Einführung in Datenbanken

- Lernskript-

Rene.Reiners@post.RWTH-Aachen.de

Hinweis

Bei dem vorliegenden Dokument handelt es sich lediglich um eine *persönliche* Zusammenfassung, wobei Formulierungen teilweise aus dem Skript übernommen, oder aber auch persönlich erstellt wurden. Inhaltliche Gewichtungen oder Interpretationen stimmen nicht unbedingt mit den Inhalten und Aussagen der Vorlesungen überein.
Daher ist das vorliegende Lernskript lediglich als Hilfestellung bei der Strukturierung der Vorlesungsinhalte zu verstehen. ***Es ersetzt keinesfalls die Vorlesung oder die Bearbeitung der Skripte und erhebt in keinster Weise Anspruch auf eine der genannten Eigenschaften sowie Vollständigkeit und Korrektheit!***

Trotz allem hoffe ich, mit diesem Skript ein wenig Unterstützung beim Erarbeiten der Veranstaltungsinhalte oder einer evtl. Prüfungsvorbereitung geben zu können.
Für Verbesserungsvorschläge und Korrekturen bin ich jederzeit dankbar.

28. Januar 2005

Vorlesungsbezug:

- Einführung in Datenbanken, Professor Jarke, Wintersemester 2003 / 04

Inhaltsverzeichnis

1. Grundlagen

Sinn und Zweck von Datenbanken
Früher mußten für jede Anwendung eigene Algorithmen und Funktion zur Datenverwaltung geschrieben werden – diese unterschieden sich zum Teil sehr deutlich. So kam es zu unterschiedlichen Konzepten des Datenmanagement und anwendungsspezifischen Datenstrukturen.
Ein weiteres großes Problem war die Kompatibilität unter den Programmen, denn aufgrund der unterschiedlichen Ansätze und Strukturen war ein Datenaustausch gar nicht oder nur mit großem Aufwand möglich.
Es sollten Standards zum Umgang mit großen Datenmengen, die auch für andere Anwendungsbereiche zugänglich sein sollten, geschaffen werden.

Mit diesen Standards kann eine zentrale Kontrolle realisiert werden und Redundanzen, die zwangsläufig in jeder Anwendung auftraten (jeder mußte sein Management selbst in die Hand nehmen), stark reduziert werden. Die zentrale Kontrolle ermöglicht weiterhin den Zugriff und die Manipulation auf den Datenbestand durch mehrere Benutzer, Inkonsistenzen und koordinierte Zugriffe sind vorhanden. Auch kann der gesamte Bestand für alle Teilnehmer von zentraler Stelle aus administriert werden.
Somit ist auch die Anwendungsunabhängigkeit gegeben, die Programmierung findet nur noch „um die Datenbank herum“ statt. Auswertungen der Daten als Entscheidungshilfen oder Anwendungen im Data Mining sind weitere, auf Datenbanksysteme gestützte Arbeiten.
Durch die Konzentration auf konkrete Problemstellungen (den Entwurf der Datenbank selbst und nicht der Implementierung des Datenmanagements) und somit den Rückgriff auf existierende Systeme bringt ein DBMS auch ökonomischem Nutzen.

Probleme beim Einsatz und Entwurf von Datenbanken liegen in der Sicherstellung der Konsistenzen von Daten und deren Gültigkeit, der Mehrbenutzersteuerung oder auch der Verwaltung von sehr großen Datenbeständen (Zugriffe, Verarbeitung von Anfragen, Auswertungen → Data Mining, siehe auch „Effizienz). Auch können nicht alle Entwürfe ohne Probleme wiederverwendet werden, da sie auf verschiedene Bereiche angepaßt werden müssen. Die Benutzerfreundlichkeit und Ausdruckskraft des Entwurfs ist ebenfalls nicht aus dem Blickfeld zu verlieren.

