Datenbanken und SQL

Kapitel 2

Das Relationenmodell

Das Relationenmodell

- Beispiel: Relation *Verkäufer-Produkt*
- Relationale Datenstrukturen
  - Begriffe
  - Definition: Relation, Relationale Datenbank
- Primärschlüssel
- Relationale Integritätsregeln
  - Regel 1: Entity Integritätsregel
  - Regel 2: Referenz Integritätsregel
- Relationale Algebra
  - Relationale Operatoren
  - Eigenschaften der relationalen Operatoren

Das Relationenmodell

- **Ziele des Relationenmodells sind**
  - Geringe Redundanz
  - Gute Handhabbarkeit
  - Einfache und schnelle Zugriffe
  - Sicherstellung von Konsistenz und Integrität

Keine doppelten Einträge

Einfache Befehle

Zugriffe über wenige Tabellen

Aufstellen von Regeln

- Folgerung:
  - Entsprechende Forderungen an Relationen

Beispiel: Relation VerkaeuferProdukt

VerkNr	VerkName	PLZ	VerkAdresse	Produktname	Umsatz
V1	Meier	80075	München	Waschmaschine	11000
V1	Meier	80075	München	Herd	5000
V1	Meier	80075	München	Kühlschrank	1000
V2	Schneider	70038	Stuttgart	Herd	4000
V2	Schneider	70038	Stuttgart	Kühlschrank	3000
V3	Müller	50083	Köln	Staubsauger	1000

Vorteil:

✓ Übersichtlich in einer Tabelle

Nachteil:

✓ Redundanz

✓ Schlechte Handhabbarkeit

Probleme mit VerkaeuferProdukt

- **Redundanz**
  - Je mehr ein Verkäufer verkauft, um so häufiger in Tabelle!
  - Ändert sich die Adresse eines Verkäufers, muss dies in **allen** entsprechenden Einträgen erfolgen. Sonst: **Inkonsistenz**!
- **Handhabung**
  - Soll Produkt Staubsauger aus dem Sortiment genommen werden, so ist auch Verkäufer Müller zu löschen!?
  - Verkäufer Schmidt kann erst eingetragen werden, wenn er etwas verkauft hat!?

Begriffe in relationalen Datenbanken

Formale relationale Bezeichner	Informelle Bezeichnung
Relation	Tabelle
Tupel	Zeile einer Tabelle
Kardinalität	Anzahl der Zeilen einer Tabelle
Attribut	Spalte einer Tabelle
Grad	Anzahl der Spalten einer Tabelle
Primärschlüssel	eindeutiger Bezeichner
Gebiet	Menge aller möglichen Werte

Relation

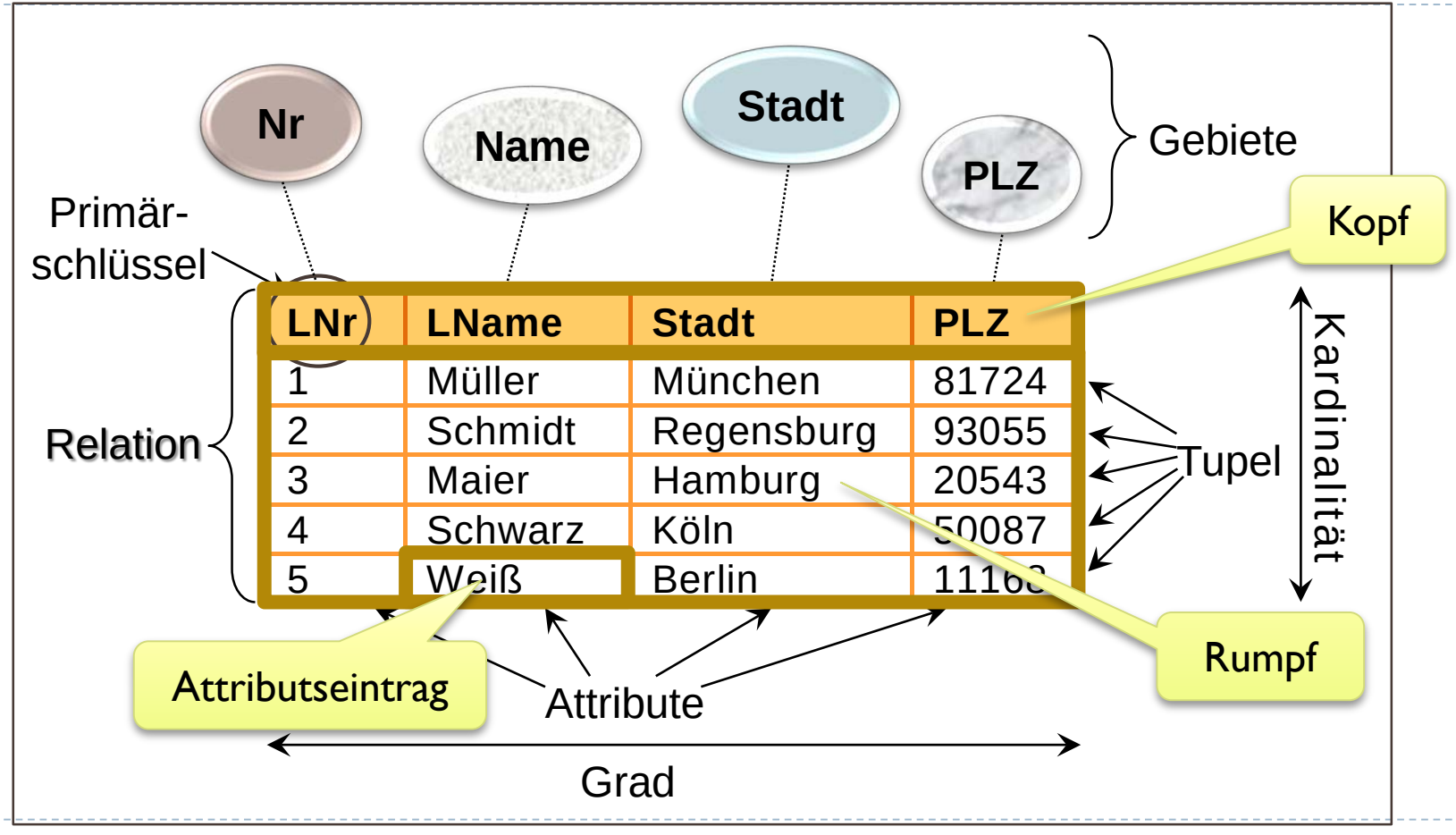
Tupel

Attribut

Kardinalität = 7

Grad = 2

Begriffe in relationalen Datenbanken (2)



