BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Grundlagen des relationalen Datenmodells

Medieninformatik Bachelor
Modul 9:
Datenbanksysteme

Sie erinnern sich: Ihre Aufgaben

- **Aufgabe 1 Anfragen & Modellierung“**
  Denken Sie mal darüber nach, welche Anfragen Sie an die AOL Daten stellen möchten. Bitte Sie bitte ein logisches und physisches Schema zur Beantwortung dieser Anfragen.
- **Aufgabe 2 „SQL und Anfrageausführung“**
  Bitte formulieren Sie für Ihre Analyseideen aus 1.) die SQL Anfragen. Sie verstehen auch Möglichkeiten der Anfrageausführung bzw. Optimierung.
- **Aufgabe 3 „Datenintegration“**
  Zur Ausführung der Ausführung fehlen Ihnen noch externe Daten, z.B. aus dem Internet Archive, DMOZ oder Freebase.org. Bitte ergänzen Sie Ihr Schema und die Datenbasis.
- **Aufgabe 4 „Analyse, Erkenntnisgewinn und Wert“**
  Stellen Sie in 5 Minuten die wichtigsten Erkenntnisse aus den Daten vor. Bewerten Sie den Erkenntnisgewinn, z.B. gegenüber Ihren Kommilitonen oder der Literatur! Welche Erkenntnisse hätten einen kommerziellen Wert?

Die Themen

- Was sind Datenbanken?
  - Motivation, Historie, Datenunabhängigkeit, Einsatzgebiete
- Datenbankentwurf im ER-Modell & Relationaler Datenbankentwurf
  - Entities, Relationships, Kardinalitäten, Diagramme
  - Relationales Modell, ER -> Relational, Normalformen, Transformationseigenschaften
- Relationale Algebra & SQL
  - Kriterien für Anfragesprachen, Operatoren, Transformationen
  - SQL DDL, SQL DML, SELECT … FROM … WHERE …
- Datenintegration & Transaktionsverwaltung
  - JDBC, Cursor, ETL
  - Mehrbenutzerbetrieb, Serialisierbarkeit, Sperrprotokolle, Fehlerbehandlung, Isolationsebenen in SQL
- Ausblick
  - Map/Reduce, HDFS, Hive …
  - Wert von Daten

Überblick

- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Konvertierung von Spezialisierung
- Modell-inhärente Integritätsbedingungen
  - Entity Integrität (Primärschlüsselbedinung)
  - Referentielle Integrität (Fremdschlüsselbedingung)
  - Wahrung der referentiellen Integrität bei Datenmanipulation
- Normalformenlehre

RDM – Integritätsbedingungen

Problemstellung:

- Kartesisches Produkt zweier Mengen ➔ Jeder Wert von Menge 1 ist mit jedem Wert von Menge 2 zu kombinieren: $W(A_1) \times W(A_2)$
- Außer der Forderung nach Eindeutigkeit der Tupel gibt es zunächst keine Einschränkung für die Attributwerte einer Relation ➔ Alle Kombinationen sind erlaubt (solange sie eindeutig sind)
- Aber ist das Erlauben aller Kombinationsmöglichkeiten auch sinnvoll?

Diskussions-Beispiel:

Dozent

DozNr	Name	Vorname	Aufenthaltsort	Gebäude
123	Blankenburg	Marco	D139	D
123	Blankenburg	Marco	B131	B
124	Sauer	Petra	D139	D
125	Steyer	Frank	D139	D

Tupel-Eindeutigkeit zwar gegeben, aber Herr Blankenburg an zwei Orten gleichzeitig?! → Logischer Widerspruch

Wofür steht das Gebäudekennzeichen **D** ?
→ Fehlende Semantik

Beziehung

Gebäude

Kennz.	Name	Post-Adresse
A	Haus Beuth	Lütticher Strasse 38, 13353 Berlin
B	Haus Gauß	Limburger Straße 20, 13353 Berlin
C	Haus Grashof	Luxemburger Straße 10, 13353 Berlin

RDM – Integritätsbedingungen

Lösungsansatz:

- Um **logische Widerspruchsfreiheit** zu erreichen sind bestimmte Einschränkungen notwendig
- Das Relationale Datenmodell offeriert hierfür zwei modellinhärente **Integritätsbedingungen**:
  - Primärschlüsselbedingung
  - Fremdschlüsselbedingung
- Durch das Formulieren (Benutzer) und Prüfen (DBMS) der Integritätsbedingungen werden die Attribut-Kombinationsmöglichkeiten derart eingeschränkt, dass sie logisch widerspruchsfrei und sinnvoll sind.

Ausgangsdefinitionen (1):

- **Was versteht man unter einem Primärschlüssel (PK)?**
  - Eine identifizierende Attributkombination für ein Relationenschema
  - Jedes Tupel einer Relation wird einzig und allein über diese (sinnvolle und minimale) Attributkombination eindeutig identifiziert
  - Alle nicht in dieser festgelegten Attributkombination eingeschlossenen Attribute eines Relationenschemas sind für die eindeutige Identifikation eines Tupels unerheblich.

Dozent

DozNr	Name	Vorname	Aufenthaltsort	Gebäude
123	Blankenburg	Marco	D139	D
123	Blankenburg	Marco	B131	B

Unerlaubte Tupelkombination, da zwei Tupel mit demselben **Primärschlüssel-Wert** (123) existieren und daher nicht mehr eindeutig sind.

  - Bei einem künstlichen PK werden Attribute in die Relation aufgenommen, die mit der abstrakten Beschreibung eines Anwendungsobjektes nichts zu tun haben, sondern „nur“ Verwaltungsinformationen sind.

Anmerkung: Manchmal existieren solche ein Objekt eindeutig identifizierenden Verwaltungsinformationen aber auch schon in der Realwelt und können als einfacher PK benutzt werden (z.B. Matrikelnummer eines Studenten, ISBN-Nummer eines Buches, Kennzeichen eines Landes)

Ausgangsdefinitionen (2):

- **Was versteht man unter einem Fremdschlüssel (FK)?**
  - Attribut, das in Bezug auf den Primärschlüssel einer anderen Relation (oder auch derselben) definiert ist. Attribut darf nur die Werte der aktuell zur Verfügung stehenden Primärschlüsselmenge annehmen.
  - Beispiel:
    - Dozent(DozNr, Name, Vorname, Aufenthaltsort, *Gebäude*)
    - Gebäude(Kennzeichen, Name, Postadresse)

Dozent

DozNr	Name	Vorname	Aufenthaltsort	*Gebäude*
123	Blankenburg	Marco	B131	B
124	Sauer	Petra	D139	D

Gebäude

Kennz.	Name	Post-Adresse
A	Haus Beuth	Lütticher Strasse 38, 13353 Berlin
B	Haus Gauß	Limburger Straße 20, 13353 Berlin
C	Haus Grashof	Luxemburger Straße 10, 13353 Berlin
D	Haus Bauwesen	Luxemburger Straße 9, 13353 Berlin

Erst durch Hinzufügen des Tupels mit dem Kennzeichen **D** in der Relation ‚Gebäude' erhält dieser Buchstabe eine Semantik und darf in der Relation ‚Dozent' als **Fremdschlüssel** verwendet werden.