Datenbanken modellieren Konzepte aus der realen Welt in einem Datenmodell, welches die Daten verwaltet. Ein DBMS ist für den Zugriff und die Wartung zuständig. Es stellt eine Sprache zur Arbeit mit der Datenbank zur Verfügung. Dabei werden nur die relevanten Aspekte modelliert. Werkzeuge zur Modellierung sind z.B. spezielle Sprachen, die Mathematik, Logik, ER-Schemas, die UML,…

Ein Datenmodell besteht aus
- *Strukturen (Typ-Definitionen, die von Objekten und speziellen Daten auf allgemeine Eigenschaften abstrahieren)*
- *Operationen auf den Strukturen (Manipulation, Aufbereitung)*
- *Einschränkungen, die die Konsistenz des Systems bewaren. Von diesen gibt es zwei Arten:*
    - *Explizit: Benutzer-spezifisch*
    - *Inherent: Implizite Beschränkungen durch das System und seine Beschränkungen (Bsp.: Schnitt(menge) entfernt Duplikate)*

Datenbankmanagementsysteme verwalten zwei Schemas, ein ***internes Schema***, welches für die physikalische Speicherung der Daten verantwortlich ist (wird mit dem DBMS geliefert) und ein ***konzeptionelles Schema***, welches den Datenbankentwurf darstellt. Diese Darstellung ist unabhängig von der physikalischen Speicherung der Daten. Nach außen hin werden auf dem konzeptionellen Schema verschiedene Sichten definiert, welche Anwendern verschiedene Aufbereitungen und Möglichkeiten zur Datenbearbeitung oder –Anzeige zur Verfügung stellen. Dabei müssen nicht jedem Benutzer alle Aspekte zugänglich gemacht werden.

2 Der Entwurf von Datenbanken

2.1 Grundlagen

Der Datenbanken – Lebenszyklus bzw. der Entwurfs Lebenszyklus ist nicht sequentiell angeordnet sondern verläuft zyklisch. Strukturen der realen Welt werden in ein Datenmodell abgebildet (Datenmodellierung). Das *Verhalten* der realen Welt wird in Transaktionen abgebildet. Zusätzlich definiert das Datenmodell Regeln für die Transaktionen (was darf wann gemacht werden?)

Datenbankentwurf:

Anforderungsanalyse → konzeptueller und logischer Entwurf (konzeptueller Entwurf: Der Entwurf ist unabhängig vom Zielsystem. Definition von Objekten (Entitäten), Relationen und Anwendungsregeln. Logischer Entwurf: Konvertierung der Konzepte in ein konkretes Datenbankmodell) → Implementierung → Validierung /Akzeptantztest (Operationen im Einsatz) → wieder zum Anfang – Spiralförmiger Verlauf (vgl. Eierschalenmodell), d.h. ***iterative Verfeinerung****.*

Der konzeptionelle Entwurf einer Datenbank kann Top-Down (schrittweise Verfeinerung) oder Bottom-Up (schrittweise Zusammenfassung von Aspekten) erfolgen (siehe auch: Spezialisierung / Generalisierung).

Ziele des Datenbankentwurfs:

- *Korrektheit:* Wiedergabe der Konzepte des Modells, Integrität, Konsistenz, Datenwiederherstellung
- *Vollständigkeit* aller relevanten Anwendungsaspekte
- *Minimalität:* Keine nichtnotwendige Redundanz
- *Sicherheit:* Datenschutz, Authorisierungsmechanismen, Verfügbarkeit
- *Effizienz:* Reaktionszeit, Kosten von Anfragen, Ressourcen
- *Adaptivität:* Anpaßbarkeit an neue oder unterschiedliche Anforderungen

2.2 Das Entity-Relationship – Modell (ER-Modell)

Dieses Modell wird seit über 20 Jahren aufgrund seiner Wichtigkeit für den konzeptionellen Entwurf zur Modellierung angewendet. Es eignet sich gut zur Kommunikation mit zukünftigen Nutzern des zu entwickelnden Systems (graphische Veranschaulichung). Außerdem geschieht die Modellierung unabhängig vom Datenbanksystem.

Entitäten sind Objekte oder Personen der realen Welt. Sie werden durch ***Attribute*** beschrieben. Symbol: ***Rechteck***.

