als plausible Kandidaten abgeschätzt werden. Den Heuristiken kommt deshalb besondere Bedeutung zu. In Kapitel 5 wurden zahlreiche Heuristiken vorgestellt, die die unbekannten Integritätsbedingungen abschätzen.

Die *Heuristiken können gegenüber den vorgestellten Regeln erweitert* werden, indem ein Synonymwörterbuch eingesetzt wird, das die Heuristiken zur Ermittlung ähnlicher Bezeichnungen der Relationen-Schemata bzw. Datenbank-Schemata erweitert. Damit können als ähnliche Bezeichnungen innerhalb eines Relationen-Schemas oder Datenbank-Schemas auch *sprachliche Synonyme* ermittelt werden. Synonyme sind abhängig vom Anwendungsgebiet, mit einem allgemeinen Synonymwörterbuch kann man deshalb nicht alle existierenden Synonyme ermitteln

und nicht alle ermittelten Synonyme werden korrekt sein, man kann jedoch einige sinnvolle Hinweise aus einem Synonymwörterbuch ableiten.

Eine weitere Möglichkeit, die Heuristikregeln zu erweitern, ist die *Auswertung von Transaktionen und Verhaltensinformationen*, die an einigen Stellen bereits angedeutet wurden. Die Auswertung von Transaktionen und Verhaltensinformationen ist schwierig zu realisieren, aber auch bei *eingeschränkten Vergleichen dieser Angaben* oder der *Suche nach bestimmten Muster in diesen Angaben* können wichtige Informationen über das Hintergrundwissen ermittelt werden.

Kapitel 13

Ausblick

Es gibt weitere Aufgabenstellungen, die zu bearbeiten sind. Diese sollen in diesem Kapitel vorgestellt werden.

13.1 Mehrwertige Abhängigkeiten

Die einfache und verständliche Akquisition mehrwertiger Abhängigkeiten gestaltet sich weitaus schwieriger als die Untersuchung funktionaler Abhängigkeiten, da hier die Repräsentation in Beispielen schwieriger ist. Mehrwertige Abhängigkeiten sind schwer zu verstehen und damit auch zu erfragen.

Intuitiv erklären sich mehrwertige Abhängigkeiten so, daß ein Attribut oder eine Attributmenge Y mehrere Werte annehmen kann, diese Werte hängen nur von einem Attribut oder einer Attributmenge X ab, sie sind vom Rest der Relation unabhängig. Eine mehrwertige Abhängigkeit wird auf folgende Weise geschrieben: $X \longrightarrow\!\!\!\!\!\rightarrow Y|Z$, wobei $Z = U - X - Y$ ist.
Aus dieser intuitiven Erklärung ist bereits zu sehen, daß eine funktionale Abhängigkeit $X \longrightarrow Y$ der Spezialfall einer mehrwertigen Abhängigkeit $X \longrightarrow\!\!\!\!\!\rightarrow Y|U - X - Y$ ist. In diesem Fall kann Y nur einen Wert annehmen, dieser hängt ausschließlich von X ab.
Also gilt: $(X \longrightarrow Y) \longrightarrow (X \longrightarrow\!\!\!\!\!\rightarrow Y|U - X - Y)$.
Die Umkehrung $(X \longrightarrow Y|Z) \longrightarrow (X \longrightarrow Y)$ gilt jedoch nicht.

Exakte Definition: Sei R ein Relationsschema, r eine Relation von R, U die Attributmenge von $R, X, Y \subseteq U$. Die *mehrwertige Abhängigkeit* $X \longrightarrow\!\!\!\!\!\rightarrow Y|U-X-Y$ gilt in der Relation r, wenn für alle Tupelpaare s, t von r mit $s[X] = t[X]$, auch ein Tupel u in r existiert, sodaß die folgenden Eigenschaften gelten:

$u[X] = s[X] = t[X]$
$u[Y] = t[Y]$ und $u[V - X - Y] = s[V - X - Y]$.

Anmerkung: u muß nicht verschieden von s und t sein.

Diese Definition stammt aus [PDG 89], sie soll an einem Beispiel erläutert werden, das in [Ull 88] verwendet wurde.

Bsp:

Course	Teacher	Hour	Room	Student	Group
CS101	Deadwood	M9	222	Klunk	B
CS101	Deadwood	W9	333	Klunk	B
CS101	Deadwood	F9	222	Klunk	B
CS101	Deadwood	M9	222	Zonker	C
CS101	Deadwood	W9	333	Zonker	C
CS101	Deadwood	F9	222	Zonker	C

In diesem Beipiel existieren folgende mehrwertige Abhängigkeiten:

Course $\twoheadrightarrow$ Hour, Room
Course $\twoheadrightarrow$ Student, Group
Teacher $\twoheadrightarrow$ Hour, Room
Teacher $\twoheadrightarrow$ Student, Group
Hour, Room $\twoheadrightarrow$ Student, Group, usw.

Intuitiv sind diese Abhängigkeiten klar, Course kann mehrere Termine haben, also mehrere Werte für Hour und Room annehmen, mehrere Studenten besuchen ihn. Diese Abhängigkeiten gelten nach Definition der mehrwertigen Abhängigkeiten in der Relation.

Mehrwertige Abhängigkeiten sich nicht so einfach in einer Relation zu erkennen wie funktionale Abhängigkeiten. In einer leeren Relation gelten alle mehrwertigen Abhängigkeiten. Ebenso wie bei funktionalen Abhängigkeiten kann man nur feststellen, ob mehrwertige Abhängigkeiten erfüllt sind, wenn man die ganze Relation betrachtet. Umgekehrt kann man die Nichterfüllheit von mehrwertigen Abhängigkeiten nicht wie bei funktionalen Abhängigkeiten aus einem Tupelpaar betrachten.

Ein weiteres Problem bei mehrwertigen Abhängigkeiten ist ihre Geltung bei Schemaänderung. Während eine funktionale Abhängigkeit $X \longrightarrow Y$ nach einer Änderung der Relation weiterhin gilt, wenn die Attribute X und Y in der Relation erhalten geblieben sind und deren Werte übernommen wurde, ist das bei mehrwertigen Abhängigkeiten nicht gegeben. Eine Erweiterung der Relation um neue Attribute oder eine Änderung von Attributen einer Relation können mehrwertige Abhängigkeiten ungültig machen. Die Begründung ist der Definition leicht zu ersehen. Mehrwertige Abhängigkeiten gelten nur über folgender Schemaänderung weiter:

$$(X \twoheadrightarrow Y(r)) \longrightarrow (X \twoheadrightarrow Y(r[A]), X, Y \subseteq A)$$

Die Erfragung mehrwertiger Abhängigkeiten ist anhand von Beispielen schwierig. Man kann stattdessen eine pseudonatürlichsprachige Erfragung wählen. Die verständlichste Form der Erfragung ist eine Diskussion von strukturellen Änderungen, da jede mehrwertige Abhängigkeit durch Einführung einer Menge ausgedrückt werden kann.