RDM – Integritätsbedingungen

Primärschlüsselbedingung (Entity-Integrität):

- **Forderung**: Die Werte der Primärschlüsselattribute sind eindeutig und nicht „NULL“ (also besitzen einen Wert)
- Die **SQL-Syntax** (DDL) zum Formulieren der Bedingung beim Anlegen eines Datenbankobjektes (also einer Tabelle) lautet allgemein wie folgt:

```
CREATE TABLE <tabellenname>
(<spalte1> <datentyp> [[CONSTRAINT <name der bedingung>] PRIMARY KEY],
...,
[[CONSTRAINT <name der bedingung>] PRIMARY KEY (<spalte1>,...)] );
```

- Beispiel (als separate Integritätsbedingung):

```
CREATE TABLE Dozent
( DozNr CHAR(3),
  …,
  PRIMARY KEY (DozNr) );
```

- Beispiel (als Erweiterung der Spaltendefinition):

```
CREATE TABLE Dozent
( DozNr CHAR(3) PRIMARY KEY,
  ... );
```

Fremdschlüsselbedingung (Referentielle Integrität):

- **Forderung**: Zu jedem Wert eines Fremdschlüsselattributs einer Relation R2 muss ein Wert des Primärschlüssels in Relation R1 vorhanden sein
- Die **SQL-Syntax** (DDL) zum Formulieren der Bedingung beim Anlegen eines Datenbankobjektes (also einer Tabelle) lautet allgemein wie folgt:

```
CREATE TABLE <tabellenname>
(<spalte1> <datentyp> [[CONSTRAINT <name der bedingung>] REFERENCES <tabellenname> [<spalte>]],
...,
[[CONSTRAINT <name der bedingung>] FOREIGN KEY (<spalte1>,...) REFERENCES <tabellenname> [<spalte>]] );
```

- Beispiel (als separate Integritätsbedingung):

```
CREATE TABLE Prüfung
( DozentNr CHAR(3), StudentNr CHAR(6), …,
  PRIMARY KEY (DozentNr, StudentNr),
  FOREIGN KEY (DozentNr) REFERENCES Dozent(DozNr),
  FOREIGN KEY (StudentNr) REFERENCES Student(MatrNr) );
```

- Beispiel (als Erweiterung der Spaltendefinition):

```
CREATE TABLE Prüfung
( DozentNr CHAR(3) REFERENCES Dozent(DozNr),
  StudentNr CHAR(6) REFERENCES Student(MatrNr),
  …,
  PRIMARY KEY (DozentNr, StudentNr));
```

Regeln in Bezug auf Fremdschlüssel:

- FK und PK sind auf **gleichem** Wertebereich definiert
  (d.h. bspw: Ist PK vom Typ CHAR(3) so muss der FK auch vom Typ CHAR(3) sein)
- Wertebereich für FK ist **explizit** anzugeben
  (d.h. leitet sich nicht implizit aus dem Datentyp und Wertebereich des referenzierten Pimärschlüssels ab)
- FK und PK gestatten die (logische) Verknüpfung von Relationen
- FK kann auch Teil des PK der Relation sein
  (siehe bspw. Relation „Prüfung“ mit DozentNr und StudentNr)
- FK können Nichtschlüsselattribute (NSA) sein. In diesem Fall dürfen/können FK „NULL“-Werte aufweisen
- FK kann zusammengesetzt sein (wenn referenzierter PK zusammengesetzt ist)
- Relation kann mehrere FK besitzen, die die gleiche oder verschiedene Relationen referenzieren. Beispiel:

```
CREATE TABLE Mitarbeiter
( PersonalNr   INTEGER PRIMARY KEY,
  Name         CHAR(20),
  Vorname      CHAR(20),
  Vorgesetzter INTEGER REFERENCES Mitarbeiter(PersonalNr),
  Assistent    INTEGER REFERENCES Mitarbeiter(PersonalNr)
);
```

Überblick

- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Konvertierung von Spezialisierung
- Modell-inhärente Integritätsbedingungen
  - Entity Integrität (Primärschlüsselbedinung)
  - Referentielle Integrität (Fremdschlüsselbedingung)
  - Wahrung der referentiellen Integrität bei Datenmanipulation
- Normalformenlehre

Kapitel 2 des Lehrbuchs

Wahrung der Referentiellen Integrität bei Datenmanipulation (1):

- Kritische Datenmanipulationsoperationen für PK-FK Beziehungen:

Verschiedene Aktionen möglich

- Löschen (DELETE) eines Tupels in der referenzierten Relation
  Beispiel: Ein Dozent oder Student soll gelöscht werden, zu dem noch Prüfungsdaten vorhanden sind.
- Aktualisieren (UPDATE) eines PK Attributs in der referenzierten Relation
  Beispiel: DozNr des Dozenten Blankenburg in Relation ‚Dozent' soll von 123 zu 500 geändert werden und es gibt bereits Prüfungsdaten zu diesem Dozenten.

Verbot

- Aktualisieren (UPDATE) des FK Attributs in der referenzierenden Relation
  Beispiel: DozNr des Dozenten Blankenburg in Relation ‚Prüfung' soll von 123 zu 500 geändert werden.
- Einfügen (INSERT) eines Tupels in die referenzierende Relation
  Beispiel: Es soll ein Prüfungsdatum mit einem Prüfer (Dozenten) eingetragen werden, der noch nicht in der Relation ‚Dozent' existiert und dessen PK daher nicht referenziert werden kann.

