Praktikum Datenbanken / DB2
Woche 3: Relationenschemata, Normalformen, Benutzung von DB2

Raum: LF 230 Bearbeitung: 2.-6. Mai 2005

Aktuelle Informationen unter:
`http://www.is.informatik.uni-duisburg.de/courses/dbp_ss03/`

E-R-Modell zu Relationenschema

Als zweiter Schritt beim Entwurf einer Datenbank steht nach der Modellierung die Umsetzung des Modells in ein Relationenschema an. Ziel soll ein Schema sein, das möglichst direkt in einer konkreten relationalen Datenbank implementiert werden kann.

Im Allgemeinen gibt es bei dieser Umsetzung mehrere geeignete Schemavarianten. Eine erste Annäherung für die Umsetzung kann wie folgt aussehen:

- ein Entitätentyp wird in eine Relation mit den gleichen Attributen wie der Entitätentyp überführt
- eine Beziehung wird in eine Relation überführt, deren Attribute die Schlüssel der von ihr verbundenen Relationen sind

Normalerweise gibt es allerdings noch einige andere Dinge zu beachten. Im Folgenden werden ein paar einfache Faustregeln dazu angegeben. Dabei ist zu bedenken, dass in Hinblick auf eine konkrete Anwendung es manchmal sinnvoll sein kann, eine theoretisch nicht optimale Umsetzung zu wählen.

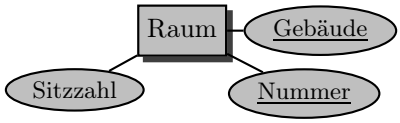
Relationenschemata werden wie in **Kemper/Eick** beschrieben spezifiziert:

SchemaName : { [$\underline{\text{Attribut}_1}$: Typ_1, Attribut_2: Typ_2, ...] }

Dabei wird der Primärschlüssel einer Relation durch Unterstreichung gekennzeichnet. Attribute innerhalb einer Relation müssen eindeutig benannt sein.

Entitätentypen

Ein Entitätentyp wird normalerweise als Relation abgebildet, deren Attribute die Attribute dieses Typs sind. Diese Relation hat zunächst einmal keinerlei Verbindung zu den Beziehungen, an denen der Entitätentyp teilnahm. Mehrwertige Attribute können in ein mengen- oder listenwertiges Attribut abgebildet werden. Diese werden dann bei der Normalisierung behandelt.



Räume : {[$\underline{\text{Nummer: integer}}$, $\underline{\text{Gebäude: string}}$, Sitzzahl: integer]}

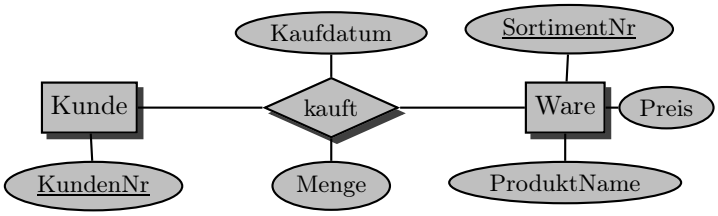
Zusammengesetzte Attribute werden entweder aufgelöst, d.h. die Attribute des zusammengesetzten Attributes werden zu Attributen der Umsetzung des Entitätstyps, oder sie werden als eigene Relation aufgefasst. Diese neue Relation enthält alle Attribute des zusammengesetzten Attributes und alle Primärschlüsselattribute der Ursprungsrelation. Diese Fremdschlüssel bilden den Primärschlüssel der neuen Relation.

Beziehungstypen

Ein Beziehungstyp wird üblicherweise als eine eigene Relation abgebildet. Als Attribute für diese Relation werden herangezogen:

- die Attribute des Beziehungstyps
- die Primärschlüsselattribute der teilnehmenden Entitätstypen als Fremdschlüssel

Der Primärschlüssel der Relation enthält die Fremdschlüssel und eventuell zusätzliche Attribute.



kauft : {[KundenNr: integer, SortimentNr: integer,
Kaufdatum: date, Menge: integer]}

Nimmt ein Entitätentyp mehrfach an einer Beziehung teil, so müssen seine Schlüsselattribute entsprechend oft in die Relation übernommen werden, die dieser Beziehung entspricht. Dabei muss man die Attribute umbenennen, um Namenskonflikte zu vermeiden.