Zu der Akquisition mehrwertiger Abhängigkeiten gibt es aber noch keine konkreteren Vorstellungen.

13.2 Unterschiedliche Behandlung von Attributwerten

In der bisherigen Version des Programms werden alle Werte, die ein Attribut annehmen kann, gleich behandelt. Dabei können Probleme auftreten, die an einem Beispiel verdeutlicht werden sollen:

Präparat	Packungsgröße	Darreichungsform	Preis	Zuzahlung
			2.75	2.75
			2.75	3.00

Können zwei Einträge in == Arzneimittel == gleiche Werte für
=== Preis
und verschiedene Werte für
=== Zuzahlung haben (j / n / u - unbekannt / m - Menu) ?

In diesem Fall müßte die allgemeine Frage mit "Ja" beantwortet werden, das Beispiel kann jedoch nicht auftreten. Das folgende Beispiel repräsentiert die gleiche Abhängigkeit, dieses Beispiel ist jedoch richtig.

Präparat	Packungsgröße	Darreichungsform	Preis	Zuzahlung
			4.75	4.75
			4.75	3.00

Können zwei Einträge in == Arzneimittel == gleiche Werte für
=== Preis
und verschiedene Werte für
=== Zuzahlung haben (j / n / u - unbekannt / m - Menu) ?

Zur Behebung dieses Problems ist es erforderlich, dem Nutzer die Möglichkeit zu geben, in den Beispielrelationen zu ändern, damit Beispiele nicht abgelehnt werden, weil ein konkreter Fall nicht auftreten kann, obwohl generell eine afunktionale Abhängigkeit gilt.

Die Integritätsbedingungen, die hinter dieser Relation stehen, lassen sich nicht verlustfrei auf die semantischen Constraints abbilden. Insbesondere Berechnungsvorschriften für Attributwerte aus anderen Attributen sollen zur späteren Überwachung der Konsistenz aufgenommen werden. Dazu muß eine Erweiterung der Constraints herangezogen werden, diese kann evt. Teilmenge der Sprache zur Darstellung des Verhaltens sein.
Ein weiteres offenes Problem ist dabei die einfache Erfragung solcher Zusammenhänge, die über semantische Constraints hinausgehen.

13.3 Nullwerte

Die getrennte Behandlung von Nullwerten erfolgt bislang nicht. Es muß geprüft werden, inwieweit es sinnvoll ist, diese im Prototypen des Tools vorzusehen und welche Arten von Nullwerten behandelt werden sollen.

13.4 Verbindung zu anderen Projektteilen

Im Projektantrag wurden die einzelnen Komponenten des zu erstellenden Tools vorgestellt. Die Semantikakquisition wird mit fast allen anderen Projektteilen in Verbindung stehen. Diese Zusammenhänge sollen hier gezeigt werden, da sich daraus weitere Aufgaben ergeben.

13.4.1 Graphischer Editor

Der graphische Editor ist ein Tool zur Strukturerfassung von Datenbankentwürfen. Es lassen sich auch einige semantische Constraints hier eingeben. So kann der Datenbankentwerfer Schlüssel zu den einzelnen Relationen und Kardinalitäten eingeben.

Aus dem Editor werden alle anderen Programmteile aufgerufen, bzw. sind in diesem eingebunden.
Der Aufruf des Tools zur Semantikakquisition erfolgt ebenfalls aus dem graphischen Editor. Dabei vollzieht sich der Datenaustausch über das DataDictionary.
In diesem befinden sich strukturelle und bereits bekannte semantische Informationen, nach Beendigung der Semantikakquisition werden strukturelle und semantische Informationen ins DataDictionary-File zurückgeschrieben.

Aus strukturellen Angaben kann Semantik abgeleitet werden, aus eingetragenen Pfaden und Kardinalitäten lassen sich Inklusionsabhängigkeiten ableiten. Strukturelle Angaben gehen in Heuristiken ein, die bei der Semantikakquisition verwendet werden können (siehe Kapitel **??**).

13.4.2 Zusammenhang zum Verhalten

An verschiedenen Stellen wurde darauf hingewiesen, daß Beispieltransaktionen und dynamische Constraints als Heuristik in die Semantikakquisition eingehen. Die dynamische Komponente ist noch nicht implementiert. Deshalb sind diese Verbindungen bislang nur bei der Beschreibung der einzelnen Methoden erläutert, jedoch noch nicht in dem Tool zur Semantikakquisition enthalten. Hierbei dürften keine theoretischen Probleme auftreten.

Ansätze zur Ableitung semantischer Constraints aus Transaktionen:

Aus einer elementaren Updatetransaktion lassen sich Schlüssel oder funktional Abhängigkeiten ableiten. Werden Werte eines Tupels geändert, so kann man ableiten, daß die Attribute, die zur Identifikation des verwendeten Tupels verwendet werden, die rechte Seite einer funktionalen Abhängigkeit darstellen, die Attribute, die geändert werden, stellen die linke Seite einer funktionalen Abhängigkeit dar. Weiterhin können die Attribute, die zur Identifikation der zu ändernden Tupel verwendet werden, Schlüssel der Relation sein und zwar in dem Fall, daß jeweils nur ein Tupel identifiziert wird.

Die Ableitung von Inklusionsabhängigkeiten kann mit den Methoden, die in **??** beschrieben sind, erfolgen.

13.4.3 Übersetzer

Bei der Übersetzung von HERM-Schemata in relationale und objektorientierte Systeme sind Informationen über die Semantik unbedingt erforderlich, um einen konsistenten und effizienten

Entwurf zu erreichen.

13.4.4 Kritikerkomponente

Bei der Bewertung von Entwürfen und der Untersuchung möglicher dadurch entstehender Probleme sind semantische Constraints eine wichtige Quelle.

In der Kritikerkomponente sollen Datenbankentwürfe untersucht werden und der Nutzer soll auf mögliche entstehende Probleme hingewiesen werden.

Die Bewertung von Entwürfen ist auch für die Semantiksuche relevant, insbesondere bei der Suche nach Inklusionsabhängigkeiten und fehlenden Pfaden. In der Literatur gibt es hierfür kaum Hinweise, ein Ansatz zur Bewertung von ER Diagrammen befindet sich in [HeK 93].

Das Problem der Bewertung von Entwürfen und die Suche nach kritischen oder noch unvollständigen Stellen ist darüber hinaus auch für die Nutzerführung von Bedeutung. Hierbei sollen Entwurfsschritte des Nutzers überwacht und die weitere Vorgehensweise beim Entwurf vorgeschlagen werden.