Attribute charakterisieren und unterscheiden Entitäten voneinander. Weiterhin charakterisieren sie ***Beziehungen*** zwischen den Entitäten. ***Schlüsselattribute*** (unterstrichen) dienen als eindeutiges Identifizierungsmerkmal. Symbole: ***Kreis***, als Oberbegriff ***Kreis mit kleineren angehängten Kreisen***.
Mehrwertige Attribute können mehrere Werte aus ihrem Wertebereich annehmen. Symbol: ***Doppelkreis***.

Beziehungen verbinden Entitäten miteinander. Sie können ebenfalls Attribute besitzen. Symbol: ***Raute*** (evtl. mit Kreisen für Attribute).
Einen Spezialfall stellen ***rekursive Beziehungen*** dar. Die Semantik der Verbindungslinien wird gekennzeichnet.

Teilnahmen von Entitäten an Beziehungen können ***total*** oder ***partiell*** sein. Bei einer totalen Teilnahme werden die Verbindungslinien doppelt gezeichnet.

Kardinalitäten geben an, wie viele Entitäten eines Typs an einer Beziehung teilnehmen können. Teilnahmezwang kann allerdings nicht dargestellt werden.
Arten:

- 1:1 – Beziehung
- 1:m – Beziehung
- n:m – Beziehung
- min:max – Beziehung (Definition unterer bzw. oberer Grenzen)

Generalisierungen und ***Spezialisierungen*** werden mit Hilfe der ***isA-Beziehung*** ausgedrückt. Sie stehen für Vererbungsbeziehungen zwischen allgemeinen und speziellen Entitäten.
Symbol: **umgedrehte Pyramide mit „isA“-Text darin**.
Möglichkeiten:

- *disjunk:* entweder – oder mit „→“ in Richtung Spezialisierung
- *nicht disjunkt:* Überlappungen möglich. Generalisierung „←“
- *total:* vollständige Dekomposition: **t** neben dem Symbol
- *teilweise:* mehr zerlegbare Teile möglich: **p** neben dem Symbol

Aggregationen eigenen sich zum Zusammenbau von komplexen Objekten. Dies erhöht die Lesbarkeit des Diagramms, interne Strukturen werden so versteckt.

- *Integration von Entitäten und deren Beziehungen untereinander ergeben neue Entitäten*
- *Integration von Beziehungs-Klassen ergeben neue Entitätsklassen*

3 Das relationale Datenbankschema

3.1 Grundlagen

Das relationale Datenbankmodell wurde 1970 von Codd eingeführt und ist heute kommerzielle Standard in vielen Systemen wie ORACLE, MySQL,…

In diesem Modell werden die Daten in Relationen gespeichert. Sie können durch „flache Datenstrukturen“ in Form von Tabellen verdeutlicht werden, die über einen eindeutige Bezeichner und eine Reihe von Attributen, welche einen festgelegten Wertebereich besitzen. Diese Tabellen, können ausgewählt, reduziert oder auch durch entsprechende Operationen kombiniert werden.
In den Zeilen der Tabellen werden die Werte Der verschiedenen Attribute repräsentiert, die Attribute einer Relation stehen in den Spalten. In der ersten Zeile einer Tabelle sind der Bezeichner der Tabelle und die Bezeichner der einzelnen Attribute aufgeführt.
Eine gesamte Zeile wird als ***Tupel*** bezeichnet. Unter den Tupeln besteht theoretisch keine Ordnung, obwohl eine solche von der physikalischen Repräsentation impliziert wird

Ein relationales Schema ist ein Verbund von Relationen und ihren Abhängigkeiten untereinander.