Wahrung der Referentiellen Integrität bei Datenmanipulation (2):

- Der SQL-92 Standard (99, 2003, 2006) spezifiziert referentielle Aktionen, die ein DBMS für jeden FK bei einer DELETE oder UPDATE Operation ausüben kann:
  - ON DELETE {NO ACTION | RESTRICT | CASCADE | SET DEFAULT | SET NULL }
  - ON UPDATE {NO ACTION | RESTRICT | CASCADE | SET DEFAULT | SET NULL }

- Die referentielle Aktion kann bei der Spezifikation des Fremdschlüssels angegeben werden (muss jedoch durchs DBMS unterstützt werden):

CREATE TABLE <tabellenname>
(<*spalte1*> <*datentyp*> [[**CONSTRAINT** <*name der bedingung*>] **REFERENCES** <*tabellenname*> [<*spalte*>]
[{**ON UPDATE** | **ON DELETE**} <*name der aktion*>], …);

Wahrung der Referentiellen Integrität bei Datenmanipulation (3):

- Wird keine Aktion spezifiziert, so ist "NO ACTION" das Standardverhalten
- NO ACTION / RESTRICT: Operation ist beschränkt und wird nur ausgeführt, sofern (noch) keine referenzierenden Datensätze (i.e. FK-Werte) vorhanden sind. Ansonsten wird die Operation durch das DBMS zurückgewiesen.
  - No action: optimistisches Verfahren, ggf. Rollback erforderlich
  - Restrict: pessimistisches Verfahren
- CASCADE: Kaskadierende (i.e. weiterreichende) Operation
  - Beispiel: Wird Dozent gelöscht, werden auch alle Prüfungsdaten gelöscht
  - Beispiel: Wird die DozNr eines Dozenten von '123' auf '500' geändert, so werden auch automatisch alle Prüfungen, die Dozent '123' referenzierten auf '500' gesetzt.
- SET DEFAULT: Fremdschlüsselattribut wird auf den eingestellten Standardwert gesetzt
  - Beispiel: Wird ein Dozent gelöscht, werden alle Prüfungen für deren Betreuung er vorgesehen war (also, die ihn referenzierten) auf Dozent '123' eingestellt, wenn dies der Default-Wert ist
- SET NULL: Fremdschlüsselattribut wird auf NULL gesetzt (jedoch nur, wenn FK nicht gleichzeitig Teil eines PK ist, so wie bei der Relation 'Prüfung')

Beispiele zur Wahrung der Referentiellen Integrität (1):

- 1.) Was passiert mit den Prüfungsdaten, wenn ein Student exmatrikuliert wird?

```
CREATE TABLE Prüfung
(
   DozentNr CHAR(3),
   StudentNr CHAR(6),
   …,
   PRIMARY KEY (DozentNr, StudentNr),
   FOREIGN KEY (DozentNr) REFERENCES Dozent(DozNr),
   FOREIGN KEY (StudentNr) REFERENCES Student(MatrNr) ON DELETE CASCADE
);
```

Sinnvoll?

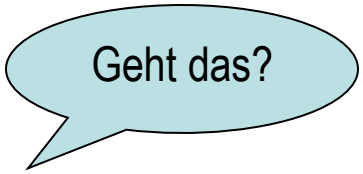
- 2.) Welche Veränderung zieht die Änderung der Matrikelnummer eines Studenten nach sich?

```
CREATE TABLE Prüfung
(
   DozentNr CHAR(3),
   StudentNr CHAR(6),
   …,
   PRIMARY KEY (DozentNr, StudentNr),
   FOREIGN KEY (DozentNr) REFERENCES Dozent(DozNr),
   FOREIGN KEY (StudentNr) REFERENCES Student(MatrNr) ON UPDATE CASCADE
);
```

Beispiele zur Wahrung der Referentiellen Integrität (2):

- 3.) Was passiert mit den Prüfungsdaten, wenn ein Professor emeritiert wird?

```
CREATE TABLE Prüfung
 (
    DozentNr CHAR(3),
    StudentNr CHAR(6),
    …,
    PRIMARY KEY (DozentNr, StudentNr),
    FOREIGN KEY (DozentNr) REFERENCES Dozent(DozNr) ON DELETE SET NULL,
    FOREIGN KEY (StudentNr) REFERENCES Student(MatrNr)
 );
```



- 4.) Was passiert mit den Prüfungsdaten, wenn ein Professor emeritiert wird?

```
CREATE TABLE Prüfung
 (
    DozentNr CHAR(3) DEFAULT '123',
    StudentNr CHAR(6),
    …,
    PRIMARY KEY (DozentNr, StudentNr),
    FOREIGN KEY (DozentNr) REFERENCES Dozent(DozNr) ON DELETE SET DEFAULT,
    FOREIGN KEY (StudentNr) REFERENCES Student(MatrNr)
 );
```

Überblick

- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Konvertierung von Spezialisierung
- Anlegen von Tabellen in der Datenbank
- Modell-inhärente Integritätsbedingungen
  - Entity Integrität (Primärschlüsselbedinung)
  - Referentielle Integrität (Fremdschlüsselbedingung)
  - Wahrung der referentiellen Integrität bei Datenmanipulation
- Normalformenlehre/Normalisierung

Redundanzen und Anomalien im RDM

Redundanz: Weitschweifigkeit, Überladung mit überflüssiger Information.

Bestellung	Lieferant	Name	Datum	Artikel	Bezeichnung	Anzahl
023	00102	Adler GmbH	02. 01. 2003	140	Topfix	200
118	10033	Bussard AG	05. 03. 2003	237	Primus	30
437	00102	Adler GmbH	28. 03. 2003	555	Leckerle	500
540	20987	Zobel & Co.	02. 04. 2003	140	Topfix	120
540	20987	Zobel & Co.	02. 04. 2003	555	Leckerle	180
543	20987	Zobel & Co.	07. 04. 2003	237	Primus	20

BESTELLUNG (Bestellung, Lieferant, Name, Datum, Artikel, Bezeichnung, Anzahl)

Grün gekennzeichneten Informationen sind redundant !

Redundanzen und Anomalien im RDM

Anomalie: Unregelmäßigkeit, Regelwidrigkeit - (1) Einfügeanormalie .

Bestellung	Lieferant	Name	Datum	Artikel	Bezeichnung	Anzahl
023	00102	Adler GmbH	02. 01. 2003	140	Topfix	200
118	10033	Bussard AG	05. 03. 2003	237	Primus	30
437	00102	Adler GmbH	28. 03. 2003	555	Leckerle	500
540	20987	Zobel & Co.	02. 04. 2003	140	Topfix	120
540	20987	Zobel & Co.	02. 04. 2003	555	Leckerle	180
543	20987	Zobel & Co.	07. 04. 2003	237	Primus	20
				679	Mopsfidel	

BESTELLUNG (Bestellung, Lieferant, Name, Datum, Artikel, Bezeichnung, Anzahl)

Fehlende Attribute bei Einfügung von Tupeln.