Schwache Entitätstypen

Ein schwacher Entitätstyp wird als eine eigene Relation abgebildet, die zusätzlich zu den eigenen Attributen auch den Primärschlüssel des Vater-Entitätstype als Fremdschlüssel enthält. Dieser bildet zusammen mit den ausgezeichneten Teilschlüsseln des schwachen Entitätstyp den Primärschlüssel der Relation.

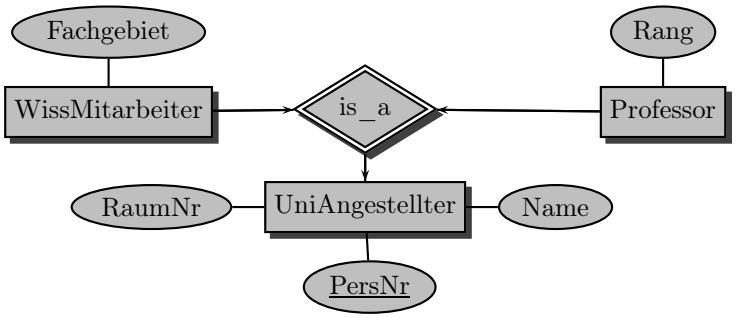
Die Beziehung zwischen einem schwachen Entitätstypen und dem Vater-Entitätstyp muss im Allgemeinen überhaupt nicht nachgebildet werden.

Generalisierung

Generalisierung wird durch das relationale Modell nicht direkt unterstützt. Das relationale Modell verfügt über keine Vererbung. Man kann aber versuchen, Generalisierung mit den existierenden Mitteln nachzumodellieren. Dafür gibt

es unterschiedliche Strategien, wobei sich die folgende am engsten an die E-R-Modellierung hält:

- Für jeden Entitätstypen E in der Generalisierungshierarchie gibt es eine Relation mit den Schlüsselattributen des Obertypen und den Attributen von E



UniAngestellte : {[PersNr, Name, RaumNr]}

Professoren : {[PersNr, Rang]}

WissMitarbeiter : {[PersNr, Fachgebiet]}

Eine Alternative wäre die Überführung des abgeleiteten Entitätstypen E in eine Relation, die als Attribute alle Attribute des Obertypen sowie die Attribute von E besitzt. In dieser Variante würden in der Relation des Obertyps nur Instanzen gespeichert, die zu keinem der Subtypen gehören.

UniAngestellte : {[PersNr, Name, RaumNr]}

Professoren : {[PersNr, Name, RaumNr, Rang]}

WissMitarbeiter : {[PersNr, Name, RaumNr, Fachgebiet]}

Zusammenführen von Relationen

Die direkte Überführung wie oben beschrieben liefert oft nicht die bestmöglichen Relationen. Oft kann man nun noch Relationen zusammenführen. Allgemein gilt: Relationen mit dem gleichen Schlüssel kann man zusammenfassen (aber auch nur diese).

Ein binärer Beziehungstyp R zwischen Entitätentypen E und F, an dem mindestens einer der beiden beteiligten Entitätstypen (sagen wir E) mit Funktionalität 1 teilnimmt, kann mit der E entsprechenden Relation zusammengeführt werden. Dazu führt man eine Relation mit folgenden Attributen ein:

- die Attribute von E
- die Primärschlüsselattribute von F
- die Attribute der Beziehung R

Überlegt selbst, warum das vorteilhafter ist, als die Überführung der Beziehung R in eine eigene Relation. Überlegt auch, was es für Gegenargumente gibt.

Normalformen

Normalformen stellen eine theoretische Basis dar, um relationale Datenbankschemata bewerten zu können. Insbesondere sollen durch Normalisierung Redundanzen und implizite Darstellung von Information verhindert werden. Durch Einhalten der Normalenformen bei der Modellierung können bei der späteren Benutzung Änderungs-, Einfüge- oder Löschungsanomalien verhindert werden.