*Jedes Tupel wird über einen **Schlüssel** identifiziert. Ein Schlüssel besteht aus einer minimalen Menge von Attributen. Jeder Obermenge wird **Superschlüssel** genannt. Entfernt man aus der Schlüsselmenge ein Attribut, so ist die resultierende Menge kein Schlüssel mehr. In einem Tupel können mehre **potentielle Schlüssel** vorhanden sein, jedoch wird nur einer als **Primärschlüssel** ausgewählt.*

3.2 Übersetzung eines ER-Modells in ein relationales Schema

<u>*Erinnerung:*</u>

- Konzeptioneller Entwurf: ER-Modell
- Logischer Entwurf: relationales Datenbankmodell, relationale Algebra, relationales Kalkül

Algorithmus zur Übersetzung:

1.) Entitäten mit Attributen
Jede Entität wird zu einer Relation (Tabelle). Ihre Attribute bilden ihre Spalten. Zusammengesetzte Attribute werden aufgelöst und ihre atomaren Bestandteile werden in der Tabelle erfaßt. Für jedes mehrwertige Attribut wird eine neue Tabelle erstellt, welche als Schlüssel den zusammengesetzten Schlüssel bestehend aus dem Primärschlüssel der Entität zusammen mit dem Attributsnamen erhält.
Bei starken Entitätstypen bilden die unterstrichenen Attribute auch den Primärschlüssel.
Bei schwachen Entitätstypen werden ebenfalls alle Attribute miteinbezogen.
Als Fremdschlüssel wird der PK der Beziehung verwendet, die mit den Eigentümerentitätstypen korrespondiert.
Der Primärschlüssel wird aus der Kombination des PK der Eigentümerentität, dem Fremd- und partiellen Schlüssel gewählt.

2.) 1:n – Beziehungen

Diese Beziehungen werden in die betroffene Entität auf der „n-Seite“ integriert. Die Relation der „n-Seite“ erhält als Fremdschlüssel den Schlüssel der Entität auf der „1-Seite“ und alle Attribute der Beziehung. Grund: Jede Entitätsinstanz hängt mit mindestens einer Instanz auf der „1-Seite“ zusammen.

3.) 1:1 Beziehungen

Handelt es sich um zwei totale Teilnahmen, so werden beide Entitäten zu einer Relation verschmolzen. Die betroffenen Entitäten dürfen nicht an anderen Beziehungen teilnehmen.
Ansonsten erhält der Entitätstyp mit totaler Teilnahme als Fremdschlüssel den PK des anderen Typen und alle Attribute der Beziehung.

4.) m:n - Beziehungen

Die Beziehung wird als neue Tabelle erstellt. Ihr Schlüssel setzt sich aus den PKs der teilnehmenden Entitäten zusammen. Die teilnehmenden Entitäten werden nach den bekannten Regeln übersetzt.
Anmerkung: Diese Methode ist auch bei allen anderen Beziehungstypen möglich. Es können so Nullwerte in Fremdschlüsseln vermieden werden.

5.) Rekursive Beziehungen

Es werden zwei unabhängige Tabellen erstellt. Ihr Schlüssel ist jeweils die Kombination der PKs der beiden teilnehmenden Entitäten.

6.) Generalisierung / Spezialisierung

Vier Möglichkeiten:

- Erstelle eine Tabelle für die Generalisierungsentität, dann für jede Spezialisierung eine Tabelle mit den Attributen der jeweiligen Spezialisierung und dem Primärschlüssel der Generalisierung.
- Erstelle eine Relation für jede Subklasse mit den passenden Attributen und dem PK der Generalisierung
- *disjunkte Subklassen:* Erstelle eine einzelne Relation mit allen Attributen, demselben PK und einem *diskriminierenden Attribut*, welches die Zugehörigkeit eines Tupels zu einer Subklasse ausdrückt (z.B. Stellentyp)
- *überlappende Subklassen:* Erstelle eine Relation mit demselben PK der Generalisierung. Es werden für jeden Subtyp diskriminierende Attribute eingeführt, die boolesche Werte die Zugehörigkeiten der Tupel zu den jeweiligen Subklassen anzeigen.