Redundanzen und Anomalien im RDM

Anomalie: Unregelmäßigkeit, Regelwidrigkeit - (2) Änderungsanormalie .

Bestellung	Lieferant	Name	Datum	Artikel	Bezeichnung	Anzahl
023	00102	Adler GmbH	02. 01. 2003	140	Topfix	200
118	10033	Bussard AG	05. 03. 2003	237	Primus	30
437	00102	Adler GmbH	28. 03. 2003	555	Kaufmich	500
540	20987	Zobel & Co.	02. 04. 2003	140	Topfix	120
540	20987	Zobel & Co.	02. 04. 2003	555	Kaufmich	180
543	20987	Zobel & Co.	07. 04. 2003	237	Primus	20

BESTELLUNG (Bestellung, Lieferant, Name, Datum, Artikel, Bezeichnung, Anzahl)

Mehrfach gespeicherte Attribute zwingen zur mehrfachen Änderung des Sachverhaltes.

Redundanzen und Anomalien im RDM

Anomalie: Unregelmäßigkeit, Regelwidrigkeit - (3) Löschanormalie.

Bestellung	Lieferant	Name	Datum	Artikel	Bezeichnung	Anzahl
023	00102	Adler GmbH	02. 01. 2003	140	Topfix	200
118	10033	Bussard AG	05. 03. 2003	237	Primus	30
437	00102	Adler GmbH	28. 03. 2003	555	Leckerle	500
540	20987	Zobel & Co.	02. 04. 2003	140	Topfix	120
540	20987	Zobel & Co.	02. 04. 2003	555	Leckerle	180
543	20987	Zobel & Co.	07. 04. 2003	237	Primus	20

BESTELLUNG (Bestellung, Lieferant, Name, Datum, Artikel, Bezeichnung, Anzahl)

Unzweckmäßiges Löschen von Attributen bei Löschung eines Tupels als semantische Einheit.
Die Firmeninformation "Lieferant Name" (10033 Bussard AG) geht beim Löschen der Bestellung verloren, obwohl möglicherweise bei diesem Lieferanten erneut bestellt wird.

Güte von Datenmodellen

Die Güte oder auch Qualität von Datenmodellen kann unterschiedlich sein und wird bestimmten Kriterien gemessen, wie z.B. an

dem Grad der Redundanzfreiheit,

dem Schutz vor potentiellen Inkonsistenzen (Anomalien),

der Übersichtlichkeit und leichten Handhabbarkeit des Datenmodelles und der Effizienz potentieller Datenzugriffe.

Zwei mögliche Wege führen zu "guten“ Datenmodellen

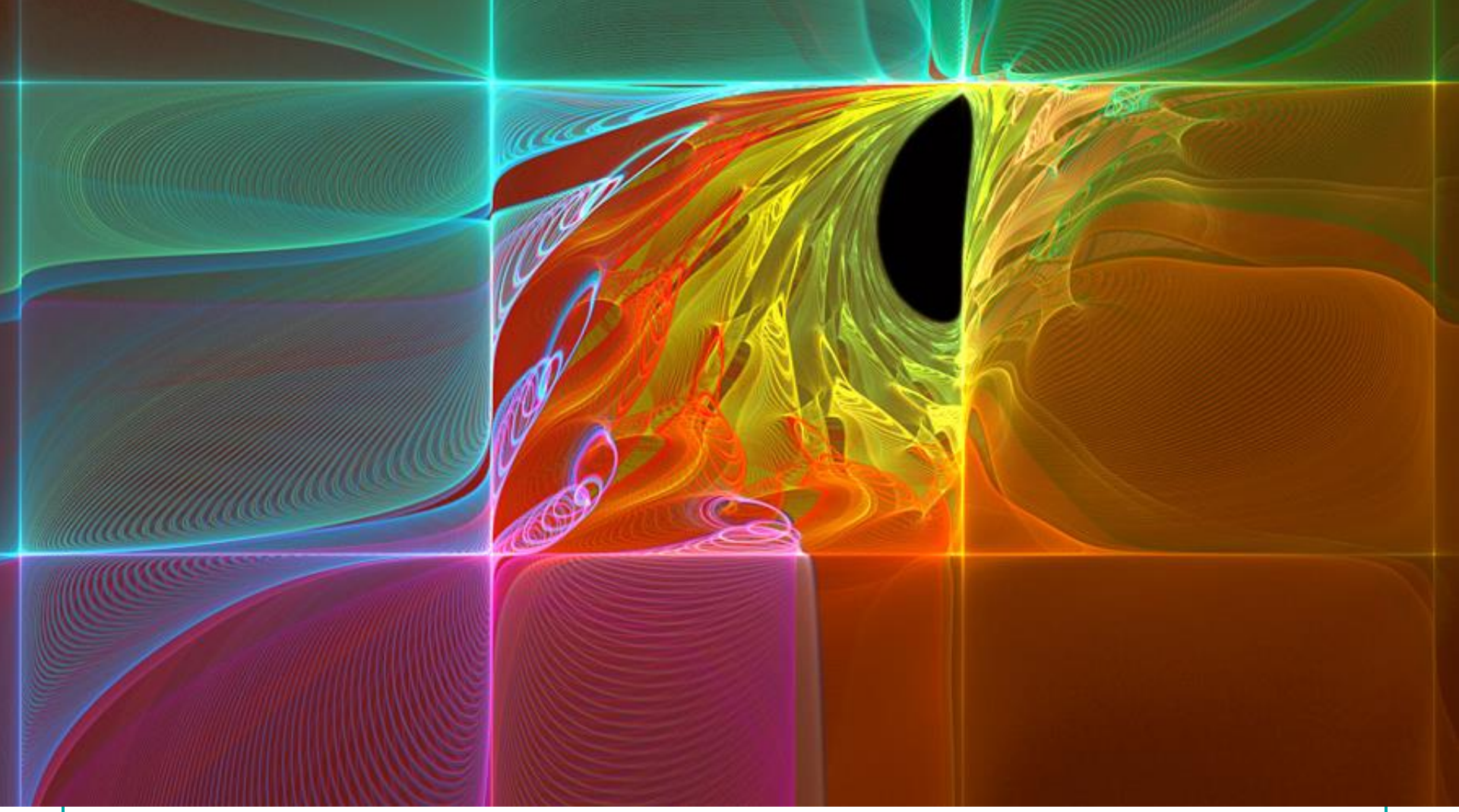
Normalisierung von Relationen
Ein vorgegebener Entwurf niedriger Güte soll verbessert werden.