Definition (Relation)

- Eine (normalisierte) **Relation** ist eine Tabelle, bestehend aus einem Kopf und einem Rumpf, mit folgenden vier Eigenschaften:

(1) Es gibt keine doppelten Tupel

Jede Zeile ist eindeutig

(2) Tupel sind nicht geordnet (z.B. von oben nach unten)

Die Reihenfolge spielt keine Rolle

Es gibt keine „erste“ oder „zweite“ Zeile

(3) Attribute sind nicht geordnet (z.B. von links nach rechts)

Die Reihenfolge spielt keine Rolle

Es gibt keine „erste“ oder „zweite“ Spalte

(4) Alle Attribute sind atomar

Es gibt nur Einzeleinträge, keine Aufzählungen oder Listen in einem Eintrag

Relation: (1) Keine doppelten Tupel

- Reduziert Redundanz ohne Informationsverlust
- Folgender doppelter Eintrag macht auch gar keinen Sinn:

VerkNr	VerkName	PLZ	VerkAdresse	Produktname	Umsatz
…	…	…	…	…	…
V1	Meier	80075	München	Kühlschrank	1000
V1	Meier	80075	München	Kühlschrank	1000
…	…	…	…	…	…

Relation: (2) Tupel sind nicht geordnet

- **Erleichtert das Einfügen neuer Zeilen**
  - Das DBMS entscheidet: Am Ende? In Lücke?
- **Erleichtert das Löschen von Zeilen**
  - Das DBMS entscheidet: Lücke? Nachrücken? Ersetzen?
- **Kein Informationsgewinn!**
  - Lieferant Maier steht an Position 17. Was bringt uns diese Info?
- **Aber: Performance**
  - Mit Sortierung könnte eventuell schneller zugegriffen werden

Relation: (3) Attribute sind nicht geordnet

- **Sehr seltene Änderungen bei Attributen**
  - Daher: kaum Nachteile oder Vorteile
- **Vermeidet Programmierschwäche**
  - Ausgabe der 7. Spalte ist nicht möglich
  - Zum Glück!
  - Beim Einfügen einer neuen Spalte wäre Programm falsch!
- **Konsequent**
  - Keine Tupelreihenfolge, also auch keine Attributreihenfolge

Relation: (4) Attribute sind atomar

- **Atomar heißt:**
  - Jeder Attributeintrag enthält nur einen Wert aus dem Definitionsgebiet
  - Aufzählungen sind nicht erlaubt
  - Listen sind nicht erlaubt
  - Atomar hat nichts damit zu tun, dass beispielsweise ein Wort aus einzelnen Buchstaben besteht!
- **Beispiel:** Attribut Stadt
  - In jeder Zeile steht eine Stadt (oder NULL)
  - Hat ein Mitarbeiter zwei Wohnsitze → Zwei Zeilen!

Nicht atomare Relation

- **Relation VerkaeuferProduktNF2**
  - Produkte zusammenfassen, Umsatz addieren

VerkNr	VerkName	PLZ	VerkAdresse	Produktname	Umsatz
V1	Meier	80075	München	Waschmaschine, Herd, Kühlschrank	17000
V2	Schneider	70038	Stuttgart	Herd, Kühlschrank	7000
V3	Müller	50083	Köln	Staubsauger	1000

- **Vorteile:** Kompakt und übersichtlich, keine Redundanz
- **Nachteile:** Komplexes Handling
  - Meier verkauft Staubsauger → nur Produktliste ergänzen
  - Schmidt verkauft Staubsauger → neue Zeile hinzufügen
  - Zeilen sind nicht mehr alle gleich lang!

Relationale Datenbank

- Eine relationale Datenbank ist eine Datenbank,

  die der Benutzer ausschließlich als eine Ansammlung von zeitlich variierenden, normalisierten Relationen passender Grade erkennt.

- Informell:
  - Eine relationale Datenbank ist eine Datenbank, die nur aus Relationen besteht.

Relationenarten

- **Basisrelationen**
  - Real existierende Relationen, persistenter Bestandteil der Datenbank
- **Sichten (Views)**
  - Virtuelle Relationen, abgeleitet aus Basisrelationen. Sie erscheinen dem Benutzer wie „normale“ Relationen.
- **Abfrageergebnisse**
  - Relationen, die temporär im Arbeitsspeicher während der Ausgabe existieren.
- **Temporäre Relationen**
  - Relationen, die nur temporär existieren. Sie werden bei bestimmten Ereignissen zerstört, etwa beim Beenden einer Transaktion.