3.3 Die relationale Algebra und der relationale Tupelkalkül

Beide stellen formale Sprachen dar, um Abfragen auf einer relationalen Datenbank zu formulieren:

- Operationen auf Tabelleninstanzen
- Tabellen sind Argumente
- Ergebnisse sind wieder Tabellen (sind wieder Relationen und können so weiter bearbeitet werden, d. h. Anfragen können gestellt werden)
- Tabellen werden als eine Menge von Tupeln aufgefaßt (ohne Ordnung)

Mit der ***relationalen Algebra*** werden Anfragen ***prozedural*** formuliert – sie können zu einer Sequenz zusammengefaßt werden, allerdings impliziert die Reihenfolge der Formulierung auch die Reihenfolge der Ausführung.
Mit dem ***relationalen Kalkül (Tupel- oder Domänenkalkül)*** werden Anfragen ***deskriptiv*** („beispielhaft“) formuliert. Es wird beschrieben, was gesucht wird, nicht wie gesucht wird).

Die relationale Algebra ist vollständig unter {σ,π,U,X}.

3.3.1 Die relationale Algebra

Sie ist der operationale Teil des relationalen Modells und stellt Basis-Operationen sowie Berechnungsvorschriften zur Verfügung. Sie unterstützt ein Mengenorientiertes Vorgehen und dient weiterhin zur Optimierung von SQL-Anfragen (welche deklarativ formuliert werden – die Reihenfolge der Ausführungen beeinflußt jedoch die Effizienz der Anfragen).
Es könnenfoglende Basistypen von Operationen verwendet werden:

- Selektion (Tupelauswahl)
- Projektion (Auswahl bestimmter Attribute zur „Anzeige“)
- Natürlicher JOIN (Vereinigung zweier Relationen und Herausstreichen von Attribubtsduplikaten)
- Umbenennung (Von Tupeln oder Attributen)
- Vereinigung
- Mengendifferenz \
- Schnitt
- Kartesisches Produkt
- Zwischenspeicherung von Ergebnissen in Variablen (Es muß nicht immer eine Sequenz als Gesamtheit erstellt werden. Zwischenschritte sind möglich)

3.3.2 Der relationale Kalkül

Mit dem relationalen Kalkül werden Charakteristika des resultierenden Tupels beschrieben. Idee der Benutzung von Variablen für jedes Tupel bzw. Attribut.
Je nach Variablentyp wird unterschieden zwischen

- ***Tupelkalkül:*** Verwendung von Variablen, die Tupel beschreiben, diese werden dann in Bedingungen (logische Formeln) spezifiziert. Ein *sicherer Ausdruck* im Tupelkalkül erzeugt eine endliche Menge von Resultaten bzw. es werden nur Tupel aus dem Wertebereich des Ausdrucks erzeugt → Vorsicht bei Quantoren!
- ***Domänenkalkül:*** Verwendung von Variablen für jedes einzelne Attribut einer oder mehrer Relationen. Beispielhafte Formulierung in den Bedingungen.

Es existiert beim relationalen Kalkül keine Reihenfolge bei der Auswertung.

Es werden viele Beispiele im Buch aufgezeigt, die die Anwendung verdeutlichen.

Es kann gezeigt werden, daß die relationale Algebra, der sichere Tupelkalkül und der Domänenkalkül dieselbe Ausdruckskraft besitzen.

3.4 SQL – Structured Query Language

Die SQL basiert auf dem relationalen Tupelkalkül und algebraischen Elementen. Somit ist sie eine deklarative Sprache. SQL wird als Definitions- und Manipulationssprache (DDL bzw. DML) verwendet. Eine Übersicht über die Befehle findet sich im Buch.

Attribute können z.B. mit NOT NULL oder UNIQUE (Schlüsseleigenschaft) eingeschränkt werden. Zudem muß für jedes Attribut ein Wertebereich angegeben werden.
Verfahrensweisen bei UPDATE- oder manipulierenden Operationen (z.B. CASCADE) können zusätzlich spezifiziert werden.

Eine SQL-Anfrage besteht grundsätzlich aus einem **SELECT-FROM-WHERE-Block**.
Im SELECT-Teil werden die Attribute ausgewählt, die das Resultat der Anfrage bilden sollen, im FROM-TEIL sind die Relationen enthalten, die angefragt werden und WHERE-Abschnitt enthält die Bedingungen der Anfrage.
Verschachtelungen Anfragen und mathematische sowie ordnende Anfragen sind ebenfalls möglich.
Weiterhin können Views ebenfalls ähnlich zu Tabellen definiert werden.