Erste Normalform

Eine Relation ist in *Erster Normalform (1NF)*, wenn alle Attribute nur atomare Werte annehmen dürfen. Zusammengesetzte oder mengenwertige Attribute sind unter der 1NF nicht zulässig. In klassischen relationalen DBMS lassen sich Relationen, die nicht in 1NF sind, nicht speichern. Die Relation

Prüfung: {Matrikel, Name, Fachbereich, FBNr, {Prüfungsnr, Prfnote}}

ist nicht in 1NF, da es ein zusammengesetzes und mengenwertiges Attribut {Prüfungsnr, Prfnote} gibt.

Nachteile einer Relation, die nicht in 1NF ist, sind unter anderem fehlende Symmetrie, Speicherung redundanter Daten und Aktualisierungsanomalien. Bei der Überführung in 1NF entstehende neue Redundanzen werden im Laufe der weiteren Normalisierung beseitigt.

Vorgehen:

- Listen- bzw. mengenwertige Attribute werden durch ihre Elemente ersetzt. Auf Tupelebene wird das durch Ausmultiplikation der Liste mit dem restlichen Tupel erreicht.
- Zusammengesetzte Attribute werden durch die Teilattribute ersetzt.
- Alternativ können auch neue Relationen gebildet werden.

Dies liefert die Relation

Prüfung: {Matrikel, Name, Fachbereich, FBNr, Prüfungsnr, Prfnote}

in der für jede Prüfung mit Prüfungsnote ein eigenes Tupel, d.h. eine eigene Zeile in der Tabelle existiert.

Zweite Normalform

Eine Relation ist in *Zweiter Normalform (2NF)*, wenn jedes Nichtschlüsselattribut von jedem Schlüssel voll funktional abhängig ist. Relationen, die diese Normalform verletzen, enthalten Information über zwei Entitätstypen, was zu

Änderungsanomalien führt. Bei der Überführung in 2NF werden die verschiedenen Entitätstypen separiert.

Die folgende Relation mit den funktionalen Abhängigkeiten *Matrikel* → *Name*, *Matrikel* → *Fachbereich*, *Matrikel* → *FBNr*, *Fachbereich* → *FBNr* und *Matrikel, Prüfungsnr* → *Prfnote* ist nicht in 2NF, da Name, Fachbereich und FBNr nur von einem Teil des Schlüssels abhängig sind.

Prüfung: {Matrikel, Name, Fachbereich, FBNr, Prüfungsnr, Prfnote}

Vorgehen:

- Ermittlung der funktionalen Abhängigkeiten zwischen Schlüssel- und Nichtschlüsselattributen wie in der Vorlesung gezeigt
- Eliminierung partieller funktionaler Abhängigkeiten, durch Überführung der abhängigen Attribute zusammen mit den Schlüsselattributen in eine eigene Relation und Entfernung der abhängigen Attribute aus der ursprünglichen Relation.

Dieses Verfahren liefert die Relationen:

Student: {Matrikel, Name, Fachbereich, FBNr}

Prüfung: {Matrikel, Prüfungsnr, Prfnote}

Dritte Normalform

Eine Relation ist in *Dritter Normalform (3NF)*, wenn jedes Nichtschlüsselattribut von keinem Schlüssel transitiv abhängig ist. In der Relation Student liegt die transitive Abhängigkeit *Matrikel* → *Fachbereich* → *FBNr* vor, sie ist daher nicht in 3NF.

Vorgehen:

- Ermittlung der funktionalen und transitiven Abhängigkeiten
- Für jede transitive funktionale Abhängigkeit $A \rightarrow B \rightarrow C$ werden alle von B funktional abhängigen Attribute entfernt und zusammen mit B in eine eigene Relation überführt.

Dieses Verfahren liefert die Relationen:

Student: {Matrikel, Name, Fachbereich}

Fachbereich: {Fachbereich, FBNr}

Benutzung von DB2

Im Praktikum wird als Datenbank-Managementsystem (DBMS) das kommerzielle System *IBM DB2 Universal Database* in der Version 8.1 verwendet. Dieses läuft auf dem Rechner `salz`, auf den übrigen Maschinen des Rechnerpools LF230 laufen *Administration Clients* für IBM DB2.