Synthese von Relationen
Es soll von vornherein ein "optimales" Modell konstruiert werden.

Ausgangspunkte sind

- funktionale Abhängigkeiten zwischen Attributen und
- gegebene ggf. "schlechte" Relationen

- Semantische Beziehungen zwischen Datenobjekten (Entitys) und
- ermittelte Attribute der Entitys.



1.-3. Normalform

Der Prozess der Normalisierung

Das Normalisieren einer Relation (Tabelle) wird immer durch Aufteilung der Relation auf mehrere Relationen realisiert (Dekomposition).

Es wird so lange normalisiert, bis die nach der Aufteilung neu entstandenen Relationen der erwünschten Normalform entsprechen.

Beispiel:
Diese Relation befindet sich in 1NF →

Artikelcode	Bezeichnung	Lieferer	Datum	kg
0105	Zucker	Meyer	12.04.2003	100
0105	Zucker	Schulze	12.04.2003	150
0237	Mehl	Meyer	18.04.2003	200

Diese Relationen befinden sich in 3NF

Artikelcode	Lieferer	Datum	kg
0105	Meyer	12.04.2003	100
0105	Schulze	12.04.2003	150
0237	Meyer	18.04.2003	200

Artikelcode	Bezeichnung
0105	Zucker
0105	Zucker
0237	Mehl

Übersicht der Normalformen

Nicht-normalisierte Relationen

1. Normalform: Atomare Attribute (Keine Wiederholungsgruppen)

2. Normalform: Attribute voll funktional abhängig vom Primärschlüssel

3. Normalform: Unabhängigkeit der Nichtschlüsselattribute voneinander

(Boyce-CoddNF: Alle Abhängigkeiten voll funktional vom Schlüssel ausgehend)

Erste Normalform

Nur atomare Attribute (Keine Wiederholungsgruppen)

Relationen nicht in 1NF:

Vater	*Mutter*	*Kinder*
Johann	Martha	{Else, Lucie}
Johann	Maria	{Theo, Josef}
Heinz	Martha	{Cleo}

Umgewandelte Relationen in 1NF:

Vater	*Mutter*	*Kind*
Johann	Martha	Else
Johann	Martha	Lucie
Johann	Maria	Theo
Johann	Maria	Josef
Heinz	Martha	Cleo

Andere Umwandlungsmöglichkeit

R(Vater, Mutter, Kind1, Kind2)

Beispiel nach Alfons Kemper (TU München)

Funktionale Abhängigkeit

Ein Attribut A2 einer Relation R(A1,A2,...) heißt *funktional abhängig* vom Attribut A1 der Relation R, wenn zu jedem Wert a_i (a_i Є A1) höchstens ein Wert a_k (a_k Є A2) möglich ist.

Beispiel:

Artikelcode	Bezeichnung	Besteller
0105	Zucker	Meyer
0105	Zucker	Schulze
0237	Mehl	Meyer
0105	Zucker	Müller
2356	Salz	Moppel
2356	Salz	Meyer
0105	Zucker	Meyer

Darstellung: Artikelcode → Bezeichnung

Volle funktionale Abhängigkeit

Ein Attribut A2 einer Relation R(A1.1,A1.2,...A2,...) heißt voll funktional abhängig von einem zusammengesetzten Schlüssel von Attributen A1.n der Relation R, wenn A2 funktional abhängig vom Schlüssel der Attribute A1.n, nicht aber funktional abhängig von einem beliebigen Teilschlüssel A1.j ist.

Beispiel:

Artikelcode	Bezeichnung	Lieferer	Datum	kg
0105	Zucker	Meyer	12.04.2003	100
0105	Zucker	Schulze	12.04.2003	150
0237	Mehl	Meyer	18.04.2003	200
0105	Zucker	Müller	22.04.2003	50
2356	Salz	Moppel	29.04.2003	30
2356	Salz	Meyer	04.05.2003	50
0105	Zucker	Meyer	04.05.2003	75

Darstellung:(Artikelcode, Lieferer, Datum) ⇒ kg

Zweite Normalform

Eine Relation ist in zweiter Normalform falls gilt: Die Relation ist in der ersten Normalform und jedes Nichtschlüsselattribut der Relation ist voll funktional abhängig von jedem Schlüsselkandidaten der Relation.

Zeugnis

Schlüsselkandidat:

{Student, LV}

Student	**LV**	**Note**
Meyer	Datenbanken (B)	1,3
Müller	Data Warehouse (M)	2,7
Schulte	Data Mining (M)	3,3
Meyer	Big Data Analytics (M)	1,3

ZEUGNIS (STUDENT , LV , NOTE): NOTE ist voll funktional abhängig vom Primärschlüssel, da sie von beiden Schlüsselattributen abhängt.

In Zweiter Normalform?

VerkäuferIn

Bereich	**BerLeit**	**Kz**	**Sekr**	**Verkäuf**	**Gehalt**
Bekleid	Krause	Kr	Mai	Sommer	2500
Wäsche	Meyer	Me	Tai	Herbst	2600
Bekleid	Krause	Kr	Mai	Zobel	3000
Schnäp	Meyer	Me	Tai	Sommer	2300
. . . . .					

Schlüsselkandidat:

{Bereich, Verkäuf}

Annahmen: Jeder Bereich hat genau einen Leiter (m/w) und eine Sekretärin (m/w). Diese können aber jeweils mehreren Bereichen zugeordnet sein. Eine Verkäuferin (m/w) kann in maximal zwei Bereichen tätig sein. Das Gehalt wird bereichsspezifisch festgelegt.

Abhängigkeiten: Die Tabelle steht nicht in der zweiten Normalform, da *BerLeit* (Bereichsleiter), *Kz* (Kennzeichen) und *Sekr* (SekretärIn) nur vom *Bereich* abhängig sind.

Methodik: Zerlegen in weitere Tabellen

VerkäuferIn

Bereich	BerLeit	Kz	Sekr	**Verkäuf**	Gehalt
Bekleid	Krause	Kr	Mai	Sommer	2500
Wäsche	Meyer	Me	Tai	Herbst	2600
Bekleid	Krause	Kr	Mai	Zobel	3000
Schnäp	Meyer	Me	Tai	Sommer	2300
. . . . .					