13.4.5 Möglichkeiten zur Bewertung von Entwürfen

Begründung der Verwendung von Entwurfsbewertung bei der Semantikakquisition Semantik liefert Hintergrundwissen über Datenbanken, das in den anderen Teilen(Modifizierung, Übersetzung, Schemaänderung) ausenutzt wird. Semantik allein reicht dort nicht aus, es müssen zusätzlich Informationen über das Verhalten und allgemeine Hinweise zum Schema (Bewertung des Schemas) ausgenutzt werden.

Bei der Bewertung von Entwürfen ist zunächst relevant, welches Kriterium angesetzt werden soll, da das Ergebnis der Bewertung sich danach richtet.

Es ist möglich, Nichtredundanz von Datenbanken als Kriterium zu verwenden, dieses wird häufig im relationalen Zugang verwendet. Dabei werden Normalisierungsschritte verwendet, um Nichtredundanz zu erreichen. Dieser Ansatz ist jedoch nur möglich, wenn vollständige Informationen über semantische Constraints bekannt sind. Er spiegelt Merkmale wie Effizienz und Güte der Darstellung nicht wider.

Eine Möglichkeit der Effizienzbewertung bzw. Effektivitätsbewertung ist die Betrachtung von Transaktionen und Länge der Antwortzeiten auf Transaktionen. Dabei muß berücksichtigt werden, welche Transaktionen wie oft laufen werden und ob einige Transaktionen zeitkritischer sind als andere. Bewertet werden müssen dabei die Anzahl der Tupel der betreffenden Relationen, die Aufwendigkeit von Joinoperationen, die Unterscheidung, ob Lese- oder Schreiboperationen vorliegen (Yao, Su).

Eine Möglichkeit der Bewertung, wie gut ein Bereich in eine Datenbank abgebildet ist, lässt sich mit den bereits erwähnten Methoden nicht finden. Ein allgemeiner Ansatz dazu ist wahrscheinlich schwer zu finden, eine Lösung kann nur über Faustregeln erfolgen, indem zum Beispiel eingeschätzt wird, wieviele Attribute ein Knoten hat, ob Module existieren, über wieviele Attribute Fremdschlüsselbeziehungen hergestellt werden und welche Verbindungen (wieviele) bestehen, auf diese Weise kann man feststellen, ob Bereiche über- oder unterspezifiziert sind. (weiter verfeinert werden sollten, zusammengelegt werden können.)- Parallele zu Entwurfsunterstützung (top-down, buttom-up)

Der logische nächste Schritt nach der Kritikerkomponente ist die Umwandlung in äquivalente Entwürfe, die besser den angelegten kriterien entsprechen. (Komponente Dresden)

13.4.6 Natürlichsprachige Komponente

Die natürlichsprachige Komponente soll alternativ zu den graphischen Eingabemöglichkeiten und den Tools zur Erfassung von Semantik und Verhalten Informationen über Struktur, Semantik und Verhalten liefern.

Diese aus Texten abgeleiteten Informationen werden teilweise vage sein, sie müssen überprüft und bestätigt werden. Im Tool zur Semantikerfassung kann die Überprüfung von semantischen Constraints, die aus Texten abgeleitet wurden, erfolgen.
Eine mögliche Vorgehensweise dabei ist folgende: Die Constraints werden als Kandidaten übernommen, im Tool zur Semantikakqusition werden Beispiele dazu generiert, diese müssen vom Datenbankentwerfer bestätigt oder abgelehnt werden.

Weiterhin kann aus vagen Aussagen, aus denen sich in der Sprachkomponente keine direkten Informationen ableiten lassen, zusätzlich Heuristiken abgeleitet werden.
So sind die Angaben, welche Bezeichnungen zusammen in einem Satz stehen, welche Begriffe am häufigsten verwendet werden und in welcher Reihenfolge Bezeichnungen verwendet werden, in Heuristiken auswertbar. Die Ausnutzung solcher Informationen wurde bereits in den vorherigen Kapiteln erläutert.

Die sprachlichen Komponente sind viele theoretische und praktische Vorarbeiten zu machen, bevor man zu analysierten Texten kommt (Erläuterungen dazu in den anderen Teile dieses Berichtes). Es kann noch nicht genau gesagt werden, wie die abgeleiteten Informationen aus diesem Projektteil aussehen, deshalb ist in der Nachbehandlung dieser Informationen noch vieles offen.

13.4.7 Komplexität der Validierung

Die Anzahl der notwendigen Schritte bei der Validierung der Kandidaten für Constraints hängt von folgenden Faktoren ab:

- Anzahl der Attribute der Relation
- Anzahl der aus den Instanzen ableitbaren negierten Constraints
- Anzahl der in den Instanzen geltenden Constraints (Schlüssel, funktionale Abhängigkeiten, Inklusions- und Exklusionsabhängigkeiten)
- Attributanzahl dieser Constraints (Bsp: unäre Schlüssel werden eher gefunden als Schlüssel aus drei Attributen)
- Weiterhin ist die Anzahl der Dialogschritte abhängig davon, ob über die Heuristiken die vorhandenen Constraints gefunden wurden. Ist das der Fall, so werden die semantischen Constraints bei der Validierung schneller ermittelt. Lieferte die Heuristik keine Hinweise auf die Constraints, so werden also falsche Abschätzungen gemacht und dadurch sind mehr Schritte erforderlich, um alle noch unbekannten semantischen Constraints zu untersuchen.

Es ist keine allgemeine Angabe der notwendigen Dialogschritte zur Validierung möglich, weil die Wirksamkeit der Heuristiken ein informaler Einfluß auf die Komplexität des Verfahrens ist, der nicht in einer Formel gefaßt werden kann.

Es wurden jedoch in einer empirischen Studie einige Fälle untersucht, bei denen eine schrittweise Erfragung ohne Heuristiken mit der vorgestellten Methode zur Erfragung verglichen wurde. Dabei ergab sich, daß mit den Heuristiken und Algorithmen des schnellen Abstiegs nur höchstens ein Viertel der Dialogschritte bis zur vollständigen Validierung notwendig ist.

13.5 Erweiterung des Verfahrens auf die Akquisition von Kardinalitäten

Das hier vorgestellte Verfahren liefert die Voraussetzungen für die Verfahren des Reverse-Engineering, die aus der Literatur bekannt sind. Es akquiriert Schlüssel, funktionale Abhängigkeiten, Inklusions- und Exklusionsabhängigkeiten. Das beschriebene Verfahren wurde implementiert und für einige einfache Beispiele getestet.

Eine Erweiterung des vorgestellten Vorgehens ist auch auf die Akquisition von Kardinalitäten mit Hilfe von Instanzen möglich. Hier kann man ebenfalls aus Instanzen einige Informationen über nicht geltende Kardinalitäten ableiten (der Ausschluß von 1:1, 1:n und n:1 Kardinalitäten ist durch Instanzen möglich), aus ermittelten Inklusionsabhängigkeiten lassen sich ebenfalls Schlußfolgerungen über die Kardinalitäten ziehen. In einer interaktiven Validierung müssen weitere unbekannte Kardinalitäten untersucht werden.
Voraussetzung dabei ist wieder die Ermittlung von Analoga, die in diesem Artikel beschrieben wurde. Mit diesen Analoga werden Beziehungen zwischen den einzelnen Relationen gefunden, für die dann die Kardinalitäten untersucht werden müssen. Da der Suchraum über die Analoga eingeschränkt wird, ist die vollständige Untersuchung und interaktive Validierung der Kardinalitäten möglich.