Die Benutzung des Kommandozeilenwerkzeugs zum Zugriff auf DB2 wurde in der Einführungsveranstaltung demonstriert. Zur Erinnerung: es wird zwischen drei Benutzungsarten unterschieden:

- Aufruf aus der Kommandozeile mit vorangestelltem `db2`
- Aufruf mit einer Batchdatei als `db2 -tvf filename.sql`
- interaktiver Aufruf, eingeleitet mit `db2 -t`

Zeilen, die mit −− beginnen, werden als Kommentar betrachtet und ignoriert. Mit `? {Befehlsname}` kann man Hilfen zu Befehlsnamen, Warnungs- oder Fehlercodes bekommen.

Die Dokumentation zu DB2 steht online und in Form der englischsprachigen Benutzerhandbücher als PDFs im Verzeichnis `/home/dbprak/doc` zur Verfügung. Diese Handbücher sind sehr umfangreich und sollten nur zum Betrachten am Bildschirm herangezogen und nicht ausgedruckt werden. Die Referenzen in diesem Arbeitsblatt beziehen sich auf diese Handbücher. Die gleichen Informationen sind jedoch auch in der Online-Dokumentation zu finden. Besonders wichtig sind hier die „SQL Reference" und die „Command Reference".

db2s1e80.pdf
db2s2e80.pdf
db2n0e80.pdf

Grundlegende DB2 Struktur

Innerhalb einer DB2 Installation existieren die folgenden wichtigen Datenbanksystem-Objekte:

- Instanzen
- Datenbanken
- Schemata
- Tabellen
- Views

Tabellen und Views werden in den folgenden Sitzungen behandelt.

Instanz

Eine Instanz (auch: Datenbankmanager) ist ein Objekt, das Daten verwaltet. Es kontrolliert, was mit den Daten gemacht werden kann, und verwaltet die ihm zugewiesenen Systemressourcen. Jede Instanz ist eine vollständige Arbeitsumgebung, die ihre eigenen, für andere Instanzen unzugänglichen Datenbanken enthält. Alle Datenbankpartitionen einer Instanz teilen sich das gleiche Systemverzeichnis. Eine Instanz hat eine eigene Rechteverwaltung, die unabhängig von der anderer Instanzen ist.

Jede Kleingruppe im Praktikum besitzt eine eigene Instanz auf `salz` unter dem Namen ihres Accounts (z.B. `dbps0501`). Auf dieser besitzt die Gruppe das Administratorrecht, und kann Datenbanken anlegen, ändern und löschen.

Die Konfiguration der eigenen Instanz kann man mit dem Befehl `db2 get dbm cfg` oder `db2 get database manager configuration` abfragen. Im interaktiven oder Batchmodus muß man natürlich das einleitende `db2` weglassen. Im Folgenden werden DB2-Befehle durch (db2) gekennzeichnet.

Um sich mit einer Instanz auf einem entfernten Rechner zu verbinden, muß man diese zunächst als einen lokalen Knoten katalogisieren. Wer dies in der Einführungsveranstaltung bereits getan hat, kann diesen Schritt überspringen. Der Befehl zum Katalogisieren einer externen Instanz über TCP/IP lautet für das Praktikum:

(db2) `db2 catalog tcpip node salz`
`remote salz.is.informatik.uni-duisburg.de server 50050`
`remote_instance dbprak system salz ostype linux`

Datenbank

Eine relationale Datenbank ist eine Sammlung von Tabellen. Eine Tabelle besteht aus einer festen Zahl von Spalten, die den Attribute des Relationenschemas entsprechen, sowie einer beliebigen Zahl an Reihen. Zu jeder Datenbank gehören Systemtabellen, welche die logische und physikalische Struktur der Daten beschreiben, eine Konfigurationsdatei mit den Parametern der Datenbank und ein Transaktionslog.

Eine neue Datenbank wird mit dem Befehl (db2) `create database {name}` erstellt. Dies erzeugt eine Datenbank mit der Standardkonfiguration. Gelöscht werden kann die Datenbank wieder mit dem Befehl (db2) `drop database {name}`.

Der Befehl (db2) `connect to {name}` stellt eine Verbindung zu einer Datenbank her, die mit (db2) `terminate` wieder beendet wird.