Erzeugen von Relationen: SQL Befehle

- **CREATE TABLE** Tabellenname (. . .) ;
  - Erzeugen einer Basisrelation
- **CREATE VIEW** Sichtname AS . . . ;
  - Erzeugen einer Sicht
- **SELECT** Spalte **FROM** Tabelle . . . ;
  - Abfrage
- **CREATE TEMPORARY TABLE** Tabellenname (. . .) . . . ;
  - Erzeugen einer temporären Relation

Primärschlüssel (informell)

- **Der Primärschlüssel identifiziert jedes Tupel eindeutig**
- **Der Primärschlüssel besteht aus**
  - einem Attribut, z.B. LNr (Lieferantennummer)
  - mehreren Attributen (Primärschlüssel von VerkaeuferProdukt?)
- **Jede Relation besitzt einen Primärschlüssel**
  - Beweis:
    - Jedes Tupel ist eindeutig
    - Alle Attribute zusammen identifizieren daher jedes Tupel eindeutig
    - Alle Attribute zusammen könnten daher der Primärschlüssel sein
- **Was ist also der Primärschlüssel genau? → Klärung!**

Tabelle der chemischen Elemente

Protonen	Atomgewicht	Name	Symbol	Schmelzpkt.	Siedepkt.
1	1,008	Wasserstoff	H	-259	-253
2	4,003	Helium	He	-272	-269
3	6,941	Lithium	Li	180	1317
4	9,012	Berylium	Be	1278	2970
5	10,811	Bor	B	2300	2550
6	12,011	Kohlenstoff	C	3550	4827
7	14,007	Stickstoff	N	-210	-196
8	15,999	Sauerstoff	O	-218	-183
…	…	…	…		

- **Welches ist der eindeutige Identifikator (Primärschlüssel)?**
  - Protonenzahl, Atomgewicht, Name, Symbol?
  - Nicht geeignet: Schmelzpunkt, Siedepunkt (da nicht zwingend eindeutig)

Chemische Elemente: Primärschlüssel

- **Protonenzahl**
  - Jedes Element ist durch die Protonenzahl eindeutig identifiziert
  - Was in der Praxis gilt, sollte auch in Datenbank gelten → **Primärschlüssel**
- **Atomgewicht**
  - Nur zufällig haben alle Elemente unterschiedliches Atomgewicht
- **Name**
  - Jedes Element hat einen eindeutigen Namen, aber sprachabhängig
  - Wenn ein neues Element entdeckt wird, hat es noch keinen Namen!
- **Symbol**
  - Jedes chemische Element hat ein eindeutiges Symbol
  - Wenn ein neues Element entdeckt wird, hat es noch keinen Namen!

Definition (Superschlüssel)

- Ein eventuell aus mehreren einzelnen Attributen zusammen gesetztes Attribut heißt **Superschlüssel**, falls es eindeutig jedes Tupel identifiziert.

- **Superschlüssel in Tabelle der chemischen Elemente:**
  - (Protonen, Atomgewicht, Name, Symbol, Schmelzpunkt, Siedepunkt)
  - (Protonen, Atomgewicht)
  - (Name, Symbol)
  - Symbol
  - Protonen usw.

Definition (Schlüsselkandidat)

- Ein eventuell aus mehreren einzelnen Attributen zusammen gesetztes Attribut heißt **Schlüsselkandidat**, falls es
  - ein Superschlüssel ist und
  - minimal ist.

- **Schlüsselkandidaten in Tabelle der chemischen Elemente:**
  - Protonen
  - Name
  - Symbol

Definition (Primärschlüssel)

- Besitzt eine Relation mehrere Schlüsselkandidaten, so wird davon einer als **Primärschlüssel** ausgewählt.
- Alle anderen heißen **alternative Schlüssel**.

- Dies impliziert:
  - Gibt es nur einen Schlüsselkandidaten, so ist dieser Primärschlüssel.
- Primärschlüssel in Tabelle der chemischen Elemente:
  - Protonen

Relation Lagerbestand

Produktname	Produkttyp	Bestand	Preis
Staubsauger	T06	25	498
Staubsauger	T17	17	219
...	...	...	...
Küchenherd	T04	10	1598
Küchenherd	T06	7	1998

← Primärschlüssel →

Superschlüssel:
Produktname + Produkttyp + Bestand + Preis
Produktname + Produkttyp + Bestand
Produktname + Produkttyp + Preis
Produktname + Produkttyp

Minimal!!!!!

Einziger Schlüsselkandidat
also: **Primärschlüssel**

also: Schlüsselkandidat

Abfragen auf Schlüsselkandidaten

- Abfragen auf Schlüsselkandidaten liefern eindeutige Ergebnisse!

```
SELECT   Produktname, Preis
 FROM    Lagerbestand
WHERE    Produktname = 'Staubsauger'
 AND     Produkttyp = 'T06' ;
```

Liefert eindeutiges Ergebnis!

Wir benötigen je eine Variable für Produktname und Preis

```
SELECT   Produktname, Preis
 FROM    Lagerbestand
WHERE    Produktname = 'Küchenherd';
```

Liefert mehrdeutiges Ergebnis!

Wir benötigen ein unbekannt großes Feld für Produktname und Preis

Integrität in Datenbanken

- Integrität kommt von integer
- Eine integre Person ist eine Person, auf die ich mich verlassen kann
- Eine integre Datenbank ist eine Datenbank,
  - auf die ich mich verlassen will
  - deren Daten in sich konsistent sind
  - deren Daten korrekt sind und mit der realen Welt übereinstimmen
  - deren Daten vor fremden Blicken geschützt sind

Arten der Integrität

- **Physische Integrität**
  - Vollständigkeit der physischen Speicherstrukturen
  - verantwortlich: Datenbank, Betriebssystem
- **Ablaufintegrität**
  - Korrektheit der Programme, z.B. keine Endlosschleife
  - verantwortlich: Anwendungsprogrammierer, Datenbankdesigner
- **Zugriffsberechtigung**
  - Korrekte Zugriffsrechte
  - verantwortlich: Datenbank-Administrator
- **Semantische Integrität**
  - Übereinstimmung der Daten aus der nachzubildenden realen Welt mit den abgespeicherten Informationen
  - verantwortlich: Datenbankdesigner, Programmierer, Anwender

Wichtig für Administrator

Zu berücksichtigen im Programm

Zu berücksichtigen im Datenbankdesign

Folgerungen zur semantischen Integrität

- Gebiete D_j so weit wie möglich einschränken
- Attributwerte v_{ij} aus D_j auswählen (für alle i)
- Soweit möglich: Nummern automatisch vergeben
- Damit können Eingabefehler reduziert werden.
- Beispiel:
  - In einer Firma arbeiten Mitarbeiter zwischen 10 und 40 Stunden pro Woche
  - → $D_{Arbeitszeit} = [10..40]$ und nicht: $D_{Arbeitszeit}$ = Int-Wert

Entitäts-Integritätsregel

- **Erste Integritätsregel**
  - **Keine Komponente** des Primärschlüssels einer Basisrelation darf **nichts** enthalten

- **Wichtig**
  - Diese Regel gilt nur für Basisrelationen
  - Diese Regel gilt nicht für alternative Schlüssel
  - Kein Teilattribut eines Primärschlüssels darf leer sein
  - Es gibt einen eigenen „Nichts“-Wert; in SQL: **NULL**
  - Die Datenbank soll die 1. Integritätsregel immer überprüfen!