Verkauf

Bereich	**Verkäuf**	**Gehalt**
Bekleid	Sommer	2500
Wäsche	Herbst	2600
Bekleid	Zobel	3000
Schnäp	Sommer	2300
. . . . .		

Bereiche

Bereich	BerLeit	Kz	Sekr
Bekleid	Krause	Kr	Mai
Wäsche	Meyer	Me	Tai
Schnäp	Meyer	Me	Tai
. . . . .			

1. Entfernen der Attribute, die nicht voll funktional vom Primärschlüssel abhängig sind.
2. Entfernte Attribute in neue Tabellen zusammenfassen, wobei die jeweils von einem Schlüssel abhängigen Attribute eine gesonderte Tabelle bilden.
3. In den neuen Tabellen werden die entsprechenden Schlüsselattribute hinzugefügt.

Create your Own Exam: Normalisierung

- Bitte erstellen Sie eine Multiple Choice Aufgabe zum Thema Normalisierung
  - Formulieren Sie eine Frage und 3 Antworten (A, B, C)
  - Davon sollte mindestens eine Antwort richtig und mindestens eine Antwort falsch sein

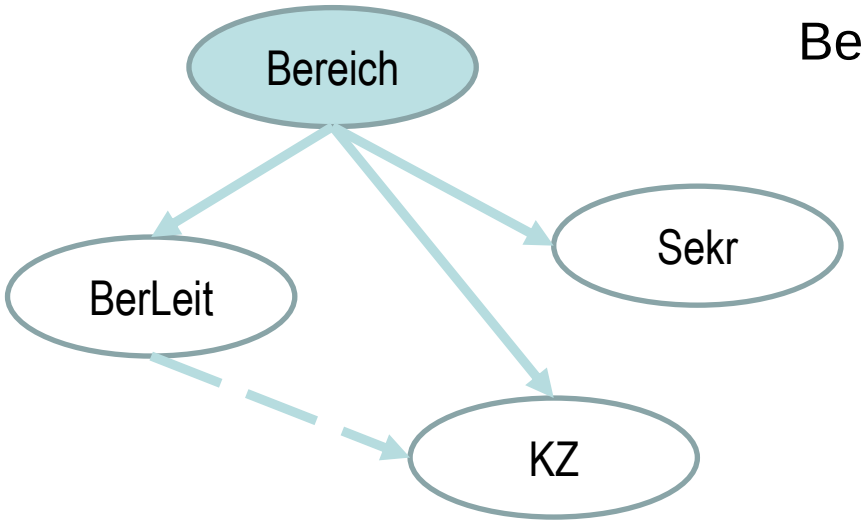
- Geben Sie die Aufgabe an Ihren rechten Nachbarn. Diskutieren Sie gemeinsam und markieren Sie die richtigen Lösungen

- Geben Sie am Ende der Vorlesung Ihre Aufgabe bei mir ab

5 min

Fällt Ihnen etwas auf?

Ist der Wertebereich C funktional abhängig von B und B ist funktional abhängig von A (A-->B-->C), dann besteht zwischen diesen Wertebereichen eine transitive Abhängigkeit.



Bereiche

Bereich	**BerLeit**	**Kz**	**Sekr**
Bekleid	Krause	Kr	Mai
Wäsche	Meyer	Me	Tai
Schnäp	Meyer	Me	Tai
. . . . .			

Das Kürzel Kz ist transitiv abhängig von dem Schlüssel Bereich.

Dritte Normalform

Eine Relation ist in dritter Normalform falls gilt:

- **die Relation ist in der zweiten Normalform und**
- **jedes Nichtschlüsselattribut der Relation ist nicht transitiv abhängig von jedem Schlüsselkandidaten der Relation.**

Bereiche

Bereich	**BerLeit**	**Kz**	**Sekr**
Bekleid	Krause	Kr	Mai
Wäsche	Meyer	Me	Tai
Schnäp	Meyer	Me	Tai
. . . . .			

Dritte Normalform - Synthesealgorithmus

1. Entfernen der Attribute (Spalten), die transitiv abhängig vom Schlüssel sind.
2. Anlegen der entfernten Attribute in neuen Tabellen, wobei die Attribute, die vom selben Nichtschlüsselattribut abhängig sind, in eine Tabelle kommen.
3. Hinzufügen des Nichtschlüsselattributes als Schlüssel in der neuen Tabelle.

Bereiche

Bereich	**BerLeit**	**Kz**	**Sekr**
Bekleid	Krause	Kr	Mai
Wäsche	Meyer	Me	Tai
Schnäp	Meyer	Me	Tai
. . . . .			

Bereiche

Bereich	**BerLeit**	**Sekr**
Bekleid	Krause	Mai
Wäsche	Meyer	Tai
Schnäp	Meyer	Tai
. . . . .		

Leiter

BerLeit	**Kz**
Krause	Kr
Meyer	Me
. . . . .	

Take Away Message

Übersicht der Normalformen

Nicht-normalisierte Relationen

1. Normalform: Atomare Attribute (Keine Wiederholungsgruppen)

2. Normalform: Attribute voll funktional abhängig vom Primärschlüssel

3. Normalform: Unabhängigkeit der Nichtschlüsselattribute voneinander

(Boyce-CoddNF: Alle Abhängigkeiten voll funktional vom Schlüssel ausgehend)

Fazit

Warum Normalisierung?

- Physische relationale können von schlechter Qualität sein.
- Ursache dafür sind mögliche Redundanzen und Anomalien.
- Doppelarbeiten und Fehlern im Umgang mit dem RDBMS resultieren.

Vorteile der Normalisierung:

- Anomalien werden durch Normalisierung weitestgehend beseitigt.

Nachteil der Normalisierung

- Einige Anfragen werden langsamer, da mehr Joins notwendig sind.

In der Praxis

- Von besonderer Bedeutung für die Modellierung von RDBMS ist die dritte Normalform (3NF).
- Es gibt verschiedene Standardmodelle die nicht normalisiert oder sogar übernormalisiert sind
- Vor- und Nachteile sind im Einzelfall abzuwägen.

Zusammenfassung und Ausblick

- Das Relationale Modell
- Von ER-Diagrammen zu Relationenschemata
- Normalformen und funktionale Abhängigkeiten

In der nächsten Veranstaltung:

- Relationale Algebra

Sie erinnern sich: Ihre Aufgaben

- **Aufgabe 1 Anfragen & Modellierung“**
  Denken Sie mal darüber nach, welche Anfragen Sie an die AOL Daten stellen möchten. Bitte Sie bitte ein logisches und physisches Schema zur Beantwortung dieser Anfragen.
- **Aufgabe 2 „SQL und Anfrageausführung“**
  Bitte formulieren Sie für Ihre Analyseideen aus 1.) die SQL Anfragen. Sie verstehen auch Möglichkeiten der Anfrageausführung bzw. Optimierung.
- **Aufgabe 3 „Datenintegration“**
  Zur Ausführung der Ausführung fehlen Ihnen noch externe Daten, z.B. aus dem Internet Archive, DMOZ oder Freebase.org. Bitte ergänzen Sie Ihr Schema und die Datenbasis.
- **Aufgabe 4 „Analyse, Erkenntnisgewinn und Wert“**
  Stellen Sie in 5 Minuten die wichtigsten Erkenntnisse aus den Daten vor. Bewerten Sie den Erkenntnisgewinn, z.B. gegenüber Ihren Kommilitonen oder der Literatur! Welche Erkenntnisse hätten einen kommerziellen Wert?

Die Themen

- Was sind Datenbanken?
  - Motivation, Historie, Datenunabhängigkeit, Einsatzgebiete
- Datenbankentwurf im ER-Modell & Relationaler Datenbankentwurf
  - Entities, Relationships, Kardinalitäten, Diagramme
  - Relationales Modell, ER -> Relational, Normalformen, Transformationseigenschaften
- Relationale Algebra & SQL
  - Kriterien für Anfragesprachen, Operatoren, Transformationen
  - SQL DDL, SQL DML, SELECT … FROM … WHERE …
- Datenintegration & Transaktionsverwaltung
  - JDBC, Cursor, ETL
  - Mehrbenutzerbetrieb, Serialisierbarkeit, Sperrprotokolle, Fehlerbehandlung, Isolationsebenen in SQL
- Ausblick
  - Map/Reduce, HDFS, Hive …
  - Wert von Daten

Ausgesuchte Vertiefungen

NF²-Relationen

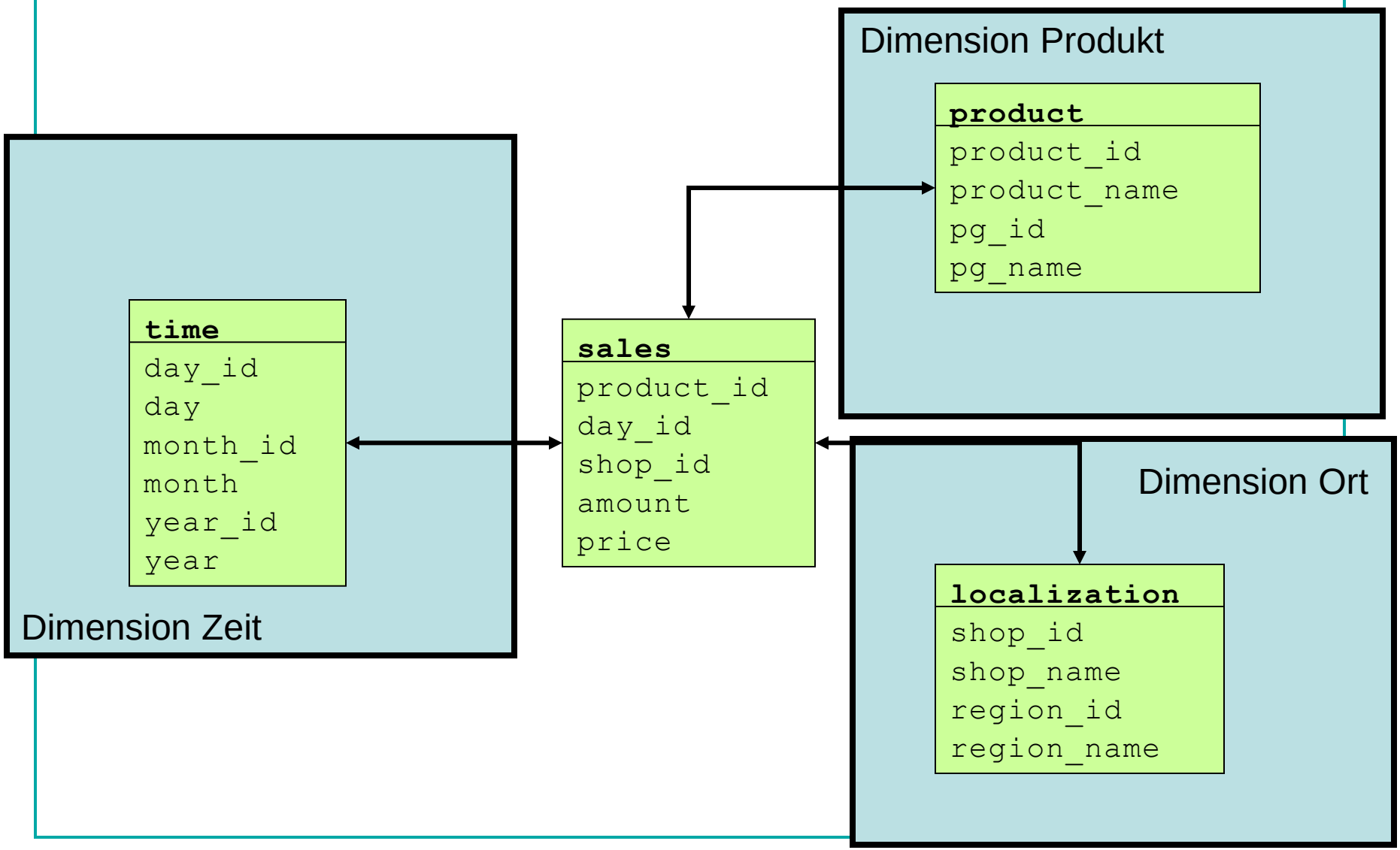
Non-First Normal-Form-Relation, Geschachtelte Relationen („Nested Tables“)
Siehe z.B. www.GoOLAP.info

goolap BETA
Search. Analyze. Decide.