Um sich von einem entfernten Rechner mit einer Datenbank zu verbinden muß diese zunächst an einem lokalen Knoten katalogisiert werden. Angenommen es existiert bereits ein lokaler Knoten *salz*, dann lautet der Befehl zum Katalogisieren einer entfernten Datenbank *sample* auf dem mit *salz* verbundenen Server wie folgt:

(db2) `catalog database sample as samplexx at node salz`

Dabei ist der *samplexx* der Aliasname, unter dem man sich mit der Datenbank lokal verbinden kann.

Schemata

Ein Schema ist eine Sammlung von benannten Objekten (Tabellen, Sichten, Trigger, Funktionen,...). Jedes Objekt in der Datenbank liegt in einem Schema. Objektnamen müssen innerhalb eines Schemas eindeutig sein. Ein Objekt in der Datenbank wird durch einen *qualifizierten* Namen identifiziert. Dieser besteht aus dem Namen des Schemas, in dem das Objekt liegt, gefolgt von einem Punkt und dem Objektnamen: DBPRAK.STAFF würde beispielsweise das Objekt STAFF im Schema DBPRAK bezeichnen.

Wird nur ein Objektname angegeben, so qualifiziert DB2 diesen standardmässig durch den Benutzernamen. Man kann diesen Standardwert auch ändern:

(db2) `SET SCHEMA dbprak`

Tabellen

Eine relationale Datenbank speichert Daten als eine Menge von zweidimensionalen Tabellen. Jede Spalte der Tabelle repräsentiert ein Attribut des Relationenschemas, jede Zeile entspricht einer spezifischen Instanz dieses Schemas. Daten in diesen Tabellen werden mit Hilfe der Structured Query Language (SQL) manipuliert, einer standardisierten Sprache zur Definition und Manipulation von Daten in relationalen Datenbanken.

Tabellenerstellung, Constraints und Datentypen werden in der nächsten Sitzung behandelt. Um sich in einer existierenden Datenbank die vorhandenen Tabellen anzeigen zu lassen, benutzt man den Befehl

(db2) `list tables for schema {name}`

Mit Hilfe von (db2) `describe table {schemaname}.{tablename}` kann man Einzelheiten über den Aufbau einer existierenden Tabelle herausfinden.

Benutzerverwaltung

Das Authentifizierungskonzept von DB2 sieht keine gesonderten Benutzernamen vor, sondern baut auf das Benutzerkonzept des Betriebssystems auf. Jede Kleingruppe meldet sich unter dem Namen ihres Accounts bei der Datenbank an, zusätzlich gehört jede Kleingruppe zur Benutzergruppe `students`.

In einer DB2-Instanz gibt es drei Stufen der Zugriffsberechtigung. Als Instanzbesitzer hat jede Gruppe in ihrer Instanz das SYSADM-Recht.

SYSADM: Zugriff auf alle Daten und Ressourcen der Instanz

SYSCTRL: Verwalten der Instanz, aber kein direkter Zugriff auf Daten

SYSMAINT: niedrigste Berechtigungsstufe; kann z.B. Backups erstellen

Um Berechtigungen in der Datenbank, mit der man gerade verbunden ist, zu setzen oder zu verändern, bedient man sich des GRANT-Statements. Umgekehrt wird das REVOKE-Statement benutzt, um Rechte auf einer Datenbank zu entziehen. `GET AUTHORIZATIONS` zeigt die aktuellen Berechtigungen an. Beispiele:

db2s2e80.pdf, S. 569/642

(db2) `GRANT dbadm ON DATABASE TO USER dbpw0301`
(db2) `GRANT connect, createtab ON DATABASE TO GROUP students`
(db2) `GRANT connect ON DATABASE TO public`
(db2) `REVOKE connect ON DATABASE TO public`

Ein Benutzer oder eine Gruppe mit dem DBADM-Recht besitzt automatische auch alle anderen Rechte auf dieser Datenbank, insbesondere auch das Recht IMPLICIT_SCHEMA. Das Recht DBADM kann nicht an PUBLIC vergeben werden. Nur ein Benutzer mit SYSADM-Recht kann das DBADM-Recht an

andere Benutzer oder Gruppen verleihen oder entziehen. Wird einem Benutzer ein Recht entzogen, bedeutet das nicht notwendig, dass dieser das Recht nun nicht mehr besitzt, da Rechte auch durch Zugehörigkeit zu Gruppen bestehen können.