Integritätsbedingungen
Zweck: Sichern die Konsistenz der Datenbank.
Referentielle Integrität: Jeder Bezug auf ein anderes Tupel durch einen Fremdschlüssel existiert auch, jeder Wert eines Attributs stammt aus der Domäne, Verfolgung von Aktualisierungen (Kaskadierung).

4 Formale Kriterien für den Entwurf „guter“ Datenbanken

4.1 Grundlegende Kriterien

Es können bei der Verwendung von Datenbanken *Anomalien* auftreten. Solche Anomalien können in *Lösch- Update- und Einfügeanomalien* unterteilt werden. Bei Löschanomalien kann es dazu kommen, daß bei bestimmten Konstruktionen des Datenbankschemas beim Löschen eines letzten Tupels aus einer Relation, zusätzliche Informationen verloren gehen, die bei einem anderen Entwurf erhalten geblieben wären. Beispiel: Firmenadressen bei Kundendaten in einer Relation, in der alle Daten zusammen in einer Relation gespeichert werden. Bei ähnlichen Relationskonstrukten kann die Änderung eines Wertes einen Änderungszwang für alle restlichen Tupel nach sich ziehen. Beispiel: Der Abteilungsleiter wird in einer Relation geändert, die neben den Mitarbeitern auch die Abteilungen, in denen sie arbeiten, und deren Leiter speichert, so müssen alle Tupel geändert werden, die den alten Wert des Abteilungsleiters ebenfalls besaßen. Beim Einfügen in solche Relationskonstrukte müssen zum Beispiel beim Einfügen eines neuen Mitarbeiters auch zugleich (noch einmal) alle Daten für die Abteilung und deren Leiter korrekt eingegeben werden. Einen anderen Fall stellt die Erstellung einer neuen Abteilung dar, in der noch kein Mitarbeiter arbeitet. Hier müssen Nullwerte eingefügt werden, die jedoch wieder aus der Tabelle entfernt werden müssen, sobald der erste Mitarbeiter für die Abteilung eingestellt wird.

Allgemein können folgende vier informelle Richtlinien formuliert werden:

1.) Entwerfe ein Relationsschema mit klarer Semantik

2.) Es sollen nach Möglichkeit keine Anomalien (s.o) auftreten. Falls dies nicht verhindert werden kann, so muß sichergestellt werden, daß die Programme, die auf der Datenbank arbeiten, korrekt arbeiten.
3.) Es sollen so wenig wie möglich Attribute mit Nullwerten eingefügt werden. Kann dies nicht vermieden werden, so sollen Attribute mit Nullwerten nur für Ausnahmen von Entitäten gelten.
4.) JOINS mit Gleichheitsbedingungen sollen auf Attributen, die weder PK noch FK sind, keine Tupel mit ungültigen Werten (unechte Tupel) erzeugen.

4.2 Funktionale Abhängigkeiten

Funktionale Abhängigkeiten werden als ein Analysemittel zur „Eignung“ bzw. „Unangemessenheit“ von relationalen Datenbankschemas eingesetzt. Sie sind wichtig für die Bestimmung der später vorgestellten ersten drei Normalformen und der Boyce-Codd-Normalform. Weiterhin stellen sie das wichtigste Konzept im relationalen Schemaentwurf dar. Eine funktionale Abhängigkeit, bezeichnet durch X→Y, zwischen zwei Attributmengen X und Y, die Teilmengen eines universellen Relationsschemas („eine große Tabelle mit eindeutigen Attributsnamen“) sind, bezeichnet die Abhängigkeit der Attributsmenge Y von der Menge X. Das heißt, daß die Werte von Y eindeutig durch die Werte von X bestimmt sind. Beispiel: Die SSN bestimmt direkt die Werte für Namen und Adreßattribute.
Die Umkehrung gilt im allgemeinen nicht. Falls X (wie im Beispiel) einen Kandidatenschlüssel darstellt, so gilt die FD für jede beliebige Teilmenge von Attributen Y von R.
Eine funktionale Abhängigkeit ist eine Eigenschaft der Semantik der Attribute. Die Datenbankdesigner wenden ihre Kenntnis der Semantik der Attribute von R an und spezifizieren so die funktionalen Abhängigkeiten. Eine solche FD wird dann als Einschränkung definiert. Relationsextensionen werden als ***legale Extensionen*** bezeichnet, wenn sie die FD erfüllen.