Literaturverzeichnis

[AAB95] Meike Albrecht, Margita Altus, Edith Buchholz, Antje Düsterhöft, Bernhard Thalheim: *The Rapid Application and Database Development (RADD) Workbench — A Comfortable Database Design Tool.* In 7th International Conference CAiSE '95, Juhani Iivari, Kalle Lyytinen, Matti Rossi (eds.), Lecture Notes in Computer Science 932, Springer Verlag Berlin, 1995, S. 327-340

[AaP94] Agnar Aamodt, Enric Plaza: *Case-based reasoning: Foundational issues, methodological variations, and system approaches.* AI Communications, 7(1), 1994, S.39-59

[ABD95a] Meike Albrecht, Edith Buchholz, Antje Düsterhöft, Bernhard Thalheim: *An Informational and Efficient Approach for Obtaining Semantic Constraints using Sample Data and Natural Language.* Workshop "Semantics in Databases", Prague, 1995, S. 1-11

[ABD95b] Meike Albrecht, Edith Buchholz, Antje Düsterhöft, Bernhard Thalheim: *Ein Werkzeug zur Gewinnung semantischer Constraints aus natürlichsprachigen Eingaben und Beispieldaten.* Datenbanksysteme in Büro, Technik und Wissenschaft, Springer Informatik Aktuell, Dresden, 1995

[ACW94] J. Akoka, I. Comyn-Wattiau: *A Framework for Automatic Clustering of Semantic Models.* 13th International Conference on the Entity-Relationship Approach, P. Loucopoulos (ed.), Manchester, United Kingdom, LNCS 881, Springer Verlag, December 1994, S. 438-450

[AHV95] Serge Abiteboul, Richard Hull, V. Vianu: *Foundations of Databases.* Addison-Wesley, 1995

[Alb92a] Meike Albrecht: *Erfassung von Semantik in Datenbanken.* Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1992

[Alb92b] Meike Albrecht: *Akquisition von Datenbankabhängigkeiten unter Verwendung von Beispielen.* Preprint CS-04-92, Universität Rostock

[Alb93] Meike Albrecht: *Akquisition von Datenbankabhängigkeiten unter Verwendung von Beispielen.* Tagung des Arbeitskreises der GI-Datenbanken, Grundlagen von Informationssystemen Graal-Müritz, 1993, S. 5-9

[Alb94a] Meike Albrecht: *Ansätze zur Akquisition von Inklusions- und Exklusionsabhängigkeiten in Datenbanken.* GI-Workshop, Tutzing, 1994, Informatik-Berichte, Universität Hannover, S.162-169

[Alb94b] Meike Albrecht: *Semantikakquisition im Reverse-Engineering.* Technische Universität Cottbus, Reihe Informatik I-4/1994

[Alb94c] Meike Albrecht: *Informal and Intelligent Acquisition of Semantic Constraints in Database Design and Reverse-Engineering.* Rostocker Informatik-Berichte (1994) 15, Universität Rostock

[And94] Martin Andersson: *Extracting an Entity Relationship Schema from a Relational Database through Reverse-Engineering.* 13th International Conference on the Entity-Relationship Approach, P. Loucopoulos (ed.), Manchester, United Kingdom, LNCS 881, Springer Verlag, December 1994

[Ape94] Jürgen Apel: *Die Entwicklung eines Strategieberaters für den Datenbankentwurf.* Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1994

[Arm74] W.W. Armstrong: *Dependency structure of database relationships.* Proceedings IFIP, North Holland, Amsterdam, S. 580-583

[AWB92] Klaus-Dieter Althoff, Stefan Weß, Brigitte Bartsch-Spörl, Dietmar Janetzkow, Frank Maurer, Angi Voß: *Fallbasiertes Schließen in Expertensystemen: Welche Rollen spielen Fälle für wissensbasierte Systeme ?* Künstliche Intelligenz - KI (4/92), 14-21, FBO-Verlag, Baden-Baden, 1992

[BCN92] Carlo Batini, Stefano Ceri, Shamkant B. Navathe: *Conceptual Database Design: An Entity-Relationship Approach.* The Benjamin/Cummings Publishing Company, Inc. 1992

[BDF84] Catriel Beeri, Martin Down, Ronald Fagin, Richard Statman: *On the Structure of Armstrong Relations for Functional Dependencies.* Journal of Association for Computing Machinery, Vol. 31, No. 1, January 1984, pp 30-46

[BeE95] Ralph Bergmann, Ulrich Eisenecker: *Fallbasiertes Schließen zur Unterstützung der Wiederverwendung objektorientierter Software: Eine Fallstudie.* Universität Kaiserslautern, Fachbereich Informatik, Daimler-Benz AG, Forschung und Technik, 1995

[Bel95] Siegfried Bell: *The expanded implication problem of data dependencies.* Technical Report LS-8-16, Universität Dortmund, Fachbereich Informatik, 1995.

[BFH77] Catriel Beeri, Ronald Fagin, J.H. Howard: *A Complete Axiomatisation for Functional and Multivalued Dependencies in Database Relations.* ACM SIGMOD, D.C.P. Smith (ed.), Toronto, 1977, S. 47- 81

[BGM85] Mokrane Bouzeghoub, Georges Gardarin, Elisabeth Metais: *An Expert System Approach.* Proceedings of the Eleventh International Conference on Very Large Databases, August 1985, S. 82-94

[BiC86] Joachim Biskup, Bernhard Convent: *A Formal View Integration Method.* Proceedings International Conference on Management of Data, ACM SIGMOD '86, Washington, D.C., S. 398-407

[Bis96] Joachim Biskup: *Grundlagen von Informationssystemen.* Vieweg Verlag, 1996

[BoG86] Mokrane Bouzeghoub, Georges Gardarin: *Tools for Database Design and Programming- a new Perspective.* Sabre Project Paris, France, 1986

[BöP96] Katy Börner, Eberhard Pippig: *Konzeptbildende Analogie: Conceptual Clustering für effizientes analoges Schließen.* Leipziger Informatiktage, 1996

[BOT90] Peter Bachmann, Bernhard Thalheim, Walter Oberschelp, Gottfried Vossen: *The design of RAD: Towards to an Interactive Toolbox for Database Design.* RUTH Aachen, FG Informatik, Aachener Informatikberichte 90-28, 1990