Wer das Recht DBADM oder SYSADM besitzt kann Berechtigungen für ein Schema ändern. Dabei kann einem Benutzer auch das Recht gegeben werden, die erhaltenen Rechte an Dritte weiterzugeben. Der Befehl zum Vergeben bzw. Entziehen von Rechten auf Schemata ist GRANT/REVOKE ... ON SCHEMA.

db2s2e80.pdf, S. 583/657

Übersicht über die neuen Befehle

?	Hilfe
? *command*	Hilfe zu einem Befehl
get dbm configuration	Konfiguration des Datenbankmanagers anzeigen
catalog tcpip node *name*	entfernten Datenbankserver als lokalen Knoten *name* katalogisieren
list node directory	Verzeichnis aller katalogisierten Knoten anzeigen
uncatalog node *name*	katalogisierten Knoten *name* aus dem Verzeichnis entfernen
catalog database *name* as *alias*	Datenbank *name* unter lokalem Alias *alias* katalogisieren
list database directory	Verzeichnis aller katalogisierter Datenbanken anzeigen
uncatalog database *alias*	als *alias* katalogisierte Datenbank aus dem Verzeichnis entfernen
create databse *name*	neue Datenbank *name* erstellen
drop database *name*	Datenbank *name* löschen
connect to *name*	Verbinden mit Datenbank *name*
terminate	Verbindung mit Datenbank beenden
set schema *name*	das aktuelle Schema auf *name* setzen
get authorizations	die Privilegien auf der aktuellen Datenbank anzeigen
grant *privilege* on database to public/group *name*/user *name*	allen, einer Gruppe oder einem Benutzer auf der aktuellen Datenbank ein Privileg verleihen
revoke *privilege* on database to public/group *name*/user *name*	ein Privileg auf der aktuellen Datenbank wieder zurücknehmen
list tables for schema *name*	alle Tabellen für das Schema *name* anzeigen
describe table *name*	den Aufbau und die Attribute der Tabelle Name anzeigen

Aufgaben

Aufgaben zur Vorbereitung

(a) Nimm die Modellierung der Mini-Welt der Länder aus der vergangenen Woche und übertrage sie zunächst möglichst modellierungsgetreu in ein relationales Schema. Verfeinenere das relationale Schema dann soweit möglich und sinnvoll wie beschrieben. Eventuell werden hier bereits erste Probleme der ursprünglichen Modellierung sichtbar. Halte diese Beobachtungen fest.

(b) Bringe Dein relationales Schema in die 3. Normalform.

Präsenzaufgaben

(a) Diskutiert in Eurer Gruppe die Probleme Eurer ursprünglichen Modellierung, die bei der Übertragung in das relationale Schema sichtbar geworden sind. Überarbeitet gegebenfalls die Modellierung. Datenbank–Entwurf ist fast immer ein iterativer Vorgang.

(b) Verschafft Euch, wenn noch nicht geschehen, Zugriff auf die existierende Datenbank `sample`. Dazu muß zunächst der Datenbank–Server als ein neuer Knoten und dann die Datenbank `sample` im Schema `dbprak` unter einem Alias katalogisiert werden. Alle Gruppen haben Leserecht auf diese Datenbank.

(c) Versucht, möglichst viel über diese Datenbank herauszufinden. Welche Tabellen existieren, wie sind diese aufgebaut? Welche Berechtigungen habt Ihr auf der Datenbank?

Tabellen:

Schema der Datenbank:

Berechtigungen:

(d) Um eine eigene, neue Datenbank anzulegen, müßt Ihr Euch mit dem Datenbank-Server verbinden. Setzt dazu aus einem Terminal heraus zunächst den Befehl `ssh salz` ab. Legt dann eine neue, eigene Datenbank mit dem Namen `almanach` an.

Befehl:

(e) Gebt Eurem Tutor (mit dem Accountnamen `dbprak`) die Berechtigung, sich mit Eurer Datenbank zu verbinden.

Befehl:

(f) Katalogisiert nun auf dem lokalen Rechner (eventuell zunächst die ssh-Verbindung beenden oder ein neues Terminal benutzen) die neu erstellte Datenbank unter dem Alias `almanXX` (ersetzt das `XX` durch Eure Gruppennummer).

Befehl: