Informatik
Datenbanken

5. Klasse TFO Brixen

Michael Mutschlechner

DATENBANKSPRACHEN AM BEISPIEL SQL

SQL

- ist eine Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen.

- Die Sprache basiert auf der relationalen Algebra,
  - einfache Syntax
  - semantisch an die englische Umgangssprache angelehnt

- Fast alle gängigen Datenbanksysteme unterstützen SQL
  - in unterschiedlichem Umfang und leicht voneinander abweichenden „Dialekten“
  - Unabhängigkeit der Anwendungen vom eingesetzten Datenbankmanagementsystem

- SQL steht im Allgemein für Structured Query Language
  - Die Bezeichnung leitet sich aber eigentlich von dem Vorgänger SEQUEL (Structured English Query Language)
    - in den 1970er Jahren durch Edgar F. Codd (IBM), Donald D. Chamberlin und Raymond F. Boyce entworfen

SQL

- Von SQL gibt es verschiedene Standards.
  - Erster SQL-Standard (1986 ANSI)
  - SQL2 bzw. SQL-92 (1992)
  - SQL3 bzw. SQL:1999 (1999 ISO)
  - SQL:2003 (2003)
  - SQL:2008 (2008)
- Die meisten Datenbankmanagementsysteme unterstützen SQL2.
  - neuere Versionen sind in der Regel nur teilweise oder gar nicht umgesetzt

- SQL-Befehle lassen sich unterteilen in
  - (DDL) Befehle zur Definition des Datenbankschemas
  - (DML) Befehle zur Datenmanipulation (Ändern, Einfügen, Löschen)
  - (DRL) Befehle zur Abfrage von Daten
  - (DCL) Befehle für die Rechteverwaltung und Transaktionskontrolle.

SQL: Allgemeines

- Bei den **Befehlen** arbeitet SQL ohne Berücksichtigung von Groß- und Kleinschreibung
  - *case insensitive*
- Tabellen und Attributnamen (Spaltennamen) können aber je nach Einstellungen case sensitive sein!
- SQL-Befehle werden mit einem Semikolon abgeschlossen.
- Ein Befehl kann beliebig auf eine oder mehrere Zeilen verteilt werden
  - Lesbarkeit
- Für eigene Bezeichner, d. h. die Namen von Tabellen, Spalten oder eigenen Funktionen gilt:
  - Schlüsselwörter vermeiden
    - führt schnell zu Problemen auch dort, wo es möglich wäre
  - Beginnen in der Regel mit einem Buchstaben oder dem Unterstrich
    - a…z A…Z _
    - Danach folgen beliebig Ziffern und Buchstaben
  - Inwieweit andere Zeichen und länderspezifische Buchstaben (Umlaute) möglich sind, hängt vom DBMS ab.
    - Nicht empfohlen

SQL: Kommentare

- Kommentare können in SQL-Befehle fast beliebig eingefügt werden (nur die Schlüsselwörter dürfen natürlich nicht „zerrissen“ werden).
- Es gibt zwei Arten von Kommentaren:
  - -- (doppelter Bindestrich, am besten mit Leerzeichen dahinter)
    - Alles von den beiden Strichen an (einschließlich) bis zum Ende dieser Zeile gilt als Kommentar und nicht als Bestandteil des Befehls
  - /* (längerer Text, gerne auch über mehrere Zeilen) */
    - Alles, was zwischen '/*' und '*/' steht (einschließlich dieser Begrenzungszeichen), gilt als Kommentar und nicht als Bestandteil des Befehls

- **DDL**

DDL: Grundsätzlicher Aufbau

```
BEFEHL OBJEKTTYP <Objektname> [<weitere Angaben>]
```

- Schreibweise:
  - [ ]: Optionale Argumente
  - < >: Variable Argumente

Datenbank erstellen

- Der Befehl zum Erstellen einer Datenbank lautet:

```
CREATE DATABASE <name> [ <Optionen> ] ;
```

- Zu den <Optionen> gehören z. B. der Benutzername des Eigentümers der Datenbank mit seinem Passwort, der Zeichensatz mit Angaben zur Standardsortierung, die Aufteilung in eine oder mehrere Dateien usw.