[CaA93] Silvana Castano, Valeria De Antonellis: *A Constructive Approach to Reuse of Conceptual Components.* Advances in Software Reuse, Ruben Prieto-Diaz, William B. Frakes (eds.), IEEE Computer Society Press, Los Alamitos, California, 1993

[CaA94] Silvana Castano, Valeria De Antonellis: *Standard-Driven Re-Engineering of Entity-Relationship Schemas.* 13th International Conference on the Entity-Relationship Approach, P. Loucopoulos (ed.), Manchester, United Kingdom, LNCS 881, Springer Verlag, Dezember 1994

[CaA96] Silvana Castano, Valeria De Antonellis, Carlo Batini: *Reuse at the Conceptual Level: Models, Methods, and Tools.* 15th International Conference on Conceptual Modeling, ER '96, Proceedings of the Tutorials, S. 104-157

[CaS93] Malú Castellonos, Fèlix Saltor: *Extraction of Data Dependencies.* Universitat Politècnica de Catalunya, Barcelona, Spain, Report LSI-93-2-R

[CaS94] Malú Castellonos, Fèlix Saltor: *Semantically Enriching Relational Databases into an Object Oriented Semantic Model.* 5th International Conference, DEXA 1994, Proceedings, Springer-Verlag, LNCS 856, S. 125-134

[CaV83] Marco A. Casanova, V.M.P. Vidal: *Towards a Sound View Integration Methodology* Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, 1983, Atlanta, S. 36-47

[CAF95] Silvana Castano, Valeria De Antonellis, A.G. Fugini, B. Pernici: *Conceptual Schema Analysis: Techniques and Applications.* Politecnico di Milano, Dipartmento di Elettronica e Informazione, Rapporto Interno, No 95-060, 1995

[CAZ92] Silvana Castano, Valeria De Antonellis, B. Zonta: *Classifying and Reusing Conceptual Schemas.* 11th International Conference on the Entity-Relationship Approach, LNCS 645, Karlsruhe, Oktober 1992

[CBS94a] Roger H. L. Chiang, Terence M. Barron, Veda C. Storey: *Extracting Domain Semantics for Knowledge Discovery in Relational Databases.* Proceedings of the Workshop on Knowledge Discovery in Databases, July 31 - August 4, 1994, Seattle, Washington

[CBS94b] Roger H. L. Chiang, Terence M. Barron, Veda C. Storey: *Reverse engineering of relational databases: Extraction of an EER model from a relational database.* Data & Knowledge Engineering 12 (1994), 107-142

[CBS94c] Roger H. L. Chiang, Terence M. Barron, Veda C. Storey: *Performance evolution of reverse engineering relational databases into extended entity-relationship models.* Proceedings of the 12th International Conference on Entity-Relationship Approach, Texas, USA, Springer-Verlag, 1993, S. 352-363

[CFP84] Marco A. Casanova, Ronald Fagin, Christos H. Papadimitriou: *Inclusion Dependencies and their Interactions with Functional Dependencies.* Journal of Computer and System Science, Vol. 28, Febr. 1984, Belgium

[Cha96] Piengjit Chaiyasut: *The Use of Analogical Retrieval to Support Data Model Reuse.* Ph.D. thesis, Faculty of Computing and Information Technology, Monash University, Australia, 1996

[Che76] Peter P.S. Chen: *The Entity-Relationship Model — Toward a Unified View of Data.* ACM Transactions on Database Systems, Vol. 1, No 1, March 1976, S. 9-36

[Che94] Shicheng Chen: *Retrival of Reusable Components in a Deductive, Object-Oriented Database Environment.* Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 1993

[Chi94] Roger H. L. Chiang: *Reverse-Engineering of Relational Databases: Extraction of Domain Semantics.* PhD thesis, University of Rochester, New York, 1994

[ChV92] Joobin Choobineh, Santosh S. Venkatraman: *A Methodology and Tool for Derivation of Functional Dependencies from Business Forms.* in Information Systems Vol. 17, No. 3, 1992, S. 269- 282

[CHP96] Linda J. Campbell, Terry A. Halpin, H.A. Proper: *Conceptual schemas with abstractions Making flat conceptual schemas more comprehensible.* Data & Knowledge Engineering 20 (1996), pp 39-85

[CKV90] Stavros S. Cosmadakis, Paris C. Kanellakis, Moshe Y. Vardi: *Polynomial-Time Implication Problems for Unary Inclusion Dependencies.* Journal of the Association for Computer Machinery, Vol. 37, No. 1, 1990, S. 15-46

[Cod70] E. Codd: *A Relational Model for Large Shared Data Banks.* Communication s of the ACM, Vol. 13, No. 6, June 1970,

[CWA96] Isabelle Comyn-Wattiau, Jacky Akoka: *Reverse-Engineering of Relational Physical Schemas.* 15th International Conference on Conceptual Modeling, Bernhard Thalheim (ed.), Springer-Verlag, LNCS 1157, S. 105-120

[Dav93] Ted Davis: *The Reuse Capability Model: A Basis for Improving an Organization's Reuse Capability.* in Advances in Software Engineering, Selected Papers from the Second International Workshop on Software Reusability, Ruben Prieto-Diaz and William B. Frakes (eds.), IEEE Computer Society Press, Los Alamitos, California, 1993, S. 126-133

[Düs96] Antje Düsterhöft: *Natürlichsprachliche Interaktive Unterstützung im Datenbankentwurf.* Dissertation, Technische Universität Cottbus, Fachbereich Informatik, 1996

[ElN94] Ramez A. Elmasri, Shamkant B. Navathe: *Fundamentals of Database Systems.* Benjamin Cummings, Redwood City etc. 1994

[Ewa95] Cathrine E. Ewald: *Foundations of Conceptual Schema Evolution.* PhD thesis, Queensland University of Santa Lucia, Brisbane, 1995

[FaV83] Ronald Fagin, Moshe Y. Vardi: *Armstrong Databases for Functional and Inclusion Dependencies.* Information Processing Letters 16(1983), North-Holland Publishing Company, S. 13-19

[FaV95a] Christian Fahrner, Gottfried Vossen: *A survey of database design transformations based on the entity-relationship model.* Data & Knowledge Engineering 15 (1995), S.213-250

[FaV95b] Christian Fahrner, Gottfried Vossen: *Transforming relational database schemas into object oriented schemas according to ODMG-93.* In Proc. 4th International Conference on Deductive and Object-Oriented Databases, DOOD '95, LNCS 1013, Tok Wang Ling, Alberto O. Mendelzon, Laurent Vieille (eds.), 1995, S. 429-447

[FaV95c] Christian Fahrner, Gottfried Vossen: *Transformation relationaler Datenbank-Schemas in objekt-orientierte Schemas gemäß ODMG-93.* Datenbanksysteme in Büro, Technik und Wissenschaft, Springer Informatik Aktuell, Dresden, 1995, S. 111-129

[FaV96] Christian Fahrner, Gottfried Vossen: *A Modular Approach to Relational Reverse-Engineering.* Bericht Nr. 22/96-I, Universität Münster, 1996

[FPM91] William J. Frawley, Gregory Piatetsky-Sharipo, Christopher J. Matheus: *Knowledge Discovery in Databases: An Overview.* In [PiF91]

[GHJ95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Publishing Company Inc. 1995

[Glu95] Dieter Gluche: *Flexibilisierung des physischen Datenbankentwurfes in kommerziellen Objektdatenbanksystemen.* Diplomarbeit, Universität Ulm, Fakultät für Informatik, 1995

[GoS89] Robert C. Goldstein, Veda C. Storey: *Some Findings on the Intuitiveness of Entity-Relationship Constructs.* 8th International Conference on Entity-Relationship Approach, Frederick H. Lochovsky (ed.), Toronto, Canada, 1989, North-Holland 1990, S. 9-23

[GoS91] Robert C. Goldstein, Veda C. Storey: *Commonsence Reasoning in Database Design.* 10th International Conference on Entity-Relationship Approach, San Mateo, California, USA, 1991, S. 77-92

[Haf93] Helge Haferkorn: The Design of Distributed Databases. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1993

[Hai96] Jean-Luc Hainaut: *Specifications perservation in schema transformations — application of semantics and statistics.* Data & Knowledge Engineering 19 (1996), S. 99-134

[HeK93] Tom Heilandt, Peter Kruck: *Ein algorithmisches Verfahren zur Bewertung und Verdichtung von Entity-Relationship-Modellen.* Informatik - Forschung und Entwicklung, Band 8, Heft 4, 1993, S. 197-206

[HeS95] Andreas Heuer, Gunther Saake: *Datenbanken, Konzepte und Sprachen.* International Thomson Publishing Group, 1995

[Heu86] Andreas Heuer: *Theorie der IRIS-SYNTHI-Synthese.* Informatik-Bericht 86/2, Technische Universität Clausthal

[Heu92] Andreas Heuer: *Objektorientierte Datenbanken: Konzepte, Modelle, Systeme.* Addison-Wesley, Bonn etc. 1992

[HTJ93] Jean-Luc Hainaut, C. Tonneau, M. Joris, M. Chandelon: *Transformation-based Database Reverse-Engineering.* 13th International Conference on the Entity-Relationship Approach, P. Loucopoulos (ed.), Manchester, United Kingdom, LNCS 881, Springer Verlag, December 1994, S. 364-375

[JaJ84] A.M. Jaglom, I.M. Jaglom: *Wahrscheinlichkeit und Information.* Deutscher Verlag der Wissenschaften, Berlin, 1984

[Jan89] Jürgen Janas: *Covers of Functional Independencies.* Proc. MFDBS 89 (eds J. Demetrovics, B. Thalheim), Berlin Springer Verlag, Lecture Notes in Computer Science 364, 1989, S. 254 - 268

[JoS96] Trevor H. Jones, Il-Yeol Song: *Analysis of binary/ternary cardinality combinations in entity-realtionship modeling.* Data & Knowledge Engineering 19 (1996), S. 39-64

[JOS94] Peter Jaeschke, Andreas Oberweis, Wolffried Stucky: *Extending ER Model Clustering by Relationship Clustering.* 13th International Conference on the Entity-Relationship Approach, P. Loucopoulos (ed.), Manchester, United Kingdom, LNCS 881, Springer Verlag, December 1994,

[Jun94] Dieter Jungnickel: *Graphen, Netzwerke und Algorithmen*, Spektrum Akademischer Verlag, 1994.

[KCV83] P.C. Kannelakis, S.S. Cosmodakis, M.Y. Vardi: *Unary inclusion dependencies have polynomial-time inference problems.* In Proceedings 1983 ACM SIGACT Symposium on Theorie of Computation, Boston, S. 264-277

[KiM92] Jyrki Kivinen, Heikki Mannila: *Approximate Dependency Inference from Relations.* in Database Theory, ICDT '92, J. Biskup, R. Hull (eds), Lecture Notes in Computer Science 646, Springer Verlag Berlin, 1992

[KlG95] René Klösch, Harald Gall: *Objektorientiertes Reverse-Engineering.* Springer Verlag, Berlin, 1995

[KRT95] Bob Kero, Lucian Russell, Shalom Tsur, Wei-Min Shen: *An Overview of Database Mining Techniques.* "Knowledge Discovery and Temporal Reasoning in Deductive and Object-Oriented Databases", Kayliang Ong, Stefan Conrad, Tok Wang Ling (eds.), Proceedings of the Post-Conference Workshops DOOD, National University of Singapore, 1995

[KüL94] Peter Küng, Ambros Lüthl: *Die Bedeutung der Datenbanksprachen in relationalen und objektorientierten Datenbanksystemen.* GI-Tagung, Kassel, Datenbankrundbrief, Ausgabe 13, Mai 1994, S.46–48

[Kün94] Peter Küng: *Datenbanksysteme, Entwicklungsstand, Anforderungen und Bedeutung neuerer Konzepte.* Verlag der Fachvereine, Zürich, 1994

[Lai95] Kari Laitinen: *Natural Naming in Software Development: Feedback from Practitioners.* 7th International Conference CAiSE '95, Juhani Iivari, Kalle Lyytinen, Matti Rossi (eds.), Lecture Notes in Computer Science, 932, Springer Verlag Berlin, 1995, S. 375-388

[LAB96] Mario Lenz, Eric Auriol, Hans-Dieter Burkhard, Michel Manago, Petra Pirk: *Fallbasierte Diagnostik und Entscheidungsunterstützung.* Künstliche Intelligenz, Themenheft Fallbasiertes Schließen, 1/1996

[Lev92] Mark Levene: *The Nested Universal Relation Database Model.* Springer Verlag, 1992

[Lew96] Jana Lewerenz: *Entwicklung einer Benutzermodellierungskomponente zur Gestaltung von natürlichsprachlichen Dialogen für RADD.* Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1996

[Lie85] Y.E. Lien: *Relational Database Design.* in [Yao85], S. 211-254

[Lip89] Udo Lipeck: *Dynamische Integrität in Datenbanken.* Informatik Fachberichte Nummer 209, 1989

[LoP86] László Lovász, Michael D. Plummer: *Matching Theorie.* Akadémiai Kiadó, Budapest 1986

[MaM90] Victor M. Markowitz, J.A. Makowsky: *Identifying Extended Entity-Relationship object structures in relational schemas.* IEEE Transactions on Software Engineering 16(8), August 1990, S. 777-790

[Man96a] Heikki Mannila: *Schema Design and Knowledge Discovery.* 15th International Conference on Conceptual Modeling, Bernhard Thalheim (ed.), Springer-Verlag, LNCS 1157