Inferenzregeln für funktionale Abhängigkeiten
Meistens werden beim Entwurf der Datenbanken bereits die FD vom Designer definiert, die offensichtlich sind. Weitere FDs können allerdings noch existieren, welche algorithmisch nach folgenden Bestimmungsregeln aufgedeckt werden:

- IR1 (Reflexivität): Falls Y Teilmenge von X, dann gilt: X→Y
- IR2 (Augmentationsregel): {X→Y} ⊨ XZ → YZ
- IR3 (Transitivitätsregel): {X→Y, Y→Z} ⊨ X→Z
- IR4 (Projektions- oder Zerlegeungsregel): {X→YZ} ⊨ X→Y
- IR5 (Additive oder Vereinigungsregel): {X→Y, X→Z} ⊨ X→YZ
- IR6 (Pseudotransitive Regel): {X→>, WY—YZ} ⊨WX→Z

Eine ausführliche Beschreibung findet sich im Buch.
Die ersten drei Regeln bilden die ***Armstrong Axiome***.
Der Algorithmus soll eine ***minimale Hülle funktionaler Anhängigkeiten*** F finden, d.h.

- Jede Abhängigkeit in F hat ein einzelnes Attribut auf der rechten Seite
- Eine Abhängigkeit X→A in F kann nicht durch eine Abhängigkeit Y→A ersetzt werden, wobei Y eine entsprechende Teilmenge von X ist, und weiterhin eine Menge von Abhängigkeiten haben, die zu F äquivalent ist.
- Es können keine Abhängigkeiten von F entfernt werden, um eine Menge zu erhalten, die zu F äquivalent ist.

Es kann immer mindestens eine minimale Hülle gefunden werden.

4.3 Normalisierungen

Die ersten drei Normalformen sowie die Boyce-Codd Normalform basieren auf den vorgestellten funktionalen Abhängigkeiten zwischen den Attributen einer Relation. Der Top-Down-Entwurf durch Analyse der Relationen auf vorhandene Normalformen und ggfs. Deren Zerlegung in weitere Relationen wird als *relationaler Entwurf durch Analyse* bezeichnet. Der *relationale Entwurf durch Synthese* werden aus gegebenen FD-Mengen 3NF-Relationen synthetisiert.
Ziele der Normalisierung sind die Erreichung von ***minimaler Redundanz*** und die ***Minimierung von Einfüge- Lösch- und Update-Anomalien***.

Eine isolierte Betrachtung des Datenbankentwurfs und dessen Prüfung auf einen Normalisierungsgrad (z.B. 3NF oder BCNF) und die darauffolgende Zerlegung gibt im allgemeinen keine Garantie über die Qualität des Entwurfs. Es muß zusätzlich auf die Existenz der folgenden beiden Eigenschaften geachtet werden:

- Die Eigenschaft des **verlustfreien Joins**, die gewährleistet, daß in Bezug auf die nach der Zerlegung erstellten Relationsschemas keine unechten Tupel erzeugt werden
- Die Eigenschaft der **Abhängigkeitswahrung**, die gewährleistet, daß jede funktionale Abhängigkeit nach der Zerlegung in mindestens einer Relation dargestellt wird.

Die Eigenschaft des verlustfreien Joins muß auf jeden Fall erreicht werden, während die Eigenschaft der Abhängigkeitswahrung manchmal geopfert werden kann (siehe später).
Im heutigen Datenbankentwurf wird der Normalisierung bis BCNF oder 4NF besondere Aufmerksamkeit geschenkt.
Weiterhin besteht kein Zwang, bis zur höchstmöglichen Normalform normalisieren zu müssen (Performanzgründe).

Übersicht über die Normalformen:

- *1NF:* Die Relation sollte keine nicht atomaren Attribute oder verschachtelte Relationen enthalten. ***Lösung:*** Erstelle für jedes nicht atomare Attribut oder jede verschachtelte Relation neue Relationen.
- *2NF:* In Relationen, deren Primärschlüssel mehrere Attribute enthalten, sollte kein Nichtschlüsselattribut funktional von einem Teil des Primärschlüssels abhängen. ***Lösung:*** Zerlege die Relation und erstelle eine neue für jeden partiellen Schlüssel mit seinen abhängigen Attributen. Erhalte die (restliche) Relation mit dem ursprünglichen Primärschlüssel und Attributen, die von diesem funktional voll abhängig sind.
- *3NF:* Eine Relation sollte kein Nichtschlüsselattribut enthalten, das funktional von einem anderen Nichtschlüsselattribut (oder von einer Menge von Nichtschlüsselattributen) bestimmt wird. Das heißt, es sollte keine transitive Abhängigkeit eines Nichtschlüsselattributs vom Primärschlüssel bestehen. ***Lösung:*** Zerlege die Relation und erstelle eine neue, die das bzw. die Nichtschlüsselattribute beinhaltet, die funktional von anderen Nichtschlüsselattributen bestimmt werden.
- *BCNF:* Geringfügiger Unterschied zu 3NF: Das verletzende Attribut kann nur prim sein. ***Problem:*** Bei der BCNF können funktionale Abhängigkeiten verloren gehen. Der später präsentierte Algorithmus stellt jedoch sicher, daß so wenige wie möglich bei der Zerlegung verloren gehen. Strenger als 3NF, da jede NF in BCNF auch in 3NF ist. Nur falls $X \rightarrow A$ in einem Relationsschema R gilt, während X kein Superschlüssel *und* A ein Prime-Attribut ist, befindet sich R nur in 3NF und nicht in BCNF.

4.4 Algorithmen zur Prüfung und Erstellung besserer Schemas

4.4.1 Die Dekomposition

Input: Ein universelles Schema R(U,F)
Output: Eine verlustfreie BCNF – Zerlegung D=(R,.) von R
→ Allerdings können funktionale Abhängigkeiten verloren gehen, der Algorithmus stellt allerdings sicher, daß nur eine minimale Menge von FDs verloren geht.

4.4.2 Der Synthesealgorithmus

Input: Ein universelles Schema R(U,F)
Output: Eine verlustfreie, unabhängige Zerlegung in 3NF (mit beibehalten aller FDs) D=(R,.) von R. Dieser Algorithmus arbeitet schneller als der DekompositionsAlgorithmus.

Anmerkung: Die beiden Algorithmen sind nicht-deterministisch. Der *Synthesealgorithmus* hängt von der erzeugten minimalen Hülle von F ab, d. h. es existieren mehrere minimale Hüllen.
Der *Zerlegungsalgorithmus* ist abhängig von der Reihenfolge, in der ihm die funktionalen Abhängigkeiten zugeführt werden.
→ Einige Entwürfe können somit anderen unterlegen oder sogar gänzlich unerwünscht sein.

5 Objektorientierte Datenbanksysteme

Objektorientierte Datenbankmodelle sollen die Grenzen des relationalen Modells überwinden. Im relationalen Modell konnten nur atomare Werte aus Domänen erfaßt und zueinander in Relation gesetzt werden. Im objektorientierten Modell sollen nun auch strukturierte Daten, wie zum Beispiel multimediale Daten, abgebildet werden können. Zudem sollen Objekte im Modell nun zusätzlich auch Umgangsformen zu den Daten kapseln. Diese operationalen Spezifikationen finden sich bereits im Strukturentwurf. Somit ist die Parallele zur Objektorientierung deutlich. Hier finden ebenfalls die Konzepte der Kapselung, Vererbung und der Polymorphie Anwendung.