- Wegen der vielen Möglichkeiten ist zu empfehlen, dass eine Datenbank nicht per SQL-Befehl, sondern innerhalb einer Benutzeroberfläche erstellt wird. (später)

Datenbank löschen

- DROP DATABASE <name> ;

- Die ganze Information in dieser Datenbank geht verloren!

MySQL: Einführung

- MySQL ist eines der weltweit verbreitetsten relationalen Datenbankverwaltungssysteme.
- Als Open-Source-Software sowie als kommerzielle Enterpriseversion für verschiedene Betriebssysteme verfügbar
- Grundlage für viele dynamische Webauftritte
- Wurde seit 1994 vom schwedischen Unternehmen MySQL AB entwickelt.
  - Im Februar 2008 vonSun Microsystems übernommen
    - seinerseits im Januar 2010 von Oracle aufgekauft
- Der Name MySQL setzt sich zusammen aus dem Vornamen My, den die Tochter des MySQL AB Mitbegründers Michael Widenius trägt, und SQL

MySQL: Einführung

- Mit MySQL-Server verbinden:

```
mysql -u <user> [–h host] –p
```

- Datenbanken auflisten:

```
SHOW DATABASES;
```

- Datenbank verwenden:

```
USE datenbankname;
```

- Relationen auflisten:

```
SHOW TABLES;
```

- Details zu einer Relation anzeigen:

```
DESCRIBE <relation>;
```

Relationen erzeugen

- Erzeugen einer Relation (Minimalform)

```
CREATE TABLE Kurs(KursNr char(3), Titel varchar(20));
```

- Wirkung:
  - 1. Es wird eine Relation „Kurs“ angelegt.
  - 2. Die Tabelle hat zwei Attribute KursNr und Titel
  - 3. KursNr hat den Attributtyp Character, feste Länge, genau 3
  - 4. Titel hat den Attributtyp Character, variable Länge, max. 20
  - 5. Für beide Attribute sind Nullwerte (dies ist der Default) zugelassen.
  - 6. Wir hätten dies auch explizit festlegen können mittels

```
... KursNr CHAR(3) NULL, Titel VARCHAR NULL
```

Relationen erzeugen

- Weiteres Beispiel:

```
CREATE TABLE Persons(
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

Datentypen (Auszug)

- Ganze Zahlen
  - `INTEGER`
- Gleitkommazahl
  - `FLOAT`
  - `REAL`
  - `DOUBLE`
- Festkommazahl
  - `DECIMAL(g,k)`
  - insgesamt g Stellen, davon k Nachkommastellen
- Zeichenkette
  - `CHAR(Länge)`
  - feste Länge, max. 254 Zeichen. Bei Länge=1 auch einfach nur CHAR
- Zeichenkette
  - `VARCHAR(Länge)`
  - variable Länge, max. 4.000 Zeichen
- Datum
  - `DATE`
  - `DATETIME`
- …

Aufgabe

- Recherchiere die SQL-Datentypen und fertige eine persönliche Liste an.
- Teste die Befehle auf dem linuxserver
  - Benutzer:
    - Dein Benutzer stxxxyyy (xxx = erste drei Buchstaben deines Nachnamens, yyy = erste drei Buchstaben deines Vornamens)
  - Kennwort:
    - mypass

Spezifikation von Integritätsbedingungen

- Festlegung des Primärschlüssels
  - Als Primärschlüssel sind nur solche Attribute oder Attributkombinationen erlaubt, für die NOT NULL (bzw. ein anderer Defaultwert) spezifiziert wurde.
    - Anstelle von NOT NULL kann man auch DEFAULT <expr> angeben
  - Das DBS gewährleistet bei Einfüge-, Lösch- und Änderungsoperationen, dass der Wert dieses Attributes bzw. dieser Attributkombination innerhalb dieser Tabelle stets eindeutig ist.
    - Die Änderungsoperation wird ansonsten zurückgewiesen.
  - Je Relation kann max. ein Primärschlüssel deklariert werden