[Man96b] Heikki Mannila: *Schema Design and Knowledge Discovery.* Vortrag auf der 15th International Conference on Conceptual Modeling, Bernhard Thalheim (ed.), Springer-Verlag, LNCS 1157

[MaR 87] Heikki Mannila, Kari-Jouko Räihä: *Automatic Generation of Test Data for Relational Queries.* Department of Computer Science, University of Tampare, Finland Report A -1987-2

[MaR86] Heikki Mannila, Kari-Jouko Räihä: *Inclusion Dependencies in Database Design.* Proceedings International Conference on Data Engineering, 1986, S.711-718

[MaR89a] Heikki Mannila, Kari-Jouko Räihä: *Automatic Generation of Test Data for Relational Queries.* Journal of Computer and System Science 38, 1989, S. 240-258

[MaR89b] Heikki Mannila, Kari-Jouko Räihä: *Practical algorithms for finding prime attributes and testing normal forms.* SIGACT-SIGMOD-SIGART, Symposium on Principles of Database Systems (PODS '89), New York: ACM, S. 128-133

[MaR89c] Heikki Mannila, Kari-Jouko Räihä: *Design by Example: An Application of Armstrong Relations* Journal of Computer and System Science 33(2), 1989 S. 126-141

[MaR92] Heikki Mannila, Kari-Jouko Räihä: *The Design of Relational Databases.* Addison-Wesley 1992

[MaR94] Heikki Mannila, Kari-Jouko Räihä: *Algorithms for inferring functional dependencies from relations.* Data & Knowledge Engineering 12 (1994), S. 83-99, North Holland

[MaS89] Victor M. Markowitz, Arie Shoshani: *On the Correctness of Representing Extended Entity-Relationship Structures in the Relations Model.* Proceedings International Conference on the Management of Data, ACM SIGMOD Protland, Oregon, SIGMOD Record, Vol 18, Number 2, June, 1989, S. 430-439

[Mat96] Daniel Matuschek: *Eigenschaften struktureller Ähnlichkeit.* Leipziger Informatiktage, 1996

[MiG90] Rokia Missaoui, Robert Godin: *The implication problem for Inclusion Dependencies: A graph approach.* SIGMOD RECORD, Vol. 19, No. 1, March 1990, S. 36-40

[Mit83a] John C. Mitchell: *The Implication Problem for Functional and Inclusion Depenencies.* Information and Control 56, 1983, S. 154-173

[Mit83b] John C. Mitchell: *Inference Rules for Functional and Inclusion Dependencies.* Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, 1983, Atlanta, S. 58-69

[Mou95] Faraj A. El-Mouadib: *Knowledge Discovery in Databases.* Institute of Computer Science, Polish Academy of Science, ICS PAS Reports 785, 1995

[Nav85] Shamkant B. Navathe: *Schema Implementation and Restructuring.* in [Yao85], S. 361-396

[Orl92] M. W. Orlowski: *An Algorithm for Maintenance of Functional Relationships.* in Proceedings of the ICCI '92, Waldemar W. Koczkodaj, Peter E. Lauer, Anestis A. Toptsis (eds.), Fourth International Conference Computing and Information,Toronto, Ontarion, Canada, S. 377-380

[PBG89] Jan Paredaens, Paul De Bra, Marc Gyssens, Dirk Van Gucht, *The Structure of the Relational Database Model.* Springer-Verlag, Berlin, 1989

[PDG89] Jan Paredaens, Paul De Bra, Marc Gyssens, Dirk Van Gucht: *The Structure of the Relational Database Model.* Springer Verlag Berlin, 1989

[Pea84] Judea Pearl: *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Publishing Company, 1984

[Pfe96] Leo Pfefferer: *Kontextvisualisierung mehrdimensionaler Daten.* 8. GI-Workshop "Grundlagen von Datenbanken", Friedrichsbrunn, Otto-von-Guericke-Universität, Preprint Nr.3, 1996

[PiF91] Gregory Piatetsky-Shapiro, William J. Frawley: *Knowledge Discovery in Databases.* AAAI Press/ The MIT Press, California, 1991

[PKB94] J-M. Petit, J. Kouloumdjian, J-F. Boulicaut, F. Toumani: *Using Queries to Improve Database Reverse-Engineering.* 13th International Conference on the Entity-Relationship Approach, P. Loucopoulos (ed.), Manchester, United Kingdom, LNCS 881, Springer-Verlag, December 1994

[PoC93] Jeffrey S. Poulin, Joseph M. Caruso: *A Reuse Metrics and Return on Investment Model.* In Advances in Software Engineering, Selected Papers from the Second International Workshop on Software Reusability, Ruben Prieto-Diaz and William B. Frakes (eds.), IEEE Computer Society Press, Los Alamitos, California, 1993

[PTW94] Manfred Paul, Sven Thelemann, Lutz Wegner: *Darstellungsmöglichkeiten für visuelle NF^2-Datenbankschnittstellen.* Datenbankrundbrief, Ausgabe 13, Mai 1994, S. 55-57

[Ric92a] Michael M. Richter: *Classification and Learning of Similarity Measures.* Workshop Ähnlichkeit von Fällen beim fallbasierten Schließen. K.D. Althoff, S. Weß, B. Bartsch-Spörl, D. Janetzko (Hrsg.), SEKI WORKING PAPER SWP-92-11 (SFB)

[Ric92b] Lutz Richter: *Wiederbenutzbarkeit und Restrukturierung oder Reuse, Reengineering und Reverse Engineering.* Wirtschaftsinformatik, 34. Jahrgang, Heft 2, April 1992, S. 127-135

[Roj93] Raul Rojas: *Theorie der neuronalen Netze.* Springer Verlag, Berlin, 1993

[Sch90] A.-W. Scheer: *Wirtschaftsinformatik – Informationssysteme im Industriebetrieb.* Springer-Verlag, 1990

[Sch91] Hans-Jochen Schneider: *Lexikon der Informatik und Datenverarbeitung.* 3.Auflage, R. Oldenbourg Verlag, München Wien, 1991

[SDG95] Veda C. Storey, Debabrata Dey, Robert C. Goldstein, Roger H. L. Chiang, Shankar Sundaresan: *Common Sence Reasoning and Learning for Database Design.*

[SiM81] Antonio M. Silva, Michael A. Melkanoff: *A Method for Helping Discover the Dependencies of a Relation.* In Advances in Data Base Theory, eds Hervé Gallaire, Jack Hinker, Jean Marie Nicolas, Plenum Press, New York and London, 1981, S. 115-133

[SJB92] William W. Song, Paul Johannesson, Janis A. Bubenko: *Semantic Similarity Relations in Schema Integration.* 11th International Conference on the Entity-Relationship Approach, LNCS 645, Springer-Verlag, Karlsruhe, Oktober 1992, S.97-120

[SJB96] William W. Song, Paul Johannesson, Janis A. Bubenko: *Semantic similarity relations and computation in schema integration.* Data & Knowledge Engineering 19 (1996), S. 65-97

[SLG94] Oreste Signore, Mario Loffredo, Mauro Gregori, Marco Cima: *Reconstruction of ER Schema from Database Applications: a Cognitive Approach.* 13th International Conference on the Entity-Relationship Approach, P. Loucopoulos (ed.), Manchester, United Kingdom, LNCS 881, Springer-Verlag, December 1994

[Spi96] Thorsten Spitta: *Wiederverwendbare Attribute als Ordnungsfaktoren der Unternehmensdaten.* Workshop Natürlichsprachlicher Entwurf von Informationssystemen – Grundlagen, Methoden, Werkzeuge, Anwendungen, 28.-30.5.1996, Tutzing, S. 79-93

[Ste96] Martin Steeg: *The Conceptual Database Design Optimizer CoDO – Concepts, Implementation, Application.* 15th International Conference on Conceptual Modeling, Bernhard Thalheim (ed.), Springer-Verlag, LNCS 1157, S. 105-120

[StG88] Veda C. Storey, Robert C. Goldstein: *Methodology for Creating User Views in Database Design.* ACM Transactions on Database Systems, Sept. 1988, S. 305-338

[Sti91] Eberhard Stickel: *Datenbank Design.* Praxis der Wirtschaftsinformatik, Gabler Verlag Wiesbaden, 1991

[Sto91] Veda C. Storey: *Relational database design based on the Entity-Relationship model.* Data & Knowledge Engineering 7 (1991), S. 47-83

[Su85] Stanley Y.W. Su: *Processing-Requirement Modeling and Its Applications in Logical database Design.* in [Yao85], S. 151-173

[SWB95] Ken Siau, Yair Wand, Izak Benbasat: *A Psychological Study on the Use of Relationship Concept – Some Preliminary Findings.* In 7th International Conference CAiSE '95, Juhani Iivari, Kalle Lyytinen, Matti Rossi (eds.), Lecture Notes in Computer Science 932, Springer Verlag Berlin, 1995, S. 341-354

[SYG96] Veda C. Storey, Heng-Li Yang, Robert C. Goldstein: *Semantic integrity constraints in knowledge-based database design systems.* Data & Knowledge Engineering 20 (1996), S. 1-37

[TAA94] Bernhard Thalheim, Meike Albrecht, Margita Altus, Edith Buchholz, Antje Düsterhöft, Klaus-Dieter Schewe: *The Intelligent Toolbox for Database Design RAD.* Datenbankrundbrief, Ausgabe 13, Mai 1994, S. 28-30.

[TAA96] Bernhard Thalheim, Meike Albrecht, Margita Altus, Edith Buchholz, Haiko Cyriaks, Antje Düsterhöft, Jana Lewerenz, Holger Mehlan, Klaus-Dieter Schewe, Martin Steeg: *Die Entwicklung einer Datenbankentwurfsumgebung der dritten Generation: RADD – Rapid Application and Database Development* Leipziger Informatiktage, 1996

[Tam96] Elisabeth-Ch. Tammer: *Spezielle Möglichkeiten der Strukturierung von Fallbasen im Fallbasierten Schließen.* Leipziger Informatiktage, 1996

[Tan90] Steven L. Tanimoto: *The Elements of Artificial Intelligence.* Computer Science Press, New York, 1990

[Tau89] Branka Tauzovich *An Expert System for Conceptual Data Modelling.* 8th International Conference on Entity-Relationship Approach, Frederick H. Lochovsky (ed.), Toronto, Canada, 1989, North-Holland 1990, S. 205-220

[Tha91a] Bernhard Thalheim: *Dependencies on Relational Databases* Teubner Texte zur Mathematik, Bd. 126, Stuttgart, Leipzig, 1991

[Tha91b] Bernhard Thalheim: *Using Semantics in Extended Entity-Relationship Models* Preprint CS-02-92, Universität Rostock, Fachbereich Informatik

[Tha91c] Bernhard Thalheim Constraints in Extended Entity- Relationship Models Preprint CS-03-92, Universität Rostock, Fachbereich Informatik

[Tha93] Bernhard Thalheim: *Database Design Strategies.* Preprint, Computer Science Institute, Cottbus Technical University, D-03013 Cottbus, 1993.

[Tha98] Bernhard Thalheim: *Fundamentals of Entity- Relationship Modelling.* Springer Verlag, 1998

[Ull82] Jeffrey D. Ullman: *Principals of Database Systems.* Computer Science Press, Rockville, MD, 1982.

[Ull88] Jeffrey D. Ullman: *Principles of Database and Knowledge-Base Systems* Computer Science Press, Rockville, 1988

[VRT82] Günther Vinek, Paul Frederick Rennert, A Min Tjoa: *Datenmodellierung: Theorie und Praxis des Datenbankentwurfes.* Verlag PHYSICA, 1982

[Wat95] Ian D. Watson (ed.): *Progress in Case-Based Reasoning.* Springer Verlag, Berlin, 1995

[Wat95a] Ian D. Watson: *An Introduction to Case-Based Reasoning.* in [Wat95]

[WBA96] Wolfgang Wilke, Ralph Bergmann, Klaus-Dieter Althoff: *Fallbasiertes Schließen zur Kreditwürdigkeitsprüfung.* KI-Themenheft: KI-Methoden in der Finanzwirtschaft 4/96

[Web93] Herbert Weber: *Uniformity and Invarianz in Support of Re-Use.* In Advances in Software Engineering, Selected Papers from the Second International Workshop on Software Reusability, Ruben Prieto-Diaz and William B. Frakes (eds.), IEEE Computer Society Press, Los Alamitos, California, 1993

[Weß93] Stefan Weß: *PATDEX - ein Ansatz zur wissensbasierten und inkrementellen Verbesserung von Ähnlichkeitsbewertungen in der fallbasierten Diagnostik.* In Günther Puppe, Expertensysteme XPS-93, 42-55, Springer-Verlag 1993

[WGS97] Duminda Wijesekera, M. Ganesh, Jaideep Srivastava, Anil Nerode: *Tableaux for functional dependencies and independencies.* Tableaux'97, Minneapolis, 1997, S. 19-29.

[WiB96] Wolfgang Wilke, Ralph Bergmann: *Considering Decision Cost During Learning of Feature Weights.* EWCBR-96, European Conference on Case-Based Reasoning, 1996

[Wil96] Wolfgang Wilke: *CREDITCBR: Fallbasierte Entscheidungsunterstützung mit INRECA.* Leipziger Informatiktage, 1996

[Yao85] S.Bing Yao (ed.): *Principles of Database Design, Volume I: Logical organizations.* Prentice-Hall, 1985.

[Zha93] Yanchun Zhang: *On Horizontal Fragmentation of Distributed Database Design.* Proceedings of the 4th Australian Database Conference, 1993

Index