Boing Competitors | Analyze

Company	Images	Competitors	Business Relations
Airbus		company: Airbus, Boeing (3)	company: Airbus, Shenzhen Airlines Limited, August 11, 2010, 2010-08-11, known Airbus, UBS Securities, known, $15.6 bln deal
Lear	LEAR CORPORATION	company: Lear, Boeing (2)	company: Lear, Cal State Stanislaus, known, speaking contract Lear, Simmons, known, last collaboration
E.A.D.S.	EADS	company: EADS, Boeing (2)	company: EADS, Boeing, known, refueling tanker contract EADS, Northrop Grumman Corporation, announced, contract EADS, Northrop Grumman Corporation, known, contract
Lockheed Martin	LOCKHEED MARTIN	company: Lockheed Martin, United Launch Alliance (1)	company: Lockheed Martin, VIKING Life-Saving Equipment A/S, known, Joint Development Agreement Lockheed Martin, Urban Operations Training Systems, known, indefinite delivery/indefinite quantity contract Lockheed Martin, Northrop Grumman 08, known, 23 mln usd satellite upgrade contract Lockheed Martin, Scaled Composites, August 31, 2006, 2006-08-31, known, contract

Dimensionen und Fakten- Star Schema



ÜBUNG

Diskussionsbeispiel

- Tabelle „Artikellieferung“ (unnormalisiert)

Artikellieferung

ArtikelNr.	Bezeichnung	Menge	LieferantNr.	Name	Ort
1	Milch	100 250 400	345 720 600	Müller Schulze Zott	12207 Berlin 14476 Golm 12207 Berlin
2	Orangensaft	85	345	Müller	12207 Berlin
3	Joghurt	500 250	345 600	Müller Zott	12207 Berlin 12207 Berlin
4	Quark	1000	600	Zott	12207 Berlin
5	Butter	500	720	Schulze	14476 Golm

Array-Datentyp

Name
{Müller, Schulze, Zott}

Die interne Repräsentation des mengenwertigen Attributwerts könnte z.B. so aussehen.

„12207 Berlin“ repräsentiert einen zusammengesetzten Wertebereich, bestehend aus einer 5-stelligen Zahlenfolge für die Postleitzahl und einer beliebigen Zeichenkette für die Ortsbezeichnung

Diskussionsbeispiel in 1NF:

Wo verletzt das Beispiel bisher die 1NF?

- Die Attribute *Menge*, *LieferantenNr.*, *Name* und *Ort* enthalten mengenartige Werte
- Die Elemente der Wertemengen in *Ort* lassen sich zudem in die atomaren Attribut-wertebereiche für die Postleitzahl (z.B. mit Datentyp Integer) und für den Ortsnamen (z.B. CHAR(25)) aufteilen.

Lösung:

ArtikelNr.	Bezeichnung	Menge	LieferantNr.	Name	PLZ	Ort
1	Milch	100	345	Müller	12207	Berlin
1	Milch	250	720	Schulze	14476	Golm
1	Milch	400	600	Zott	12207	Berlin
2	Orangensaft	85	345	Müller	12207	Berlin
3	Joghurt	500	345	Müller	12207	Berlin
3	Joghurt	250	600	Zott	12207	Berlin
4	Quark	1000	600	Zott	12207	Berlin
5	Butter	500	720	Schulze	14476	Golm

Achtung: Nun ein zusammengesetzter Primärschlüssel nötig, da sonst die **Tupel**-Eindeutigkeitsforderung des RDM verletzt wäre.

Diskussionsbeispiel in 2NF:

Wo verletzt das 1NF-Beispiel die 2NF?

- Die Attribute *Name, PLZ* und *Ort* sind funktional abhängig von *LieferantNr* und damit nicht voll funktional abhängig vom aktuellen Primärschlüssel {*ArtikelNr*, *LieferantNr*}
- Das Attribut *Bezeichnung* ist funktional abhängig von *ArtikelNr* und damit nicht voll funktional abhängig vom aktuellen Primärschlüssel {*ArtikelNr*, *LieferantNr*}

Lösung:

Lieferant

LieferantNr.	Name	PLZ	Ort
345	Müller	12207	Berlin
600	Zott	12207	Berlin
720	Schulze	14476	Golm

Artikel

ArtikelNr.	Bezeichnung
1	Milch
2	Orangentrunk
3	Joghurt
4	Quark
5	Butter

Lieferung

ArtikelNr.	LieferantNr.	Menge
1	345	100
1	720	250
1	600	400
2	345	85
3	345	500
3	600	250
4	600	1000
5	720	500

Aufteilen der bisher nicht voll funktional abhängigen NSA in zwei zusätzliche Relationen, in denen sie voll funktional abhängig werden.

Diskussionsbeispiel in 3NF:

Wo verletzt das 2NF-Beispiel die 3NF?

- In der Relation Lieferant treten der Ort „Berlin“ und die Postleitzahl „12207“ jeweils zweimal auf. Das NSA *Ort* ist somit über das NSA *PLZ* transitiv abhängig vom Schlüsselattribut *LieferantNr* (Formal: LieferantNr → PLZ → Ort)

Lösung:

Lieferant

LieferantNr.	Name	*PLZ*
345	Müller	12207
600	Zott	12207
720	Schulze	14476

Adresse

PLZ	Ort
12207	Berlin
14476	Golm

Lieferung

ArtikelNr.	LieferantNr.	Menge
1	345	100
1	720	250
1	600	400
2	345	85
3	345	500
3	600	250
4	600	1000
5	720	500

Artikel

ArtikelNr.	Bezeichnung
1	Milch
2	Orangentrunk
3	Joghurt
4	Quark
5	Butter

Auslagern des transitiv abhängigen NSA in eigene Relation, in der das „Transitions-NSA“ als Primärschlüssel auftritt. In der ursprünglichen Relation wandelt sich das „Transitions-NSA“ in einen FK.

Abschließende Bemerkungen

Daumenregeln

- Relationen in 1NF, die einen aus *nur einem* Attribut bestehenden Primärschlüssel besitzen sind automatisch in 2NF (da alle NSA automatisch voll funktional abhängig vom PK sind)
- Relationen in 2NF, die nur Schlüsselattribute und/oder maximal ein NSA enthalten, sind automatisch in 3NF

Gedächtnisstütze

- „Der Schlüssel, der ganze Schlüssel und nix als der Schlüssel - so wahr mir Codd helfe“

Denormalisierung

- Denormalisierung = *geplantes* Zurückgehen von hohem auf niedrigeres Normalisierungsniveau für *ausgewählte* Relationen.
- In bestimmten Anwendungsfällen kann es sinnvoll sein, auf Normalisierung zu verzichten oder sie durch Denormalisierung rückgängig zu machen, um
  - Anfragen generell zu vereinfachen (keine komplexen Querykonstrukte)
  - Die Verarbeitungsgeschwindigkeit von Anfragen zu erhöhen (JOINS sind rechenintensiv)
  - Besonderheiten von Geschäftsprozessen abzubilden