Definition (Fremdschlüssel)

- Ein Attribut einer Basisrelation heißt **Fremdschlüssel**,
  - falls das ganze **Attribut** nichts oder einen definierten Inhalt enthält,
  - eine Basisrelation existiert, so dass jeder definierte Wert des Fremdschlüssels einem Wert des Primärschlüssels jener Basisrelation entspricht.
- **Wichtig:**
  - Ein zusammengesetzter Fremdschlüssel darf nicht in einigen Teilattributen NULL-Werte besitzen und in anderen nicht!
  - Jeder Wert eines Fremdschlüssels bezieht sich auf einen existierenden Primärschlüsselwert!

Beispiel zu Fremdschlüsseln (1)

Relation Auftrag

Auftrnr	Datum	Kundnr	Persnr
1	04.01.2013	1	2
2	06.01.2013	3	5
3	07.01.2013	4	2
4	18.01.2013	6	5
5	03.02.2013	1	2

Derzeit nur Werte zwischen 1 und 9 erlaubt

Relation Personal (Auszug)

Persnr	Name	Ort	Vorgesetzt	Gehalt
1	Maria Forster	Regensburg	NULL	4800.00
2	Anna Kraus	Regensburg	1	2300.00
3	Ursula Rank	Frankfurt	6	2700.00
4	Heinz Rolle	Nürnberg	1	3300.00
5	Johanna Köster	Nürnberg	1	2100.00
6	Marianne Lambert	Landshut	NULL	4100.00
7	Thomas Noster	Regensburg	6	2500.00
8	Renate Wolters	Augsburg	1	3300.00
9	Ernst Pach	Stuttgart	6	800.00

Beispiel zu Fremdschlüsseln (2)

Auftrnr	Datum	Kundnr	Persnr
1	04.01.2013	1	2
2	06.01.2013	3	5
3	07.01.2013	4	2
4	18.01.2013	6	5
5	03.02.2013	1	2

Relation Auftrag

Derzeit nur Werte zwischen 1 und 6 erlaubt

Relation Kunde

Nr	Name	Strasse	PLZ	Ort
1	Fahrrad Shop	Obere Regenstr. 4	93059	Regensburg
2	Zweirad-Center Staller	Kirschweg 20	44276	Dortmund
3	Maier Ingrid	Universitätsstr. 33	93055	Regensburg
4	Rafa - Seger KG	Liebigstr. 10	10247	Berlin
5	Biker Ecke	Lessingstr. 37	22087	Hamburg
6	Fahrräder Hammerl	Schindlerplatz 7	81739	München

Referenz-Integritätsregel

- **Zweite Integritätsregel**
  - Eine relationale Datenbank enthält keinen Fremdschlüsselwert (ungleich *Null*), der im dazugehörigen Primärschlüssel nicht existiert.

- **Nichts Neues: Teil der Definition des Fremdschlüssels!**
- **Diese Regel ist extrem wichtig!**
- **Diese Regel muss immer eingehalten werden!**

Beispiele zur 2. Integritätsregel

- **1. Fall**
  - Ein Auftrag wird vergeben. Eine Kundennr. 13 wäre nicht erlaubt!
  - Hier werden selten Fehler gemacht, außer aus Versehen!
- **2. Fall**
  - Frau Köster mit Persnr 5 scheidet aus der Firma aus.
  - Wir löschen das Tupel mit Persnr 5
  - Jetzt enthält der Fremdschlüssel *Persnr* in Relation *Auftrag* einen nicht erlaubten Wert!
  - Aber: Wie verhindern wir einen solchen Fall?

Sicherstellen der 2. Integritätsregel

- Der Mensch macht Fehler!
- → Die Datenbank muss die 2. Integritätsregel garantieren

- Szenario:
  - Frau Köster scheidet aus, das Tupel soll gelöscht werden
  - Die Datenbank stellt fest, dass ein Fremdschlüssel dazu existiert
- Die Datenbank muss reagieren (aber wie?):
  - Sie verhindert das Löschen des Tupel mit Fehlermeldung
  - Oder: Sie entfernt auch die Einträge im Fremdschlüssel (NULL!)
  - Oder: Sie löscht auch die Tupel, die auf Frau Köster verweisen

SQL und die 2. Integritätsregel (1)

- **Fremdschlüsselbedingungen in SQL**
  - ON DELETE **NO ACTION**
  - ON DELETE **SET NULL**
  - ON DELETE **CASCADE**
- **Funktionsweise**
  - Wird ein Tupel gelöscht, auf den ein Fremdschlüssel mit obiger Bedingung verweist, dann
    - wird das Löschen dieses Tupel verhindert (Nichtstun, No Action)
    - wird der darauf verweisende Fremdschlüssel auf Null gesetzt (Set Null)
    - wird auch das den Fremdschlüssel enthaltene Tupel gelöscht (Cascade)

SQL und die 2. Integritätsregel (2)

- **Analog wird das Ändern eines Primärschlüssel behandelt:**
  - ON UPDATE **NO ACTION**
  - ON UPDATE **SET NULL**
  - ON UPDATE **CASCADE**
- **Funktionsweise**
  - Wird ein Primärschlüsselwert geändert, auf den ein Fremdschlüssel mit obiger Bedingung verweist, dann
    - wird das Ändern dieses Tupel verhindert (Nichtstun, No Action)
    - wird der darauf verweisende Fremdschlüssel auf Null gesetzt (Set Null)
    - wird der Fremdschlüsselwert mit geändert (Cascade)

Kaskadierendes Löschen

- **Auf ein Tupel mit einem Fremdschlüssel kann wiederum ein Fremdschlüssel verweisen, usw.**
- **Im Falle von ON DELETE CASCADE gilt:**
  - Ein Tupel soll gelöscht werden
  - Ein Fremdschlüssel verweist auf dieses Tupel, das Tupel mit diesem Fremdschlüssel wird mit gelöscht
  - Auf letztes Tupel verweist ein weiterer Fremdschlüssel, der entsprechende Eintrag wird dann auch gelöscht usw.
- Wir nennen dies: **Kaskadierendes Löschen**

Beispiel zum kaskadierenden Löschen

Auftrnr	Datum	Kundnr	Persnr
1	04.01.2013	1	2
2	06.01.2013	3	5
3	07.01.2013	4	2
4	18.01.2013	6	5
5	03.02.2013	1	2

- Für alle Fremdschlüssel gelte ON DELETE CASCADE
- Frau Forster soll gelöscht werden

Welche Tupel werden noch gelöscht?

Persnr	Name	Ort	Vorgesetzt	Gehalt
1	Maria Forster	Regensburg	NULL	4800.00
2	Anna Kraus	Regensburg	1	2300.00
3	Ursula Rank	Frankfurt	6	2700.00
4	Heinz Rolle	Nürnberg	1	3300.00
5	Johanna Köster	Nürnberg	1	2100.00
6	Marianne Lambert	Landshut	NULL	4100.00
7	Thomas Noster	Regensburg	6	2500.00
8	Renate Wolters	Augsburg	1	3300.00
9	Ernst Pach	Stuttgart	6	800.00

Kaskadierendes Löschen u. Transaktion

- **Merke:**
  - Ein kaskadierendes Löschen wird als **Transaktion** ausgeführt
    - Entweder wird das komplette kaskadierende Löschen ausgeführt
    - Oder es wird nichts gelöscht
- **Folgerung:**
  - Gibt es in der Kette ein CASCADE, so wird entsprechend weiter gelöscht
  - Gibt es in der Kette ein SET NULL, so endet dieses Glied
  - Gibt es in der Kette ein NO ACTION, so wird das komplette kaskadierende Löschen **rückgängig** gemacht (Transaktion)!

Fremdschlüssel und NULL-Werte

- NULL-Werte können in einem Attribut explizit verboten werden. Wir setzen dazu im CREATE-TABLE-Befehl:
  - NOT NULL
- Es gilt:
  - Ist ein Fremdschlüssel auch Primärschlüssel oder ist explizit NOT NULL gesetzt,
    - so sind NULL-Werte nicht erlaubt
    - so darf ON DELETE SET NULL nicht verwendet werden
    - so darf ON UPDATE SET NULL nicht verwendet werden

Begriffe zur Relationalen Algebra

- Menge
  - Behälter, der unterscheidbare Elemente enthält
- Operator
  - Vorschrift zur Überführung eines oder mehrerer Elemente in ein anderes Element
- Unärer Operator
  - Vorschrift zur Überführung eines Elements
- Binärer Operator
  - Vorschrift zur Überführung von zwei Elementen
- Relationale Algebra
  - Abfragesprache auf relationale Datenbanken, in der geeignete Operatoren definiert sind

$\mathfrak{R}$ = Menge aller Relationen

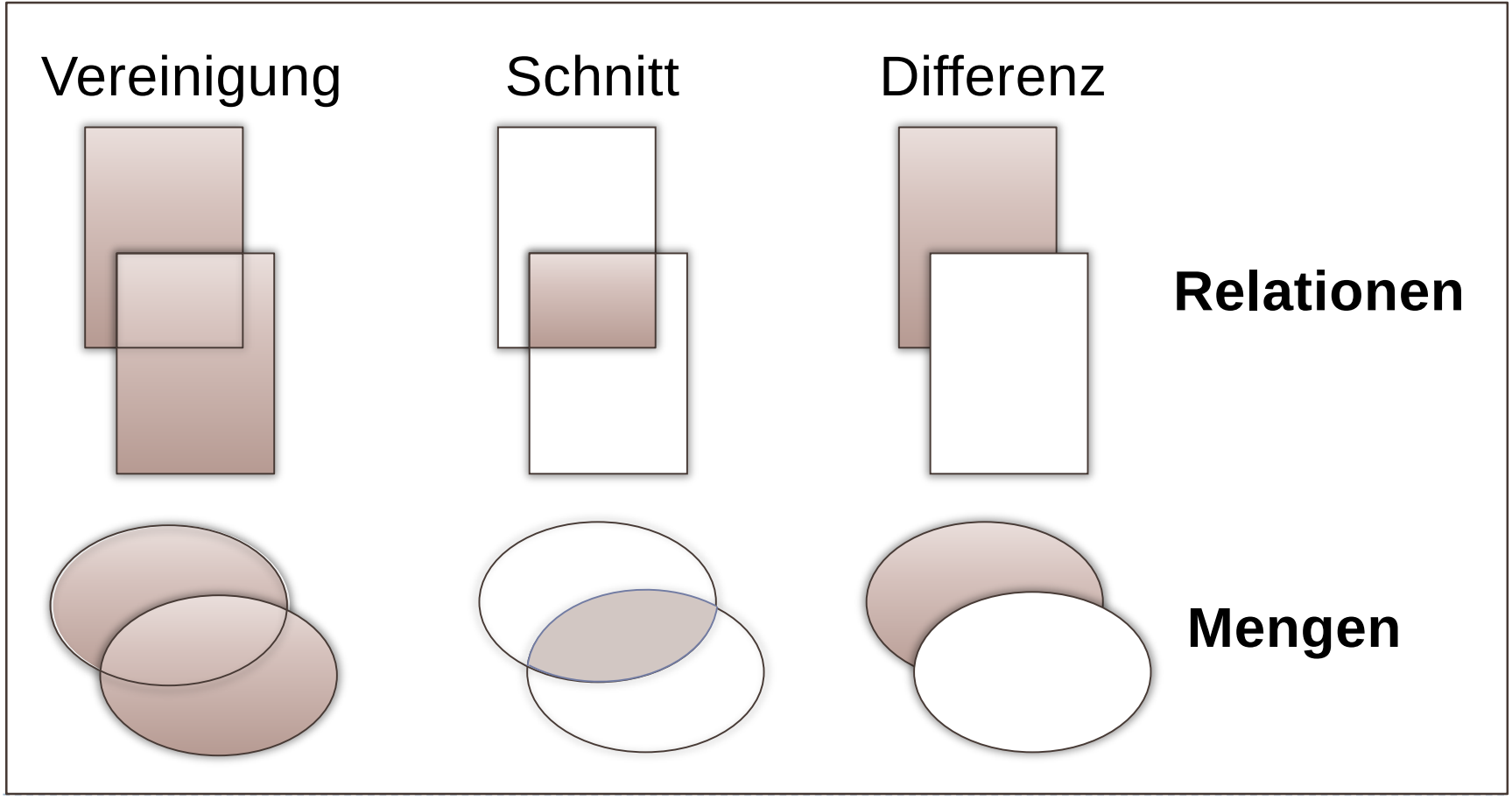
$op : \mathfrak{R} \rightarrow \mathfrak{R}$

$op : \mathfrak{R} \times \mathfrak{R} \rightarrow \mathfrak{R}$

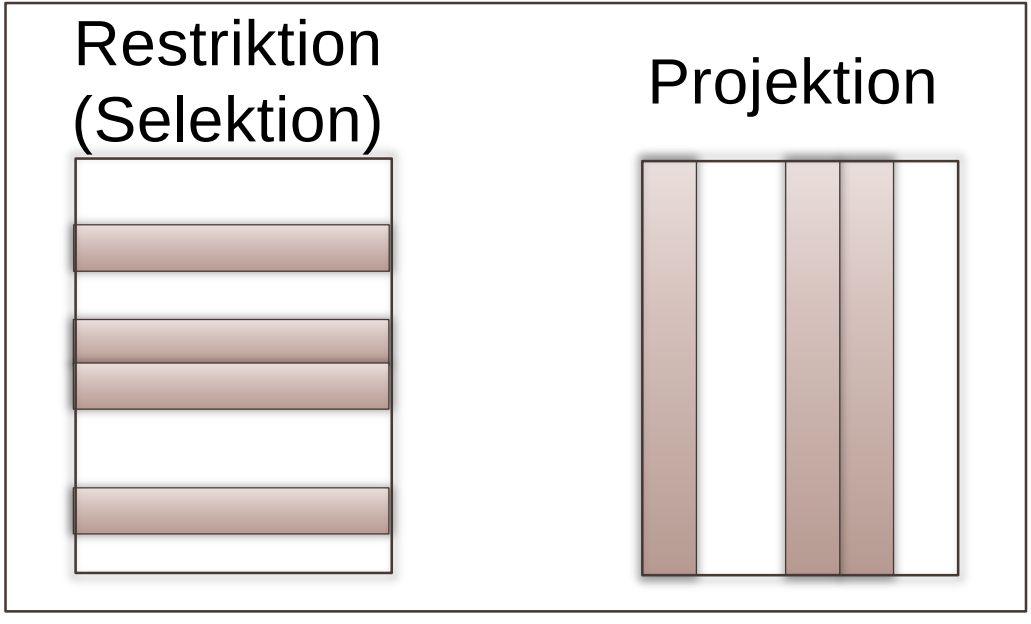
Alle neun relationalen Operatoren

	Operator	Beispiel
Vereinigung	$\cup$	$R1 \cup R2$
Schnitt	$\cap$	$R1 \cap R2$
Differenz	$\setminus$	$R1 \setminus R2$
Kreuzprodukt	$\times$	$R1 \times R2$
Restriktion	σ	$\sigma_{Bedingung}(R)$
Projektion	π	$\pi_{Auswahl}(R)$
Verbund	$\bowtie$	$R1 \bowtie R2$
Division	$\div$	$R1 \div R2$
Umbenennung	ρ	$\rho_{Rneu}(R)$

Vereinigung, Schnitt, Differenz

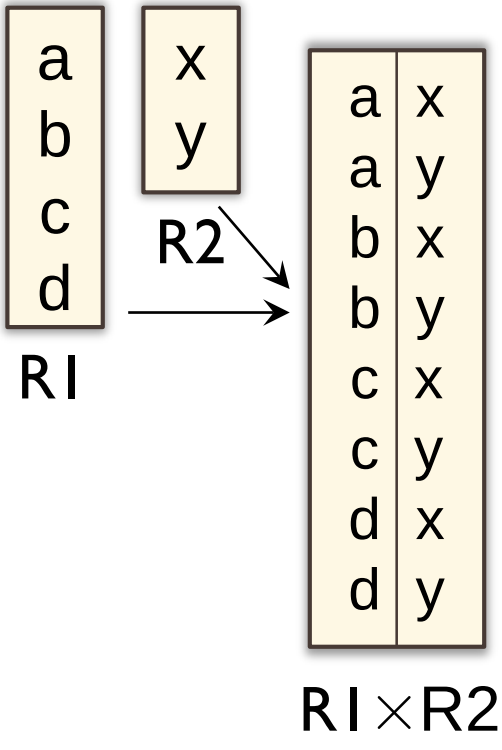


Projektion, Restriktion



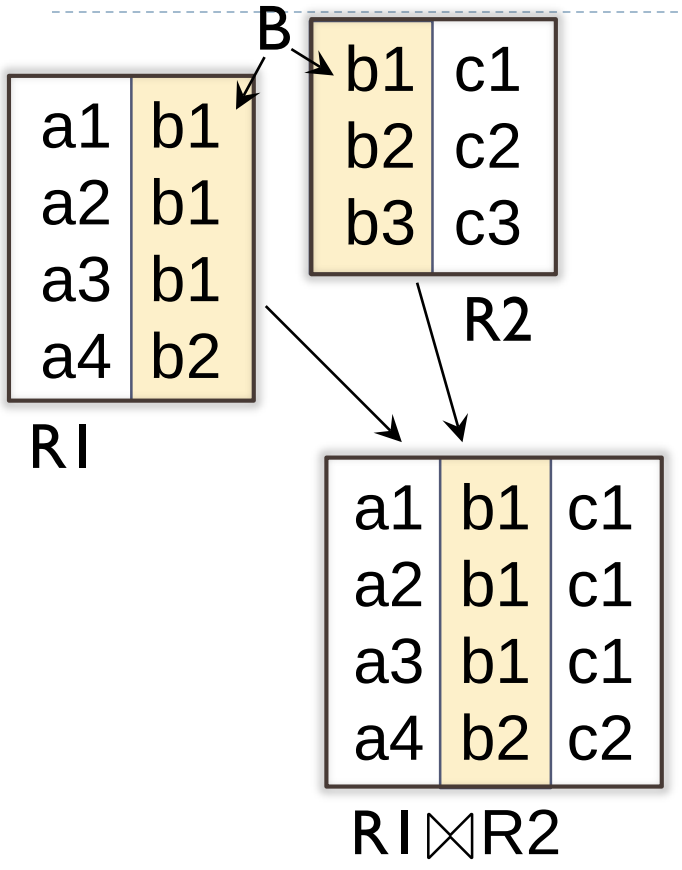
- Projektion:
  - Einschränkung auf weniger Attribute (Spalten)
- Restriktion:
  - Einschränkung auf weniger Tupel (Zeilen)

Kreuzprodukt



- Jede Zeile der einen Tabelle wird mit jeder Zeile der anderen Tabelle verknüpft
  - Sei n=Kardinalität(R1),
    sei m=Kardinalität(R2),
  - dann: **Kardinalität(R1×R2) = n · m**

(Natürlicher) Verbund



- Voraussetzung:
  - R1 und R2 besitzen Attribut mit gleichem Namen (hier: Spalte B)
- Verbund verbindet alle Tupel, die im Attribut B gleiche Einträge haben
- Verbund führt Attribut B nur einmal auf

Begriffe zum Verbund

- **Natürlicher Verbund (Natural Join):** $\bowtie$
  - Alle Attribute gleichen Namens dienen als Verbindung, Überprüfung auf Gleichheit
- **Equi-Join:** $\bowtie_{R1.A=R2.B}$
  - Die angegebenen Attribute R1.A und R2.B dienen als Verbindung, Überprüfung auf Gleichheit
- **Theta-Join:** $\bowtie_{R1.A\ op\ R2.B}$
  - Die angegebenen Attribute R1.A und R2.B dienen als Verbindung, Überprüfung mittels Operator op (z.B. <, <=)
- **Outer Join:** $⟕$, $⟖$, $⟗$
  - Erweiterung: Auch nicht betroffene Tupel werden aufgelistet

Verbund: Ein Beispiel

Auftrnr	Datum	Kundnr	Persnr
1	04.01.2013	1	2
2	06.01.2013	3	5
3	07.01.2013	4	2
4	18.01.2013	6	5
5	03.02.2013	1	2

- Verbund über Persnr
- Nur Persnr 2 und 5 bleiben übrig

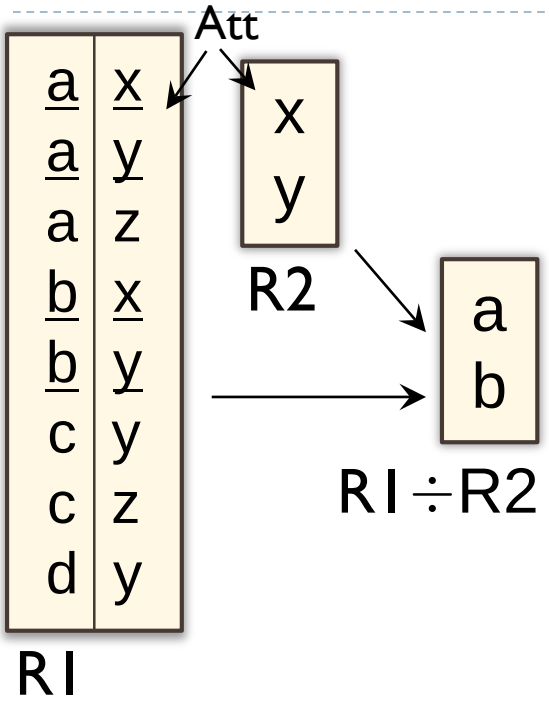
Persnr	Name	Ort	Vorgesetzt	Gehalt
1	Maria Forster	Regensburg	NULL	4800.00
2	Anna Kraus	Regensburg	1	2300.00
3	Ursula Rank	Frankfurt	6	2700.00
4	Heinz Rolle	Nürnberg	1	3300.00
5	Johanna Köster	Nürnberg	1	2100.00
6	Marianne Lambert	Landshut	NULL	4100.00
7	Thomas Noster	Regensburg	6	2500.00
8	Renate Wolters	Augsburg	1	3300.00
9	Ernst Pach	Stuttgart	6	800.00

Verbund: Das Ergebnis

AuftrNr	Datum	Kundnr	Persnr	Name	Vorgesetzt	Gehalt	Ort
1	04.01.13	1	2	Anna Kraus	1	3400.00	Regensburg
2	06.01.13	3	5	Joh. Köster	1	3200.00	Nürnberg
3	07.01.13	4	2	Anna Kraus	1	3400.00	Regensburg
4	18.01.13	6	5	Joh. Köster	1	3200.00	Nürnberg
5	06.02.13	1	2	Anna Kraus	1	3400.00	Regensburg

- Alle Attribute von Auftrag
- Alle Attribute von Personal
- Aber: Verknüpfendes Attribut Persnr nur einmal

Division



- Voraussetzung:
  - R1 enthält alle Attribute Att von R2
- Die Division liefert die restlichen Attribute von R1 (ohne Att)
- Die Division enthält alle Werte, die in R1 mit allen Attributen aus R2 verknüpft sind
  (im Beispiel unterstrichen)

Division: Beispiel (1)

Lieferung:

ANr	Liefnr	Lieferzeit	Nettopreis	Bestellt
500001	5	1	6.50	0
500002	2	4	71.30	10
500002	1	5	73.10	0
500003	3	6	5.60	0
500003	4	5	6.00	0
500003	2	4	5.70	0
500004	3	2	5.20	0
500004	4	3	5.40	0
500005	4	5	6.70	0
500006	1	1	31.00	0
500007	1	2	16.50	0

Artikelnr

Lieferantennr

- Lieferant 3 liefert Artikel 500003 und 500004.
- Gibt es weitere, die mindestens diese beiden Artikel liefern?

$R1 = \pi_{ANr,Liefnr}(Lieferung)$

$R2 = \sigma_{Liefnr=3}(R1)$

ANr	Liefnr
500003	3
500004	3

$R3 = \pi_{ANr}(R2)$

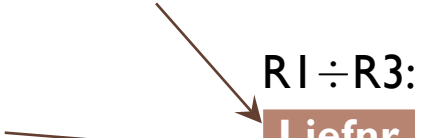
Division: Beispiel (2)

R1:

Liefnr	ANr
5	500001
2	500002
1	500002
3	500003
4	500003
3	500004
4	500004
4	500005
1	500006
1	500007

R3:

ANr
500003
500004



R1÷R3:

Liefnr
3
4

- Die Division liefert:
  - Lieferanten 3 und 4 liefern Artikel 500003 und 500004
  - Lieferant 3 war klar
- Ergebnis:
  - Lieferant 4 liefert alle Artikel, die auch Lieferant 3 liefert

Umbenennung

- Umbenennung ist Hilfsoperator, um Algebra zu vervollständigen
- Beispiel:
  - $R = \rho_{Nr->Kundnr}(Kunde) \bowtie Auftrag$
- Alternative (hier: Equi-Join):
  - $R = Kunde \bowtie_{Kunde.Nr=Auftrag.Kundnr} Auftrag$

Kommutativ- und Assoziativgesetze

- $A \cup B = B \cup A$
- $A \cap B = B \cap A$
- $A \times B = B \times A$
- $A \bowtie B = B \bowtie A$

- $A \cup (B \cup C) = (A \cup B) \cup C = A \cup B \cup C$
- $A \cap (B \cap C) = (A \cap B) \cap C = A \cap B \cap C$
- $A \times (B \times C) = (A \times B) \times C = A \times B \times C$
- $A \bowtie (B \bowtie C) = (A \bowtie B) \bowtie C = A \bowtie B \bowtie C$ Vorsicht!

Weitere Regeln (1)

- $\sigma_{BedingungA}(\sigma_{BedingungB}(R)) = \sigma_{BedingungB}(\sigma_{BedingungA}(R))$
  $= \sigma_{BedingungA\ AND\ BedingungB}(R)$
  $= \sigma_{BedingungA}(R) \cap \sigma_{BedingungB}(R)$

  Vertauschbarkeit von Bedingungen

- $\sigma_{BedingungA\ OR\ BedingungB}(R) = \sigma_{BedingungA}(R) \cup \sigma_{BedingungB}(R)$

  Bedingungen direkt verknüpfen

- $\sigma_{Bedingung}(R1 \cup R2) = \sigma_{Bedingung}(R1) \cup \sigma_{Bedingung}(R2)$

  Bedingungen auf einzelne Relationen einschränken

Weitere Regeln (2)

- $\sigma_{Bedingung}(R1 \bowtie R2) = \sigma_{Bedingung}(R1) \bowtie \sigma_{Bedingung}(R2)$

Bedingung direkt auf Relation anwenden.
Welche Seite ist performanter?

Spezialfall

- $\sigma_{Bedingung\_an\_R2}(R1 \bowtie R2) = R1 \bowtie \sigma_{Bedingung\_an\_R2}(R2)$

- $\pi_{Auswahl}(\sigma_{Bedingung}(R)) = \sigma_{Bedingung}(\pi_{Auswahl}(R))$

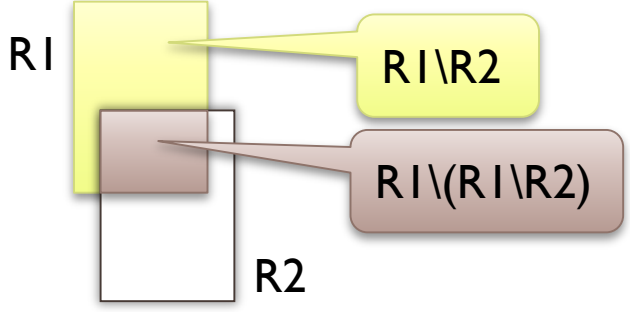
Der Name Division
ist berechtigt

Vertauschbarkeit von
Projektion und Restriktion

- $R1 = (R1 \times R2) \div R2$

Drei Operatoren sind „überflüssig“

- 3 Operatoren lassen sich aus den anderen 6 Operatoren ableiten!
- Dies sind
  - Schnitt: $R1 \cap R2 = R1 \setminus (R1 \setminus R2)$



  - Verbund: $R1 \bowtie R2 = \pi_{R1.X,R1.Y,R2.Z}(\sigma_{R1.Y=R2.Y}(R1 \times R2))$

gefolgt von Projektion (verbindendes Attribut nur einmal)

gefolgt von Restriktion

Kreuzprodukt

  - Division: $R1 \div R2 = \pi_{R1.X}(R1) \setminus (\pi_{R1.X}((\pi_{R1.X}(R1) \times R2) \setminus R1))$