Prof. Dr.-Ing. Johannes Schildgen
johannes.schildgen@oth-regensburg.de

Datenbanken

Kapitel 3: Das Relationenmodell

OTH REGENSBURG

14

Relation (= Tabelle)

Metadaten

- Name der Relation (= Tabellennname)
- Attribute (= Spalten)
- Datentypen und Eigenschaften der Attribute (Primärschlüssel, Fremdschlüssel, ...)

Daten

- Menge von Tupeln (= Zeilen)
- Tupel besitzt einen Wert in jedem Attribut

Relation: Produkte

Metadaten

PRODUKTE(produktnummer,bezeichnung,preis,hersteller)

W(produktnummer) = integer,

W(bezeichnung) = string, usw.

Daten

PRODUKTE = {(17, Schokoriegel, 0.89, Monsterfood), ...}

Der Primärschlüssel (Produktname) wird unterstrichen. W(A) ist der Wertetyp des Attributs A.

Relation = Menge von Tupeln

$R(A1,A2,...) \subseteq W(A1) \times W(A2) \times ...$

- Mengen haben keine Duplikate
- Mengen haben keine Ordnung
- Ordnung der Attribute spielt keine Rolle

Die in einer Relation befindlichen Tupel sind eine Teilmenge aus dem kartesischen Produkts der Datentypen ihrer Attribute. Mengen haben keine Ordnung, d. h. es gibt keine erste, zweite, etc. Zeile. Auch auf Attributen besteht keine Ordnung. Es gibt kein erstes Attribut. Attribute werden über ihren Attributsnamen identifiziert. Mengen sind frei von Duplikaten, d. h. jede Zeile ist eindeutig.

NULL-Werte

NULL = nicht vorhandener Wert

produkte

produktnr	bezeichnung	preis
17	Schokoriegel	0.89
88	Katzenfutter	-

Mögliche Bedeutungen für Preis IS NULL:

- Der Preis ist unbekannt
- Das Produkt ist ausverkauft
- Produkte dieser Art haben keinen Preis
- Preis nur auf Anfrage

In unseren Beispieltabellen stellen wir NULL-Werte als "-" dar. Wenn der Preis 0 ist, ist das Produkt kostenlos, Preis NULL hat jedoch eine andere Bedeutung.

Primärschlüssel

PK ⊆ {A1, A2, ...}

- Primärschlüssel identifiziert Tupel eindeutig
- Es darf Relationen ohne Primärschlüssel geben
- Primärschlüssel ist eindeutig:
  Es darf keine zwei verschiedene Tupel in der Relation geben, die in den Primärschlüsselattributen die gleichen Werte haben.
- Primärschlüssel dürfen keine NULL-Werte enthalten

Der Primärschlüssel einer Relation ist eine Teilmenge ihrer Attribute.

Beispiel: hersteller

hersteller

firma	**land**
Calgonte	Italien
Monsterfood	USA
Sonstige	-

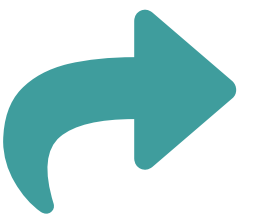
- Der Firmenname eines Herstellers ist eindeutig
- Der Firmenname darf nicht NULL sein
- Das Land darf NULL sein

Hier wird die Beziehung "Produkte sind von Hersteller" mittels einer Fremdschlüsselbeziehung modelliert. Die Attributmenge {hersteller} referenziert die Attributmenge {firma} der Herstellertabelle. Die Datentypen von den Spaten "hersteller" und "firma" müssen übereinstimmen. In die Spalte "hersteller" dürfen nur Werte stehen, die auch tatsächlich in der Firma-Spalte der Hersteller-Relation existieren.

7

Fremdschlüssel

15



- Mit Fremdschlüsseln werden Beziehungen Wert-basiert modelliert
- Fremdschlüssel referenziert Attributmenge
- Fremdschlüsselattribute haben die gleichen Datentypen wie die referenzierten Attribute
- In Fremdschlüsselattributen dürfen nur Werte stehen, die auch tatsächlich in der referenzierten Relation in den referenzierten Attributen existieren; NULL ist aber auch erlaubt

Beispiel: produkte und hersteller

16

produkte

produktnr	bezeichnung	preis	hersteller
17	Schokoriegel	0.89	Monsterfood
29	Spülmaschinentabs	3.99	Calgonte

hersteller

firma	land
Calgonte	Italien
Monsterfood	USA

produkte.hersteller ist Fremdschlüssel auf hersteller.firma

Zusammengesetzte Schlüssel

hersteller

firma	land
Calgonte	Italien
Monsterfood	USA
Monsterfood	China

produkte

produktnr	bezeichnung	preis	hersteller	land
17	Schokoriegel	0.89	Monsterfood	USA
29	Spülmaschinentabs	3.99	Calgonte	Italien

produkte(hersteller,land) ist Fremdschlüssel auf hersteller(firma,land)

In diesem Beispiel haben wir als Primärschlüssel der Herstellerrelation die Kombination aus den Spalten Firma und Land gewählt. Nun darf es also zwei Hersteller mit dem gleichen Firmennamen geben, vorausgesetzt sie sind von einem unterschiedlichen Land. Alle Relationen, die die Hersteller-Relation referenzieren (hier: Produkte) müssen dementsprechend zusammengesetzte Fremdschlüssel verwenden.

Zusammengesetzte Schlüssel

termine

Datum	**Uhrzeit**	**Raum**	**Dauer**	**Bezeichnung**
2020-10-14	14:15	17-123	90	Dings-Meeting
2020-10-21	16:00	17-222	60	Treffen mit Jürgen

personen

PersNr	**Name**
4	Ute
5	Peter
8	Anna

nehmen_teil

PersNr	**Datum**	**Uhrzeit**	**Raum**
5	2020-10-14	14:15	17-123
8	2020-10-14	14:15	17-123
5	2020-10-21	16:00	17-222

nehmen_teil.persnr ist Fremdschlüssel auf personen.persnr

nehmen_teil(datum,uhrzeit,raum) ist Fremdschl. auf termine(datum,uhrzeit,raum)

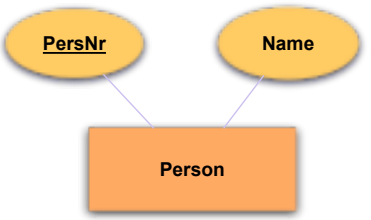
Hier wird eine N:M-Beziehung "Personen nehmen an Terminen teil" mittels einer nehmen_teil-Relation modelliert. Peter nimmt an der Vorlesung und am Praktikum teil.

Transformation ER-Diagramm → Relationenmodell

ER-Diagramm		**Relationenmodell**
Entitätstyp	→	Relation (=Tabelle)
Attribut	→	Attribut (=Spalte)
Primärschlüssel	→	Primärschlüssel
Sub-Attribute	→	Einzelne Attribute
Mehrwertiges Attribut	→	Relation
1:N-Beziehung	→	Fremdschlüssel
N:M-Beziehung	→	Relation
Schwache Entitätstypen	→	Relation
Generalisierung	→	Relation(en)

Entitätstyp ⟶ Relation

Entitätstyp	→	Relation (=Tabelle)
Attribut	→	Attribut (=Spalte)
Primärschlüssel	→	Primärschlüssel

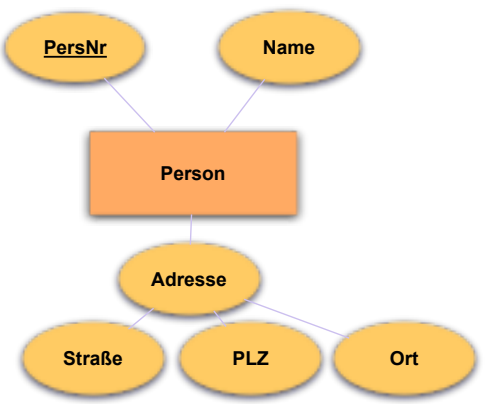


personen

PersNr	**Name**
4	Ute
5	Peter
8	Anna

Jeder Entitätstyp des ER-Diagramms wird in eine Relation überführt. Für jedes Attribut gibt es eine Spalte in der Relation, Primärschlüsselattribute sind genau wie im ER-Diagramm unterstrichen.

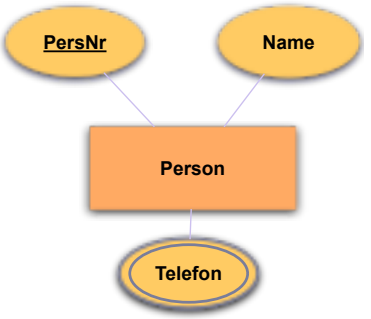
Sub-Attribute ⟶ Einzelne Attribute



Personen(
PersNr,
Name,
Adresse_Strasse,
Adresse_PLZ,
Adresse_Ort
)

Die einfachste Möglichkeit, Unterattribute im Relationenmodell abzubilden, ist es die Attributshierarchie flachzuklopfen. Da Attribute im Relationenmodell atomare Werte haben, erstellen wir für jedes Sub-Attribut eine eigene Spalte.

Mehrwertiges Attribut ⟶ Relation



Personen

PersNr	Name
4	Ute
5	Peter
8	Anna

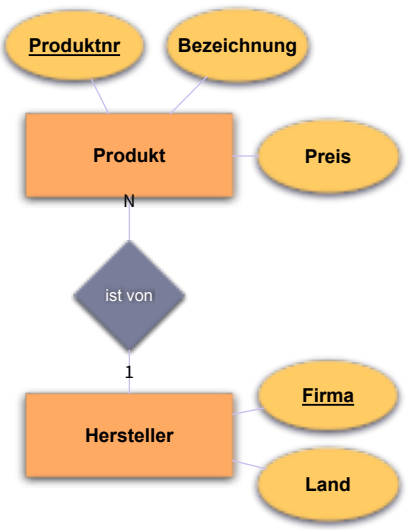
Telefonnummern

PersNr	Telefon
4	0151-1
4	0151-2
5	0151-3

telefonnummern.persnr ist Fremdschlüssel auf personen.persnr

Ute hat zwei Telefonnummer, Peter nur eine und Anna gar keine.

1:N-Beziehung ⟶ Fremdschlüssel



produkte

produktnr	bezeichnung	preis	hersteller
17	Schokoriegel	0.89	Monsterfood
18	Müsliriegel	1.19	Monsterfood
88	Katzenfutter	4.99	-

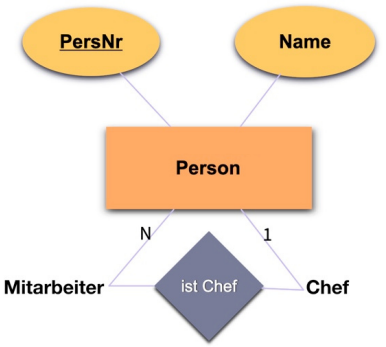
hersteller

firma	land
Holzkopf	Österreich
Monsterfood	USA

produkte.hersteller ist Fremdschlüssel auf hersteller.firma

An die Relation, die im ER-Diagramm an der gegenüberliegenden Seite von der 1 steht, wird ein Fremdschlüssel hinzugefügt. Und zwar hat dieser die gleichen Spaltentypen wie der referenzierte Primärschlüssel. Der Fremdschlüssel-Spaltenname ist hier der Name der referenzierten Tabelle, man kann aber auch den Namen der referenzierten Spalte (Firma) oder den Beziehungsnamen (sind_von) nehmen.

Rekursive Beziehung → Fremdschl.



personen

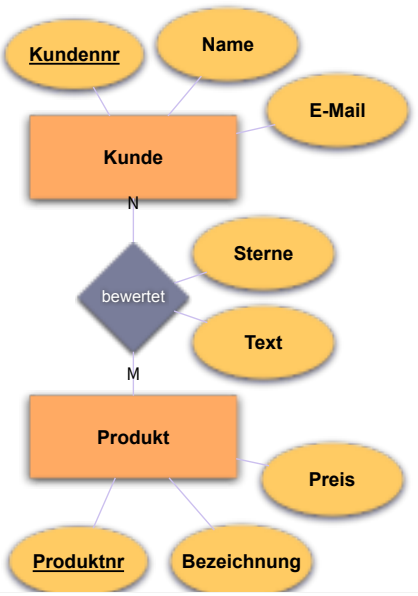
PersNr	Name	Chef
4	Ute	-
5	Peter	4
8	Anna	5

personen.chef ist Fremdschlüssel auf personen.persnr

Personen ist nun eine sich selbst referenzierende Tabelle. In der Fremdschlüsselspalte "Chef" ist die ID des Chefs einer Person zu finden. Ute hat keinen Chef, daher ist bei ihr Chef NULL. Es wäre sogar möglich, dort die eigene PersNr einzutragen.

N:M-Beziehung → Relation

17



kunden_bewerten_produkte

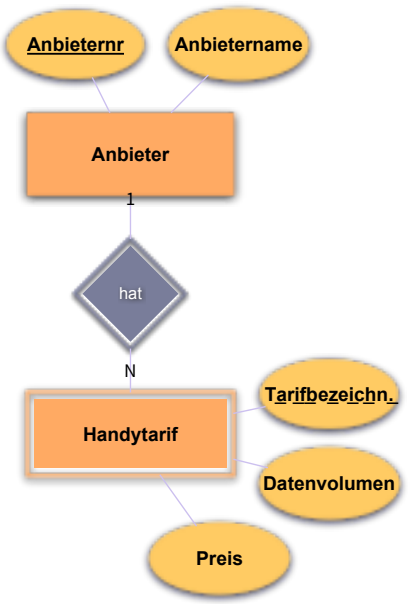
kundennr	produktnr	sterne	text
5	17	4	Guter Schokoriegel

kunden_bewerten_produkte.kundennr
ist Fremdschlüssel auf kunden.kundennr

kunden_bewerten_produkte.produktnr
ist Fremdschlüssel auf produkte.produktnr

Aus einer N:M-Beziehung wird eine eigene Relation. Diese trägt als Namen z. B. den Beziehungsnamen (bewerten) oder etwas anderes, was verständlich ist (kunden_bewerten_produkte, bewertungen, ...). Die neue Relation besitzt Fremdschlüsselspalten, welche die Primärschlüssel der an der Beziehung teilnehmenden Entities referenzieren. Die Kombination all dieser Fremdschlüsselspalten bilden den Primärschlüssel der Beziehungstabelle. Als weitere Nicht-Schlüssel-Attribute werden die Beziehungsattribute - sofern vorhanden - hinzugefügt.

Schwache Entitätstypen → Relation



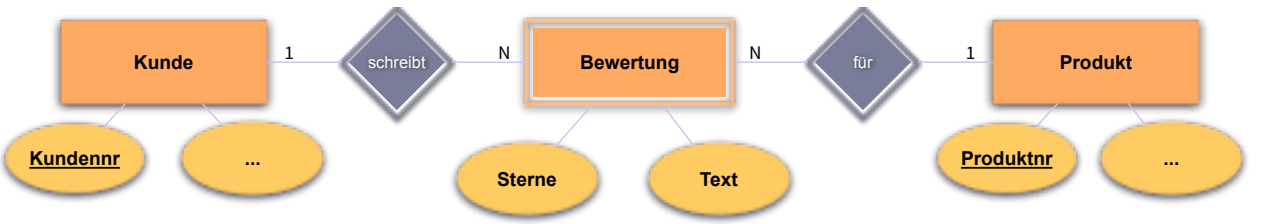
Anbieter(
<u>Anbieternr</u>,
Anbietername
)

Handytarife(
<u>Anbieternr</u>,
<u>Tarifbezeichnung</u>,
Datenvolumen,
Preis
)

handytarife.anbieternr ist Fremdschlüssel auf anbieter.anbieternr

Der schwache Entitätstyp erbt den Primärschlüssel von anderen Entitätstypen. Hier erbt Handytarife den Primärschlüssel von Anbieter, zusätzlich wird er erweitert um die Tarifbezeichnung.

Schwache Entitätstypen → Relation

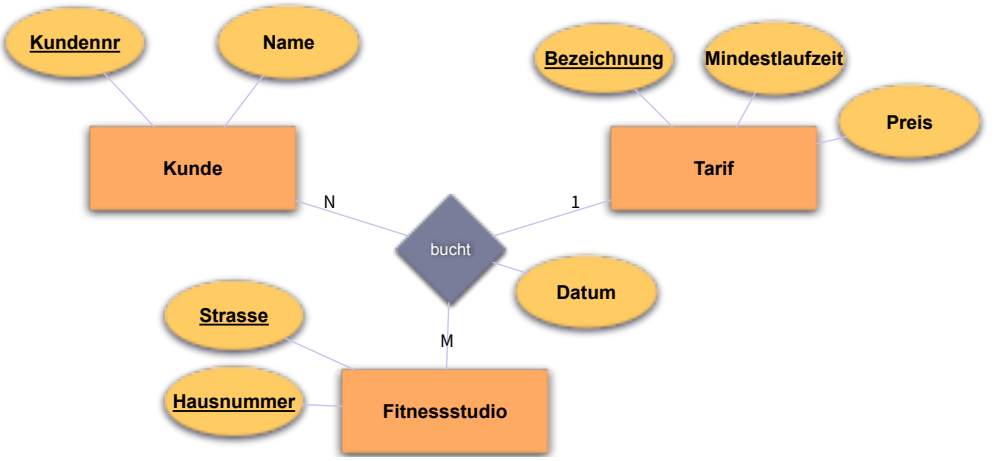


Kunden(<u>Kundennr</u>, ...)
Produkte(<u>Produktnr</u>, ...)
Bewertungen(<u>Kundennr, Produktnummer</u>, Sterne, Text)

bewertungen.kundennr ist Fremdschlüssel auf kunden.kundennr
bewertungen.produktnr ist Fremdschlüssel auf produkte.produktnr

Der schwache Entitätstyp Bewertung ist von zwei Entitätstypen existenzabhängig: Kunden und Produkte. Die Bewertungen-Relation hat also als Primärschlüssel die Kombination aus den Primärschlüsseln ebendieser beider Tabellen: Kundennr, Produktnr.
Das Resultat ist genau das gleiche wie die Relation, die aus der N:M-Beziehung "bewerten" entstanden ist (siehe 2 Folien zurück).

Ternäre Beziehung → Relation

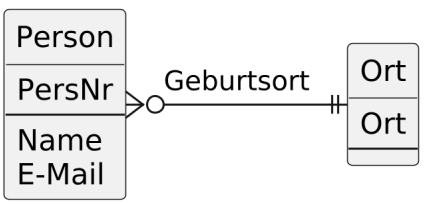


Buchung(Kundennr, Studio_Str, Studio_Hausnr, Tarif_Bezeichnung, Datum)

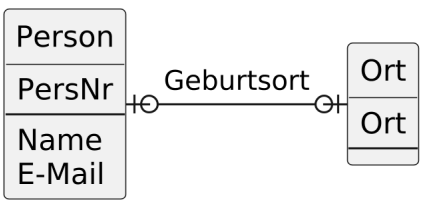
buchung.kundennr ist Fremdschlüssel auf kunden.kundennr
buchung.tarif_bezeichnung ist Fremdschlüssel auf tarife.bezeichnung
buchung(studio_str, studio_hausnr) ist FK auf fitnessstudios(strasse, hausnummer)

Genau wie bei einer binären N:M-Beziehung, wird auch bei einer höhergradigen Beziehung eine separate Relation erstellt. Der Primärschlüssel wird nur aus den Entitätstypen gebildet, an denen keine 1 steht.

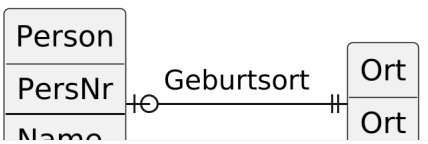
NOT NULL / UNIQUE



⇒ Geburtsort ist NOT NULL



⇒ Geburtsort ist UNIQUE

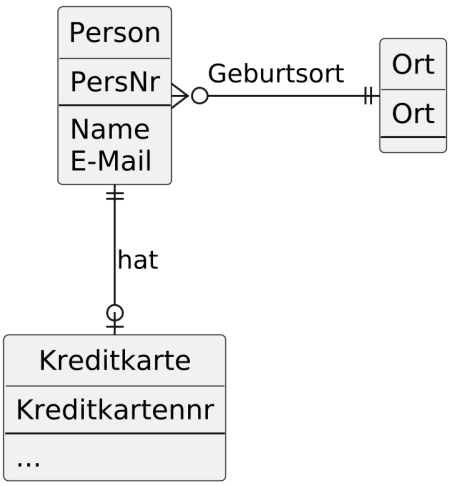


⇒ Geburtsort ist UNIQUE NOT NULL

Wenn im ER-Diagramm die Krähenfuß-Notation verwendet wird, können die genaueren Kardinalitätsrestriktionen ins Relationenmodell übernommen werden. Im ersten und dritten Diagramm sind Personen in genau einem Ort geboren, d. h. die Geburtsort-Spalte muss NOT NULL sein. NOT NULL heißt, es darf keine NULL-Werte in der Spalte geben. UNIQUE heißt, dass keine doppelten Werte vorkommen dürfen. Das wäre der Fall, wenn in einem Ort nur eine Person geboren sein dürfte.

NOT NULL / UNIQUE

Personen(
<u>PersNr</u>,
Name,
E-Mail,
Geburtsort NOT NULL,
Kreditkarte UNIQUE
)

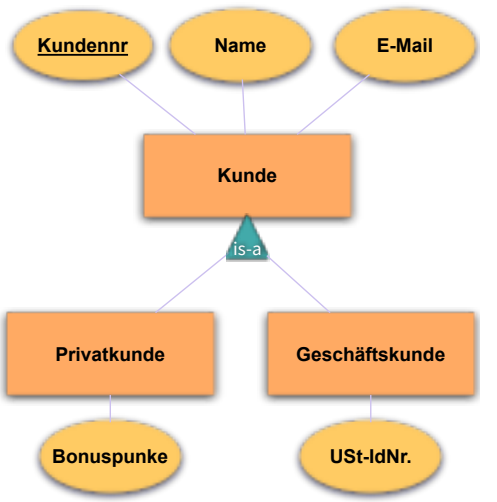


Personen brauchen einen Geburtsort, daher muss die Spalte NOT NULL sein. Da jede Kreditkarte nur einmal verwendet werden darf, ist die Fremdschlüsselspalte "Kreditkarte" UNIQUE. Der "o" im ER-Diagramm bei Kreditkarten gibt an, dass es Personen ohne Kreditkarte geben darf, daher sind NULL-Werte in der Kreditkarten-Spalte erlaubt.

Generalisierung im Relationenmodell

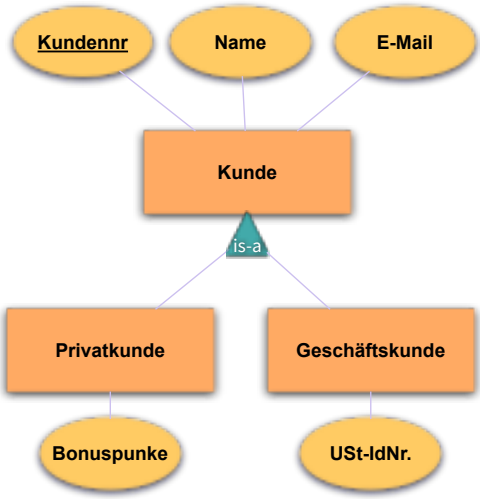
Mehrere Möglichkeiten der Umsetzung:

- Volle Redundanz
- Hausklassenmodell
- Vertikale Partitionierung
- Hierarchierelation



Volle Redundanz

- Jeder Entitätstyp wird zur eigenständigen Relation (alle Spalten)
- Beim Einfügen in Sub-Relationen wird redundante Information in die entsprechenden Super-Relationen eingefügt.



Kunden(<u>Kundennr</u>, Name, E-Mail)
Privatkunden(<u>Kundennr</u>, Name, E-Mail, Bonuspunkte)
Geschäftskunden(<u>Kundennr</u>, Name, E-Mail, USt-ID)

Privatkunden.Kundennr und Geschäftskunden.Kundennr sind Fremdschlüssel auf Kunden.Kundennr.

Durch die Fremdschlüsselbeziehungen wird garantiert, dass die Zeile auch in der Über-Relation existiert.

Volle Redundanz

kunden

kundennr	**name**	**email**
4	Ute	ute@example.com
5	Peter	peter@example.com
8	Anna	anna@example.com

privatkunden

kundennr	**name**	**email**	**bonuspunkte**
5	Peter	peter@...	2811

geschaeftskunden

kundennr	**name**	**email**	**ust_id**
8	Anna	anna@...	555-12-3-456789

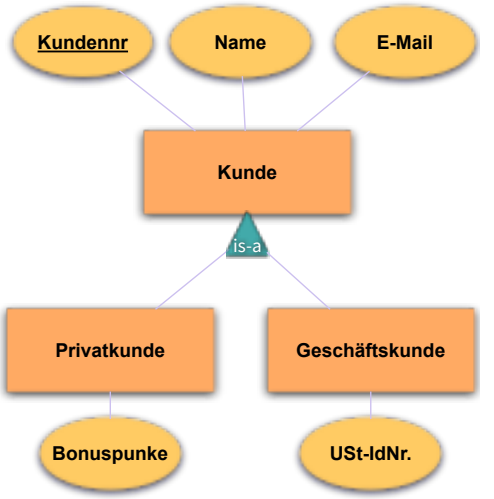
Peter ist Privatkunde, Anna ist Geschäftskunde und Ute einfach nur Kunde. Beim Einfügen, Ändern und Löschen von Kunden muss sorgfältig darauf geachtet werden, dass diese Operationen konsistent auf allen betreffenden Tabellen erfolgen. Daher ist diese Variante in der Regel nicht empfehlenswert.

Hausklassenmodell

- Jeder Entitätstyp wird zur eigenständigen Relation (alle Spalten)
- Es wird nur in die speziellste Relation eingefügt.

Kunden(Kundennr, Name, E-Mail)
Privatkunden(Kundennr, Name, E-Mail, Bonuspunkte)
Geschäftskunden(Kundennr, Name, E-Mail, USt-ID)

Hier keine Fremdschlüssel.



Im Hausklassenmodell ist die Suche aufwändig, da diese häufig mehrere Relationen betreffen kann.

Hausklassenmodell

kunden

kundennr	**name**	**email**
4	Ute	ute@example.com

privatkunden

kundennr	**name**	**email**	**bonuspunkte**
5	Peter	peter@...	2811

geschaeftskunden

kundennr	**name**	**email**	**ust_id**
8	Anna	anna@...	555-12-3-456789

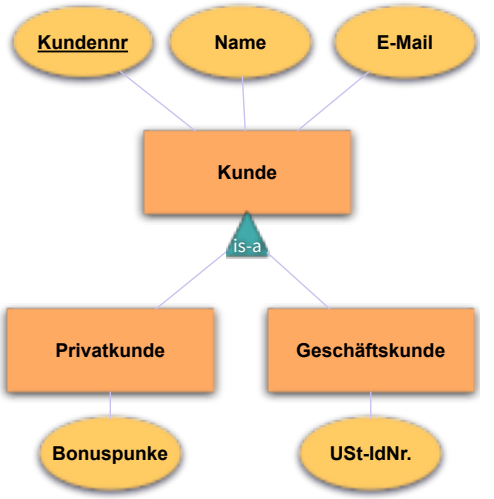
Peter ist Privatkunde, Anna ist Geschäftskunde und Ute einfach nur Kunde. Wollen wir nun alle Kunden finden, muss eine Vereinigung der drei Relationen gebildet werden.

Vertikale Partitionierung

- Jeder Entitätstyp wird zur eigenständigen Relation (PK + spezielle Spalten)

Kunden(Kundennr, Name, E-Mail)
Privatkunden(Kundennr, Bonuspunkte)
Geschäftskunden(Kundennr, USt-ID)

Privatkunden.Kundennr und Geschäftskunden.Kundennr sind Fremdschlüssel auf Kunden.Kundennr.



Lediglich die Primärschlüsselwerte sind in dieser Variante redundant. Alles andere wird in den speziellen Relationen gespeichert. Zur Suche sind oft Verbundoperationen nötig, da Daten ein und derselben Entität über mehrere Relationen verteilt (partitioniert) gespeichert werden.

Vertikale Partitionierung

kunden

kundennr	**name**	**email**
4	Ute	ute@example.com
5	Peter	peter@example.com
8	Anna	anna@example.com

privatkunden

kundennr	**bonuspunkte**
5	2811

geschaeftskunden

kundennr	**ust_id**
8	555-12-3-456789

Möchte man hier den Namen und die Umsatzsteuer-ID aller Geschäftskunden wissen, muss man in zwei Tabellen (Kunden und Geschäftskunden) schauen.

Hierarchierelation

- Nur eine einzige Relation mit ALLEN Spalten.
- Type_Tag gibt den Entitätstypen an

Kunden(<u>Kundennr</u>, Name, E-Mail, Bonuspunkte, USt-ID, Type_Tag)

kunden

kundennr	**name**	**email**	**bonuspunkte**	**ust_id**	**type_tag**
4	Ute	ute@example.com	-	-	Kunde
5	Peter	peter@example.com	2811	-	Privatkunde
8	Anna	anna@example.com	-	555-...	Geschäftskun

In dieser Variante ist sowohl Suchen als auch Einfügen besonders einfach. Bei komplexen Generalisierungshierarchien kann es jedoch sehr viele Spalten mit vielen NULL-Werten geben.

18

Relationale Algebra

Die Relationale Algebra besteht aus Operationen, die auf ein oder mehreren Relationen angewendet werden können. Das Ergebnis einer solchen Operation ist wieder eine Relation.

Beispiel: Vereinigung

kunden

kundennr	**name**
4	Ute

privatkunden

kundennr	**name**
5	Peter

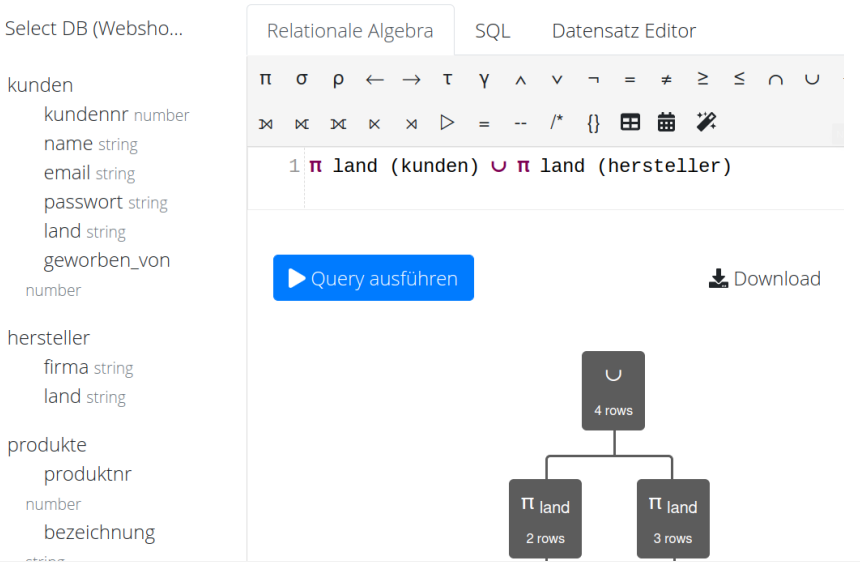
kunden ⋃ privatk.

kundennr	**name**
4	Ute
5	Peter

Der Vereinigungsoperation ⋃ wird auf zwei Relationen angewandt und liefert wieder eine Relation zurück.

RelaX - relational algebra calculator

https://dbis-uibk.github.io/relax/



Mit dem webbasierten Tool RelaX kann man Ausdrücke der relationalen Algebra formulieren und ausführen. Oben links im Tool wähnt man aus verschiedenen Beispielschemas, man kann auch ein eigenes Schema erstellen. Das Webshop-Schema aus dieser Vorlesung steht dort ebenfalls zur Auswahl und kann zum Üben von Anfragen auf diesem Schema verwendet werden.

33

19

Mengenoperationen

- Relationen sind Mengen von Tupeln
- Mengen können vereinigt, geschnitten und voneinander subtrahiert werden
- Das geht aber nur, wenn die Mengen vereinigungsverträglich sind

Vereinigungsverträglichkeit

- Gleiche Anzahl von Spalten
- Kompatible Datentypen

19

⋂ Schnittmenge

Diejenigen Zeilen, die in beiden Relationen vorkommen.

kunden

kundennr	name
4	Ute
5	Peter

privatkunden

kundennr	name
5	Peter
8	Anna

kunden ⋂ privatk.

kundennr	name
5	Peter

Die Zeile muss exakt gleich aussehen, sodass sie im Ergebnis zu sehen ist. Die Operation ⋂ würde nicht funktionieren, wenn die beiden Tabellen eine unterschiedliche Anzahl an Spalten haben. Auch, wenn Datentypen nicht kompatibel wären (z. B. erste Spalte keine Zahl sondern ein Datum), sind die Mengen nicht vereinigungsverträglich und daher kann auch keine Schnittmenge gebildet werden. Auf Englisch heißt die Schnittmenge Intersection.

19

\ Mengensubtraktion

Die Zeilen der ersten ohne die der zweiten Relation.

kunden

kundennr	name
4	Ute
5	Peter

privatkunden

kundennr	name
5	Peter
8	Anna

kunden \ privatk.

kundennr	name
4	Ute

Der Minus-Operator \ ist anders als die Vereinigung und Schnittmenge nicht symmetrisch. Die Zeilen der zweiten Relation werden von denen der ersten abgezogen. Im gezeigten Beispiel werden von {Ute, Peter} die Personen {Peter, Anna} abgezogen. Wenn man etwas abzieht, was nicht in der Menge ist (Anna), passiert nichts. Aber Peter wird abgezogen, sodass im Ergebnis lediglich Ute ist.

⋃ Vereinigung

Alle Zeilen aus beiden Relationen.

kunden

kundennr	name
4	Ute
5	Peter

privatkunden

kundennr	name
5	Peter
8	Anna

kunden ⋃ privatk.

kundennr	name
4	Ute
5	Peter
8	Anna

Relationen sind Mengen von Tupeln und in Mengen gibt es keine Duplikate. Daher erscheint hier im Ergebnis der Kunde Peter nur einmal, obwohl er in beiden Relationen jeweils einmal vorkommt. Auf Englisch heißt Vereinigung Union.

π Projektion

$$\pi_{A_1, A_2, \ldots, A_n}(R)$$

Beschränkung der Relation R auf die Spalten $A_1, A_2, \ldots, A_n$

kunden

kundennr	name	email
4	Ute	ute@example.com
5	Peter	peter@example.com
8	Anna	anna@example.com

$$\pi_{kundennr, name}(Kunden)$$

kundennr	name
4	Ute
5	Peter
8	Anna

Die Projektion ist ein unärer Operator, das heißt er nimmt nur eine Relation als Eingabe (Vereinigung etc. sind binär). Das Ergebnis ist gleich der Eingaberelation, jedoch nur mit den spezifizierten Spalten. Im Beispiel interessieren wir uns nur für die Kundennummern und Namen von Kunden, nicht für weitere Attribute.

π Projektion

Achtung: Duplikateliminierung!

produkte

produktnr	bezeichnung	hersteller
17	Schokoriegel	Monsterfood
18	Müsliriegel	Monsterfood
29	Spülmaschinentabs	Calgonte
88	Katzenfutter	-
999	Katalog	-

$\pi_{hersteller}(Produkte)$

hersteller
Monsterfood
Calgonte
-

Relationen sind Mengen von Tupeln und Mengen beinhalten keine Duplikate. Daher kann es vorkommen, dass das Ergebnis einer Projektion weniger Zeilen als die Eingaberelation hat.

22

σ Selektion

$\sigma_P(R)$

Auswahl derjenigen Zeilen der Relation R, die das Kriterium P erfüllen.

produkte

duktnr	bezeichnung	hersteller
	Schokoriegel	Monsterfood
	Müsliriegel	Monsterfood
	Spülmaschinentabs	Calgonte
	Katzenfutter	-
	Katalog	-

$\sigma_{hersteller='Monsterfood'}(Produkte)$

produktnr	bezeichnung	herstelle
17	Schokoriegel	Monsterf
18	Müsliriegel	Monsterf

Selektion

$\sigma_{preis>1}(\sigma_{hersteller='Monsterfood'}(Produkte))$

Klammern weglassen:
$\sigma_{preis>1}\sigma_{hersteller='Monsterfood'}(Produkte)$

Selektionen mit AND verbinden:
$\sigma_{preis>1 \wedge hersteller='Monsterfood'}(Produkte)$

OR geht so:
$\sigma_{preis>1 \vee hersteller='Monsterfood'}(Produkte)$

Das entspricht:
$\sigma_{preis>1}(Produkte) \cup \sigma_{hersteller='Monsterfood'}(Produkte)$

Welche Produkte vom Hersteller Monsterfood kosten mehr als 1 EUR? Die unteren beiden Ausdrücke liefern Produkte, die von Monsterfood sind oder mehr als einen Euro kosten (oder beides ist der Fall).

Operatorabfolgen

"Von welchen Herstellern aus Österreich gibt es keine Produkte?"

1. Welche Hersteller sind aus Österreich?
   $\sigma_{land='Österreich'}(Hersteller)$
2. Wie heißen diese Hersteller?
   $\pi_{firma}\sigma_{land='Österreich'}(Hersteller)$
3. Vor welchen Herstellern sind unsere Produkte?
   $\pi_{hersteller}(Produkte)$
4. Subtraktion von 2. und 3.:
   $\pi_{firma}\sigma_{land='Österreich'}(Hersteller) \setminus \pi_{hersteller}(Produkte)$

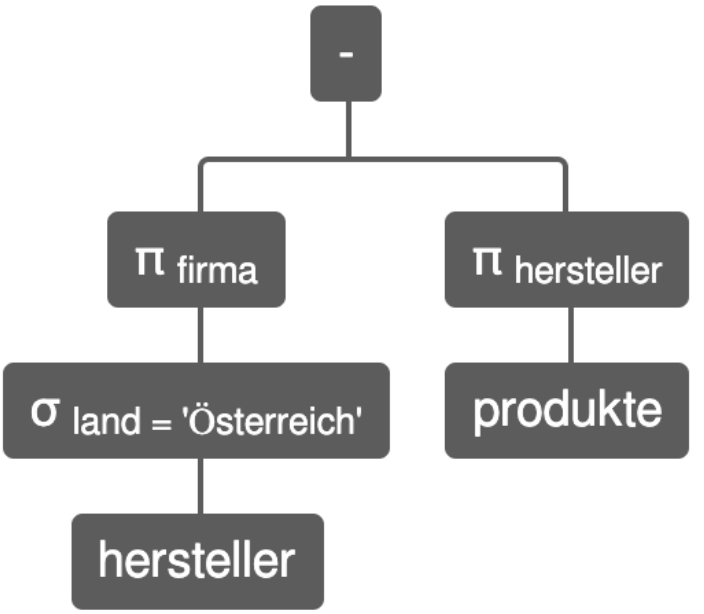
firma
Holzkopf

firma	land
Holzkopf	Österreich

firma
Holzkopf

hersteller
Monsterfood
Calgonte
-

Operatorbäume



Ein Operatorbaum stellt ein Ausdruck der relationalen Algebra in Baum-Form dar. Die Wurzel (ganz oben) liefert das Ergebnis, in den Blättern (ganz unten) befinden sich die verwendeten Relationen. Dazwischen bilden unäre und binäre Operationen die Knoten des Baumes.

× Kartesisches Produkt

25

"Jedes mit jedem"

$$Produkte \times Bewertungen$$

produktnr	bezeichnung	preis	hersteller	kundennr	produktnr	sterne	text
17	Schokoriegel	0.89	Monsterfood	5	17	4	...
17	Schokoriegel	0.89	Monsterfood	5	29	1	...
17	Schokoriegel	0.89	Monsterfood	8	29	2	...
18	Müsliriegel	1.19	Monsterfood	5	17	4	...
18	Müsliriegel	1.19	Monsterfood	5	29	1	...
18	Müsliriegel	1.19	Monsterfood	8	29	2	...
29	Spülmaschinentabs	3.99	Calgonte	5	17	4	...
29	Spülmaschinentabs	3.99	Calgonte	5	29	1	...

Das kartesische Produkt aus zwei Relationen hat alle Attribute beider Relationen und besteht aus jedem Tupel der einen verknüpft mit jedem Tupel der anderen Relation.

Tabellenprefix

Der Name der Relation kann bei Attributen als Prefix angegeben werden.

$$\sigma_{Produkte.Produktnr=Bewertungen.Produktnr}(Produkte \times Bewertungen)$$

produktnr	bezeichnung	preis	hersteller	kundennr	produktnr	sterne	text
17	Schokoriegel	0.89	Monsterfood	5	17	4	...
29	Spülmaschinentabs	3.99	Calgonte	5	29	1	...
29	Spülmaschinentabs	3.99	Calgonte	8	29	2	...

In dieser Anfrage werden alle Produkte mit allen Bewertungen verbunden und im Anschluss eine Selektion darüber gemacht, sodass die Produktnummer des Produktes und der Bewertung übereinstimmt. Alles andere wären unsinnige Zeilen (Bewertung eines anderen Produkts). Da das Attribut Produktnr in beiden Relationen vorkommen, verwenden wir den Relationennamen als Prefix, z. B. produkte.produktnr, um die Attribute voneinander zu unterscheiden

ρ Umbenennungsoperator

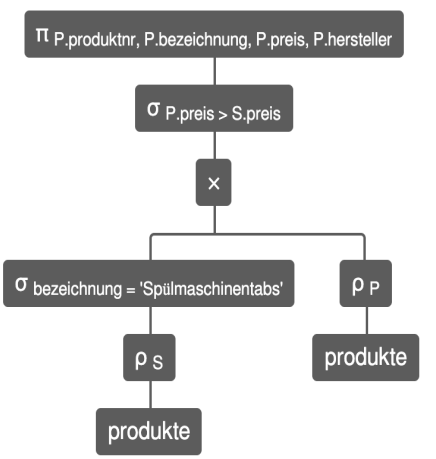
Relation umbenennen

$$\rho_{P1}(Produkte)$$

Attribut umbenennen

$$\rho_{bez \leftarrow bezeichnung}(Produkte)$$

Welche Produkte kosten mehr als die Spülmaschinentabs?



Spätestens wenn man ein und dieselbe Relation mehrfach innerhalb einer Anfrage braucht, ist es hilfreich Relationen oder Attribute einen Alias zu geben.

⋈ Join (Verbund)

Ein Join ist ein Kreuzprodukt mit anschließender Selektion, welche die Spaltenwerte der beiden Relationen vergleicht

$$R \bowtie_P S = \sigma_P(R \times S)$$

Welche Produkte sind von einem Hersteller aus den USA?

$$\pi_{Bezeichnung}\sigma_{Land='USA'}(Produkte \bowtie_{Produkte.Hersteller=Hersteller.Firma} Hersteller)$$

Es gilt: $R \bowtie S = S \bowtie R$

$R \bowtie_P S$ bedeutet, dass die beiden Relationen R und S anhand des Join-Prädikats P verbunden werden.

Suche nach Join-Partnern

$$Produkte \bowtie_{Produkte.Hersteller=Hersteller.Firma} Hersteller$$

produkte

produktnr	bezeichnung	preis	hersteller
17	Schokoriegel	0.89	Monsterfood
18	Müsliriegel	1.19	Monsterfood
29	Spülmaschinentabs	3.99	Calgonte
88	Katzenfutter	4.99	-

hersteller

firma	land
Calgonte	Italien
Monsterfood	USA
Holzkopf	Österreich

Man kann sich die Ausführung eines Joins auch so vorstellen, dass eine Relation von oben nach unten durchscannt wird - z. B. hier die Produkttabelle - und für jede Zeile ein (oder kein oder mehrere) Join-Partner in der anderen Relation - hier: Hersteller - gesucht wird. Die Attributwerte der gefundenen Zeile wird an die Ergebniszeile drangehangen. Wird kein Join-Partner gefunden - hier beim Katzenfutter der Fall -, taucht die Zeile nicht im Ergebnis auf. Würde eine Zeile mehrere Joinpartner finden, taucht sie mehrfach im Ergebnis auf. Jeweils einmal mit dem entsprechenden Join-Partner.

Ergebnis des Joins

$$Produkte \bowtie_{Produkte.Hersteller=Hersteller.Firma} Hersteller$$

produktnr	bezeichnung	preis	hersteller	firma	land
17	Schokoriegel	0.89	Monsterfood	Monsterfood	USA
18	Müsliriegel	1.19	Monsterfood	Monsterfood	USA
29	Spülmaschinentabs	3.99	Calgonte	Calgonte	Italien

Der Join ist verlustbehaftet.

Die Ergebnisrelation des Joins zwischen zwei Relationen besitzt alle Spalten beider Relationen. Zu jeder Zeile der beiden Relationen existieren entsprechend viele Zeilen im Ergebnis, je nachdem wie viele Join-Partner zu ihr gefunden werden. Der Hersteller Monsterfood hat sogar zwei Join-Partner gefunden, daher taucht die Hersteller-Zeile (Monsterfood, USA) im Ergebnis zweimal auf. Das Produkt Katzenfutter hat hier keinen Join-Partner gefunden, der Hersteller Holzkopf ebenfalls nicht. Da also beim Join etwas verloren gegangen ist (Katzenfutter und Hersteller Holzkopf), nennt man den Join verlustbehaftet.

Verlustfreier Join

produkte

produktnr	bezeichnung	preis	hersteller
17	Schokoriegel	0.89	Monsterfood
18	Müsliriegel	1.19	Monsterfood
29	Spülmaschinentabs	3.99	Calgonte

hersteller

firma	land
Calgonte	Italien
Monsterfood	USA

$$Produkte \bowtie_{Produkte.Hersteller=Hersteller.Firma} Hersteller$$

produktnr	bezeichnung	preis	hersteller	firma	land
17	Schokoriegel	0.89	Monsterfood	Monsterfood	USA
18	Müsliriegel	1.19	Monsterfood	Monsterfood	USA
29	Spülmaschinentabs	3.99	Calgonte	Calgonte	Italien

Hier taucht jedes Tupel aus beiden Relationen im Join-Ergebnis auf. Der Join ist verlustfrei.

Rekonstruktion der Tabellen

V

produktnr	bezeichnung	preis	hersteller	firma	land
17	Schokoriegel	0.89	Monsterfood	Monsterfood	USA
18	Müsliriegel	1.19	Monsterfood	Monsterfood	USA
29	Spülmaschinentabs	3.99	Calgonte	Calgonte	Italien

$$produkte = \pi_{produktnr,bezeichnung,preis,hersteller}(V)$$

$$hersteller = \pi_{firma,land}(V)$$

Aus dem Join-Ergebnis V lassen sich die beiden ursprünglichen Relationen wieder mittels Projektionen rekonstruieren. Das funktioniert allerdings nur, wenn der Verbund verlustfrei ist.

Äußerer Verbund

⋈ Innerer Verbund

Nur die Zeilen, die Join-Partner finden, sind im Ergebnis

⟕ Linker äußerer Verbund

Alle Zeilen der linken Relation sind definitiv im Ergebnis

⟖ Rechter äußerer Verbund

Alle Zeilen der rechten Relation sind definitiv im Ergebnis

⟗ Voller äußerer Verbund

Alle Zeilen beider Relation sind definitiv im Ergebnis

⟕ Linker äußerer Verbund

27

$$Produkte \text{⟕}_{Produkte.Hersteller=Hersteller.Firma} Hersteller$$

produktnr	bezeichnung	preis	hersteller	firma	land
17	Schokoriegel	0.89	Monsterfood	Monsterfood	USA
18	Müsliriegel	1.19	Monsterfood	Monsterfood	USA
29	Spülmaschinentabs	3.99	Calgonte	Calgonte	Italien
88	Katzenfutter	4.99	-	-	-

Alle Zeilen der links vom Left-Join-Operator stehenden Relation erscheinen auf jeden Fall im Ergebnis. Wenn sie keinen Join-Partner in der rechten Tabelle finden - das ist hier beim Katzenfutter der Fall -, bleiben die Attribute der rechten Tabelle alle NULL.

53

⟖ Rechter äußerer Verbund

28

$$Produkte \text{⟖}_{Produkte.Hersteller=Hersteller.Firma} Hersteller$$

produktnr	bezeichnung	preis	hersteller	firma	land
29	Spülmaschinentabs	3.99	Calgonte	Calgonte	Italien
17	Schokoriegel	0.89	Monsterfood	Monsterfood	USA
18	Müsliriegel	1.19	Monsterfood	Monsterfood	USA
-	-	-	-	Holzkopf	Österreic

Es gilt: $R \text{⟖} S = S \text{⟕} R$

Beim right outer Join sind zusätzlich zu den normalen Join-Ergebniszeilen diejenigen Zeilen der rechten Tabelle im Ergebnis wiederzufinden, die keinen Join-Partner in der linken Tabelle finden. Auch hier werden die Spalten der linken Tabelle mit NULL-Werten belegt.

54

⟗ Voller äußerer Verbund

$$Produkte \text{⟗}_{Produkte.Hersteller=Hersteller.Firma} Hersteller$$

produktnr	bezeichnung	preis	hersteller	firma	land
17	Schokoriegel	0.89	Monsterfood	Monsterfood	USA
18	Müsliriegel	1.19	Monsterfood	Monsterfood	USA
29	Spülmaschinentabs	3.99	Calgonte	Calgonte	Italien
88	Katzenfutter	4.99	-	-	-
-	-	-	-	Holzkopf	Österreic

Es gilt: $R \text{⟗} S = S \text{⟗} R$

Jede Zeile der linken Relation und jede Zeile der rechten Relation tauchen stets im Ergebnis des full outer Joins auf. Der volle äußere Verbund ist damit immer verlustfrei.

55

Join-Varianten

Innerer / linker / rechter / voller äußerer Join

$$R \bowtie_P S \quad R \text{⟕}_P S \quad R \text{⟖}_P S \quad R \text{⟗}_P S$$

Gleichverbund (Equi-Join)

$$R \bowtie_{R.a=S.x \wedge R.b=S.y \wedge \ldots} S$$

Theta-Join

$$R \bowtie_{R.a \theta S.x \wedge \ldots} S \text{ mit } \theta \in \{<, \leq, =, \neq, \geq, >\}$$

Beispiele: Equi-/Theta-Joins

Finde zu jedem Produkt seinen Hersteller:

$$produkte \bowtie_{Produkte.Hersteller=Hersteller.Firma} hersteller$$

Finde zu jedem Produkt teurere Produkte als es selbst:

$$\rho_{P1}(produkte) \bowtie_{P1.preis<P2.preis} \rho_{P2}(produkte)$$

produktnr	bezeichnung	preis	hersteller	produktnr	...
18	Müsliriegel	1.19	Monsterfood	29	
18	Müsliriegel	1.19	Monsterfood	88	
18	Müsliriegel	1.19	Monsterfood	91	
18	Müsliriegel	1.19	Monsterfood	92	

Join-Varianten

Natürlicher Verbund (natural Join)

$$R \bowtie S$$

Gleichverbund über die gleich heißenden Attribute. Im Ergebnis sind solche Attribute nur einmal vorhanden.

Self-Join

$$R \bowtie_P R$$

Semi-Join

$$R \ltimes_P S = \pi_{R.*}(R \bowtie_P S)$$

Beispiel: Natürlicher Verbund

$$produkte \bowtie bewertungen$$

produktnr	bezeichnung	preis	hersteller	kundennr	sterne	bewertungstext
17	Schokoriegel	0.89	Monsterfood	5	4	Guter Schokoriegel, aber die Verpackung geht schwer auf
29	Spülmaschinentabs	3.99	Calgonte	5	1	Mein Geschirr wird nicht sauber!
29	Spülmaschinentabs	3.99	Calgonte	8	2	Nicht gut, aber billig.

Entspricht:

$$\pi_{produkte.produktnummer,bezeichnung,preis,hersteller,kundennr,sterne,bewertungstext}(produkte \bowtie_{produkte.produktnr=bewertungen.produktnr} bewertungen)$$

In den Relationen Produkte und Bewertungen gibt es ein gemeinsames Attribut, die Produktnummer. Daher ist der natürliche Verbund ein Gleichverbund über die Produktnummer. Diese Spalte taucht im Ergebnis dann aber nur einmal auf.

Beispiel: Self-Join

personen

persnr	name	chef
4	Ute	-
5	Peter	4
8	Anna	5

$$\rho_{K1}(personen) \bowtie_{K1.chef=K2.persnr} \rho_{K2}(personen)$$

persnr	name	chef	persnr	name	chef
5	Peter	4	4	Ute	-
8	Anna	5	5	Peter	4

Wir joinen die Personentabelle mit sich selbst anhand der Fremdschlüsselbeziehung zwischen Chef und Kundennr. Im Ergebnis sehen wir zu jeder Person, die einen Chef hat, die Details zur Person und zum jeweiligen Chef. Würden wir den Join $\bowtie$ durch einen Left-Join $⟕$ ersetzen, so erschienen auch Personen ohne Chef im Ergebnis. Dann mit NULL-Werten in den hinteren drei Spalten.

Beispiel: Semi-Join

$$produkte \ltimes_{produkte.produktnr=bewertungen.produktnr} bewertungen$$

produktnr	bezeichnung	preis	hersteller
17	Schokoriegel	0.89	Monsterfood
29	Spülmaschinentabs	3.99	Calgonte
29	Spülmaschinentabs	3.99	Calgonte

Alle Produkte, die schon einmal bewertet wurden.

Der Semi-Join unterscheidet sich vom inneren Verbund dadurch, dass im Ergebnis nur die Spalten der linken Relation zu sehen ist. Der Semi-Join hat also die gleichen Attribute wie die linke Relation, aber nur diejenigen Zeilen, die einen Join-Partner in der rechten Relation finden würden. Im Beispiel hier interessiert man sich also nicht dafür, wer ein Produkt wie bewertet hat, sondern lediglich *dass* es bewertet wurde. Verwendet wurde hier der linke Semi-Join ⋉. Beim rechten Semi-Join ⋊ wird sich auf die Attribute der rechten Relation beschränkt.

÷ Division

$$R \div S$$

Diejenigen Tupel aus R (ohne die Spalten von S), die in jeder Kombination mit allen Tupeln aus S vorkommen.

Es gilt: $(R \times S) \div S = R$

Die Division ist hilfreich bei Fragen der Art "Wer hat alle..." oder "Wer hat jedes...". Der Operator kann auch mit den bisher vorgestellten Operationen dargestellt werden:
$R \div S = \pi_{R.*\setminus S.*}(R) \setminus \pi_{R.*\setminus S.*}((\pi_{R.*\setminus S.*}(R) \times S) \setminus R)$ (Formel nicht klausurrelevant)

Beispiel: Division

Welche Kunden haben alle Produkte bewertet?

$\pi_{kundennr,produktnr}(bewertungen)$

kundennr	produktnr
5	17
5	29
8	29
4	29

$\pi_{produktnr}(produkte)$

produktnr
17
29

$\pi_{kundennr,produktnr}(bewertungen) \div \pi_{kundennr}(kunden)$

kundennr

Hier gehen wir der Einfachheit halber davon aus, dass es nur die beiden Produkte 17 und 29 gibt. Die Division liefert diejenigen Kunden, die alle diese Produkte bewertet hat.

Anfrageoptimierung

"Wie heißen die Produkte, die Kunde Nr. 5 bewertet hat?"

Äquivalente Ausdrücke:

a. $\pi_{bezeichnung}\sigma_{kundennr=5}\sigma_{produkte.produktnr=bewertungen.produktnr}(produkte \times bewertu$
b. $\pi_{bezeichnung}(\pi_{produktnr,bezeichnung}produkte \bowtie \pi_{produktnr}\sigma_{kundennr=5}(bewertungen)$
c. $\pi_{bezeichnung}\sigma_{kundennr=5}(produkte \bowtie bewertungen)$
d. $\pi_{bezeichnung}(produkte \bowtie \sigma_{kundennr=5}(bewertungen))$

Welcher Ausführungsplan ist besser / "billiger"?

Anfrageoptimierung

Überführung eines Ausdrucks in einen äquivalenten möglichst effizient auszuführenden Ausdruck.

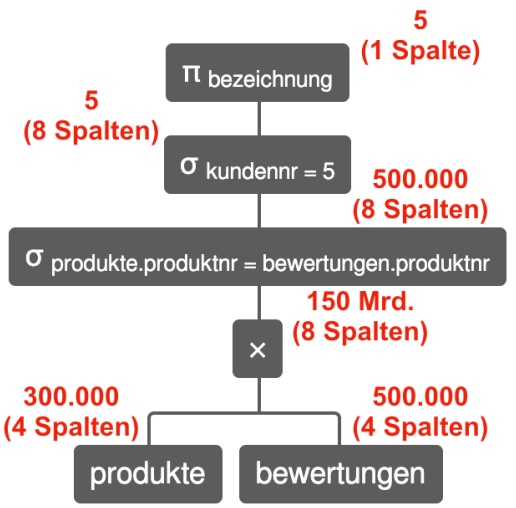
Kostenbasierte Optimierer

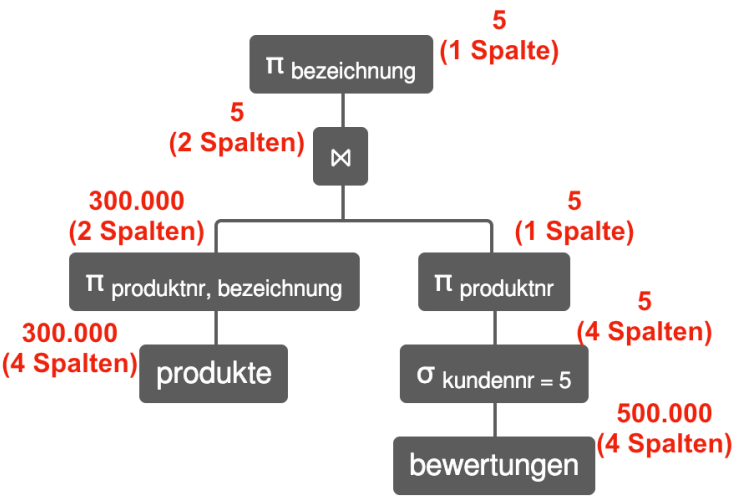
Jeder Ausführungsplan erhält Kostenschätzung.
Der Plan mit den geringsten Kosten wird gewählt.

Beispiel: Kosten = Größe der Zwischenergebnisse

Beispiel: Anfrageoptimierung

Annahme: 100.000 Kunden, 500.000 Bewertungen, 300.000 Produkte





Der rechte Plan ist billiger, da die Zwischenergebnisse deutlich kleiner sind: 300.000+300.000+500.000+5+5+5+5=1.100.020; links: 150.001.300.010; Auch haben die Zwischenergebnisse weniger Spalten.

Heuristiken

- Frühstmögliche Selektion
- Join statt Kreuzprodukt
- Frühstmögliche Projektion (ohne Duplikateliminierung)
- Join-Reihenfolge so wählen, dass Zwischenergebnisse klein sind
- Folgen von Selektionen und Projektionen zusammenfassen
- Selektionen statt Mengenoperationen
- Nichts doppelt berechnen

Heuristiken

Frühstmögliche Selektion

Vorher: $\sigma_{kundennr=5}(produkte \bowtie bewertungen)$
Nachher: $produkte \bowtie \sigma_{kundennr=5}(bewertungen)$

Join statt Kreuzprodukt

Vorher: $\sigma_{produkte.produktnr=bewertungen.produktnr}(produkte \times bewertungen)$
Nachher: $produkte \bowtie_{produkte.produktnr=bewertungen.produktnr} (bewertungen)$

Frühstmögliche Projektion (ohne Duplikateliminierung)

Vorher: $\pi_{bezeichnung}(produkte \bowtie bewertungen)$
Nachher:

Bei der frühestmöglichen Projektion ist darauf zu achten, dass keine Spalten frühzeitig eliminiert werden, die noch für Joins, Selektionen, etc. benötigt werden. Außerdem darf bei der Projektion noch keine Duplikateliminierung erfolgen, da sonst das Ergebnis evtl. nicht mehr äquivalent ist.

Heuristiken

Join-Reihenfolge so wählen, dass Zwischenergebnisse klein sind

Vorher: $(kunden \bowtie bewertungen) \bowtie \sigma_{hersteller='Monsterfood'}(produkte)$
Nachher: $(bewertungen \bowtie \sigma_{hersteller='Monsterfood'}(produkte)) \bowtie kunden$

Folgen von Sel. und Proj. zusammenfassen

Vorher: $\pi_{bezeichnung}\pi_{produktnr,bezeichnung}\sigma_{preis\leq 5}\sigma_{hersteller='Monsterfood'}\ produkte$
Nachher: $\pi_{bezeichnung}\sigma_{preis\leq 5 \wedge hersteller='Monsterfood'}\ produkte$

Selektionen statt Mengenoperationen

Vorher: $\sigma_{hersteller='Monsterfood'}(produkte) \cup \sigma_{hersteller='Calgonte'}(produkte)$
Nachher: $\sigma_{hersteller='Monsterfood' \vee hersteller='Calgonte'}(produkte)$

Kardinalitätsschätzung

Wie viele Tupel sind im Ergebnis einer Operation zu erwarten?

Hilfreiche Statistiken:

- Kardinalitäten der Tabellen (Anzahl Zeilen)
- Kardinalitäten der Spalten (Anzahl distinkter Werte)
- Kleinster, größter Wert je Spalte, Median, ...
- Werte-Histogramme (Häufigkeitsverteilung)
- Erfahrungen über Verschätzungen in der Vergangenheit (→ lernende Optimierer)
- ...

Kardinalitätsschätzung: Selektion

Annahme: Gleichverteilung

|R| (Anzahl Zeilen in R)
|R.a| (Anzahl distinkter Werte in Spalte R.a)

$$|\sigma_{R.a=x}(R)| = \frac{|R|}{|R.a|}$$

Beispiel:
$|\sigma_{geschlecht='weiblich'}(Personen)| = \frac{1}{3}|Personen|$

Wird nach einem bestimmten Wert in einer Spalte gesucht und liegen keine weiteren Informationen über Werteverteilungen innerhalb dieser Spalte vor, wird von *Gleichverteilung* ausgegangen. Im Beispiel auf dieser Folie wird geschätzt, dass ein Drittel aller gespeicherten Personen weiblich sind, weil es drei verschiedene Werte in der Spalte Geschlecht gibt (männlich, weiblich, divers).

Kardinalitätsschätzung: Selektion

Selektivitätsfaktor sf_P: $|\sigma_P R| = sf_P \cdot |R|$

Annahme: Werteunabhängigkeit

$$sf_{P \wedge Q} = sf_P \cdot sf_Q$$

Beispiel:
3 verschiedene Geschlechter, 1000 verschiedene Vornamen

$|\sigma_{geschlecht='weiblich' \wedge vorname='Peter'}(Personen)| = \frac{1}{3} \cdot \frac{1}{1000}|Personen|$

$|\sigma_{geschlecht='männlich' \wedge vorname='Peter'}(Personen)| = \frac{1}{3} \cdot \frac{1}{1000}|Personen|$

Da in der Regel die Information nicht vorliegt, dass zwischen bestimmten Spalten eine Werteabhängigkeit besteht, wird von Unabhängigkeit ausgegangen. Wahrscheinlich wird in Wirklichkeit keine Frau in der Personentabelle Peter heißen. In der unteren Anfrage filtert das Geschlechts-Kriterium wahrscheinlich nichts aus.

Kardinalitätsschätzung: ×

$$|R \times S| = |R| \cdot |S|$$

Beispiel: $|kunden \times kunden| = |kunden|^2$

Kardinalitätsschätzung: ⋈

$$|R \bowtie_{R.a=S.a} S|$$

Im Allgemeinen:

$$0 \leq |R \bowtie_{R.a=S.a} S| \leq |R| \cdot |S|$$

Wenn R.a Fremdschlüssel auf S.a ist:

$$|R \bowtie_{R.a=S.a} S = |R|$$

Beispiel: $|produkte \bowtie bewertungen| = |bewertungen|$

Haben die Tabellen R und S völlig verschiedene Werte in der Spalte a, ist das Ergebnis des Joins leer. Wenn die Join-Spalten eine Fremdschlüssel-Primärschlüssel-Beziehung darstellen, hat das Ergebnis des Joins so viele Zeilen, wie die Tabelle mit der Fremdschlüsselspalte (abzgl. der Anzahl von NULL-Werten in dieser), da jede solche Zeile genau einen Join-Partner in der anderen Tabelle findet, unabhängig davon, wie groß diese andere Tabelle ist. Gibt es beispielsweise nur ein Produkt, sind alle Bewertungen über dieses eine Produkt. Gibt es deutlich mehr Produkte als Bewertungen, sind einige Produkte nicht bewertet worden. Im Join-Resultat sind in jedem Fall genau so viele Zeilen wie in der Bewertungs-Tabelle. Die Produktnr-Spalte ist dort nie NULL.

Normalformenlehre

- Was ist ein gutes DB-Schema?
- Funktionale Abhängigkeiten
- Superschlüssel, Schlüsselkandidaten
- Normalformen: 1NF, 2NF, 3NF

Ist dies ein gutes DB-Schema?

cd_id	**tracknr**	**album**	**band**	**land**	**song**
101	1	Jupiter	Eddy G.	DE	All ducks
101	2	Jupiter	Eddy G.	DE	Far away
202	1	Mars	Eddy G.	DE	Meersalz
202	2	Mars	Eddy G.	DE	Mehr Salz
303	1	Stone	Bob 88	EN	I'm 88

Dieses Datenbankschema ist eher schlechter Natur. Es herrschen viele Redundanzen. Dinge, die in unterschiedliche Relationen gehören, wurden in eine Relation zusammengeworfen.

Anomalien bei Redundanzen

cd_id	tracknr	album	band	land	song

Einfügeanomalie

Wir können keine neue Band hinzufügen, wenn sie noch kein Album herausgebracht hat.

Änderungsanomalie

Wenn wir das Land einer Band ändern, muss diese Änderung an vielen Stellen erfolgen.

Löschanomalie

Wenn wir das letzte Album einer Band löschen, verlieren wir die Band-Infos.

Funktionale Abhängigkeit

A und B sind Attributmengen aus der Relation R

$A \rightarrow B$ (lies: A bestimmt B)

Immer wenn zwei Zeilen in R die gleichen Werte in den A-Attributen haben, dann sind auch die Werte in den B-Attributen gleich.

Beispiel: $band \rightarrow land$

Immer wenn zwei Zeilen in unserer CD-Track-Tabelle in der Spalte "band" das gleiche stehen haben, so muss auch der Wert in "land" gleich sein. Bei der Band Eddy G. ist das Land immer DE.
Funktionale Abhängigkeiten ergeben sich aus der Semantik der Anwendung. Sie herrschen in einem DB-Schema. Man kann sie nicht von den aktuell gespeicherten Daten ableiten. Funktionale Abhängigkeiten sind Bedingungen, die in jedem DB-Zustand stets gelten.

Funktionale Abhängigkeiten

cd_id	**tracknr**	**album**	**band**	**land**	**song**
101	1	Jupiter	Eddy G.	DE	All ducks
101	2	Jupiter	Eddy G.	DE	Far away
202	1	Mars	Eddy G.	DE	Meersalz
202	2	Mars	Eddy G.	DE	Mehr Salz
303	1	Stone	Bob 88	EN	I'm 88

Es gilt: $cd\_id \rightarrow album$; $cd\_id \rightarrow band$; $band \rightarrow land$; $cd\_id, tracknr \rightarrow song$

Volle funktionale Abhängigkeit

$A \Rightarrow B$ (lies: A bestimmt B voll-funktional)

Dies ist der Fall wenn

- $A \rightarrow B$ und
- $\nexists X \subset A : X \rightarrow B$

Beispiel: $cd\_id, tracknr \rightarrow land$
Aber nicht: $cd\_id, tracknr \Rightarrow land$, weil bereits $cd\_id \rightarrow land$

Superschlüssel

Die Attributmenge A ist Superschlüssel der Relation R, genau dann wenn $A \rightarrow R$ (sie bestimmt alle Attribute)

Ist cd_id Superschlüssel?

$cd\_id \rightarrow album; cd\_id \rightarrow band; band \rightarrow land;$

$\Rightarrow cd\_id \rightarrow cd\_id, album, band, land$

$\Rightarrow cd\_id$ ist kein Superschlüssel (z. B. $cd\_id \nrightarrow tracknr$)

Ist cd_id, tracknr Superschlüssel?

$cd\_id \rightarrow album; cd\_id \rightarrow band; band \rightarrow land; cd\_id, tracknr \rightarrow song$

$\Rightarrow cd\_id, tracknr \rightarrow cd\_id, tracknr, album, band, land, song = R$

$\Rightarrow cd\_id, tracknr$ ist Superschlüssel

Schlüsselkandidat

Die Attributmenge A ist Schlüsselkandidat von R, wenn

1. A Superschlüssel ist und
2. keine echte Teilmenge von A Superschlüssel ist.

Ein Schlüsselkandidat ist ein minimaler Superschlüssel.

Ist cd_id, tracknr Schlüsselkandidat?

- cd_id, tracknr ist Superschlüssel (siehe vorherige Folie)
- cd_id ist kein Superschlüssel (siehe vorherige Folie)
- tracknr ist kein Superschlüssel (nur $tracknr \rightarrow tracknr$)

$\Rightarrow cd\_id, tracknr$ ist Schlüsselkandidat

Ein Schlüsselkandidat eignet sich dafür, als Primärschlüssel der Relation eingesetzt zu werden. Wenn es mehrere Schlüsselkandidaten gibt, wählt man einen davon aus, der Primärschlüssel wird.

1. Normalform (1NF)

Eine Relation ist in erster Normalform, wenn alle ihre Attribute weder zusammengesetzt noch mengenwertig noch relationenwertig sind.

cd_id	**tracknr**	**album**	**band**	**land**	**song**
101	1	Jupiter	Eddy G.	DE	All ducks
101	2	Jupiter	Eddy G.	DE	Far away
202	1	Mars	Eddy G.	DE	Meersalz
202	2	Mars	Eddy G.	DE	Mehr Salz
303	1	Stone	Bob 88	EN	I'm 88

=> Ist hier der Fall! Die Relation ist in 1NF.

$(NF)^2$ = Non First Normal Form

pizza_id	**bezeichnung**	**zutaten**	**preis**	
1	Spezial	Salami, Schinken, Pilze	**größe**	**preis**
			klein	7.00
			groß	8.50
2	Hawaii	Ananas, Schinken	**größe**	**preis**
			klein	6.50
			groß	7.50

Zutaten ist ein mengenwertiges Attribut, Preis ist ein relationenwertiges Attribut.

=> Nicht in 1NF.

Überführung in 1NF

Attribute flachklopfen.

pizza_id	**bezeichnung**	**zutat**
1	Spezial	Salami
1	Spezial	Schinken
1	Spezial	Pilze
...	...	...

pizza_id	**bezeichnung**	**größe**	**preis**
1	Spezial	klein	7.00
1	Spezial	groß	8.50
...	...	...	...

Jetzt ist das Schema in 1NF.

2. Normalform (2NF)

Eine Relation ist in zweiter Normalform, wenn sie in 1NF ist und jedes Nicht-Schlüsselattribut voll vom ganzen Schlüssel abhängt.

Ist Pizza1(pizza_id, bezeichnung, zutat) in 2NF?

Überprüfe für Nicht-Schlüsselattribut "bezeichnung":
$pizza\_id, zutat \Rightarrow bezeichnung$?

Nicht der Fall, weil bereits $pizza\_id \rightarrow bezeichnung$

Damit ist die Relation nicht in 2NF.

Überführung in 2NF

Zerlegung in Relationen, die den Teil der Schlüsselattribute besitzen, von dem die jeweiligen Nicht-Schlüssel-Attribute voll funktional abhängen.

Gegeben: Pizza1(pizza_id, bezeichnung, zutat)
$pizza\_id \rightarrow bezeichnung$

Resultat in 2NF:
Pizza1a(pizza_id, zutat) und Pizza1b(pizza_id, bezeichnung)
Pizza1a.pizza_id ist Fremdschlüssel auf Pizza1b.pizza_id

Wichtig bei Zerlegungen: Zerlegung muss verlustfrei erfolgen und ein Join der neuen Tabellen muss wieder die ursprüngliche Tabelle ergeben.

CD-Track-Beispieltabelle

cd_id	**tracknr**	**album**	**band**	**land**	**song**

$cd\_id \rightarrow album; cd\_id \rightarrow band; band \rightarrow land; cd\_id, tracknr \rightarrow song$

Ist die Tabelle in 2NF?

Überprüfe für Nicht-Schlüsselattribut "album", ob $cd\_id, tracknr \Rightarrow album$?

Nein, weil $cd\_id \rightarrow album$ ⇒ Die Relation ist also nicht in 2NF

Überführung in 2NF:

- Tracks(cd_id,tracknr, song)
- Alben(cd_id, album, band, land)

Tracks.cd_id ist Fremdschlüssel auf Alben.cd_id

Bei der Überführung der CD-Track-Tabelle in 2NF wurde diese Tabelle in zwei Tabellen zerlegt. Wir geben ihnen sinnvolle Namen: Alben und Tracks. Lediglich die Spalte "song" ist vom ganzen Schlüssel voll funktional abhängig. Die anderen Nicht-Schlüsselattribute hängen nur von der CD_ID ab und werden daher in die neue Alben-Relation ausgelagert.

3. Normalform (3NF)

Eine Relation ist in dritter Normalform, wenn sie in 2NF ist und es keine nicht-trivialen transitiven Abhängigkeiten zwischen Nicht-Schlüsselattributen gibt.

Triviale Abhängigkeit: $A \rightarrow X$ mit $X \subseteq A$

Transitive Abhängigkeit: $A \rightarrow B; B \rightarrow C$

Ist Alben(cd_id, album, band, land) in 3NF?

Nein, weil es folgende transitive Abhängigkeit gibt:
$cd\_id \rightarrow band; band \rightarrow land$

Ein Nicht-Schlüssel-Attribut (das sind Attribute, die zu keinem Schlüsselkandidaten gehören) darf keine anderen Nicht-Schlüssel-Attribute funktional bestimmen. Ist dies der Fall, ist die Relation in 3NF.

89

Überführung in 3NF

Transitive Abhängigkeit: $A \rightarrow B; B \rightarrow C$

Entfernen von C aus der Relation

Neue Relation hat Attribute B (Primärschlüssel) und C

Gegeben: Alben(cd_id, album, band, land)
$cd\_id \rightarrow band; band \rightarrow land$

Resultat in 3NF:

- Alben(cd_id, album, band)
- Bands(band, land)

Alben.band ist Fremdschlüssel auf Bands.band

Kapitelzusammenfassung

- Relation/Tabelle = Menge von Tupeln
- Primärschlüssel, Fremdschlüssel, NULL-Werte
- Überführung ER → Relationenschema
- Relationale Algebra: π, σ, $\bowtie$, $\times$, $\cup$, $\div$, $\ldots$
- Join: Inner, Left/Right/Full Outer, ...
- Anfrageoptimierung / Heuristiken
- Funktionale Abhängigkeiten
- Superschlüssel, Schlüsselkandidaten
- Normalformen: 1NF, 2NF, 3NF