```
CREATE TABLE Kurs
( KursNr CHAR(3) NOT NULL,
  Titel VARCHAR(20),
  PRIMARY KEY(KursNr)
);
```

alternativ:

```
CREATE TABLE Kurs
( KursNr CHAR(3) NOT NULL PRIMARY KEY,
  Titel VARCHAR(20));
```

Spezifikation von Integritätsbedingungen

- Fremdschlüssel und Fremdschlüsselbedingungen
  - DBS-seitiges Erzwingen korrekter Fremdschlüsselwerte:
  - Beispiel: Angebot(AngNr, KursNr, Datum)

```
CREATE TABLE Angebot
( AngNr INTEGER NOT NULL,
  KursNr CHAR(3) NOT NULL,
  Datum DATE,
  PRIMARY KEY(AngNr, KursNr),
  FOREIGN KEY(KursNr) REFERENCES Kurs(KursNr)
);
```

alternativ:

```
CREATE TABLE Angebot
( AngNr INTEGER NOT NULL,
  KursNr CHAR(3) NOT NULL REFERENCES Kurs(KursNr),
  Datum DATE,
  PRIMARY KEY(AngNr, KursNr)
);
```

References

- Wirkungsweise von REFERENCES
  - Beim Einfügen in der „Kind-Tabelle“ muss der Fremdschlüsselwert als Primärschlüsselwert in der übergeordneten Tabelle („Vater-Tabelle“) vorhanden sein.
  - Beim Löschen in der „Vater-Tabelle“ dürfen in den abhängigen Tabellen keine „Waisenkinder“ entstehen.
  - **Ansonsten wird die Operation zurückgewiesen!**

Weiteres Beispiel

- Dienstwagen(ID, Kennzeichen, Farbe)

```
CREATE table Dienstwagen
( ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  Kennzeichen VARCHAR(30) NOT NULL,
  Farbe VARCHAR(30)
);
```

- ID ist eine ganze Zahl, darf nicht NULL sein, wird automatisch hochgezählt und dient gleichzeitig als Primärschlüssel.
- Das Kennzeichen ist eine Zeichenkette von variabler Länge (maximal 30 Zeichen), die unbedingt erforderlich ist.
- Die Farbe ist ebenfalls eine Zeichenkette, deren Angabe entfallen kann.

Beispiel zum Ausprobieren

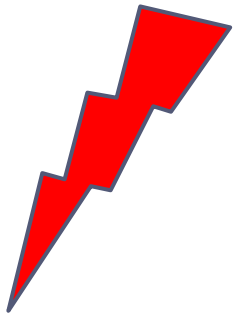
```
CREATE TABLE products
( product_id INT PRIMARY KEY,
  product_name VARCHAR(50) NOT NULL,
  category VARCHAR(25)
);

CREATE TABLE inventory
( inventory_id INT PRIMARY KEY,
  product_id INT NOT NULL REFERENCES products (prod_id),
  quantity INT,
  min_level INT,
  max_level INT
);
```

- Sind die Anweisungen korrekt? Warum funktionieren sie (nicht)?

Eigenheit von MySQL

- Aus dem MySQL Reference Manual
  - Furthermore, MySQL parses but ignores ”inline REFERENCES specifications” (as defined in the SQL standard) where the references are defined as part of the column specification. MySQL accepts REFERENCES clauses only when specified as part of a separate FOREIGN KEY specification.



Aufgabe

- Erstelle das Schema aus einem der ausgearbeiteten Relationenmodelle in mysql!
  - Überprüfe anschließend mit dem Befehl ‚describe‘

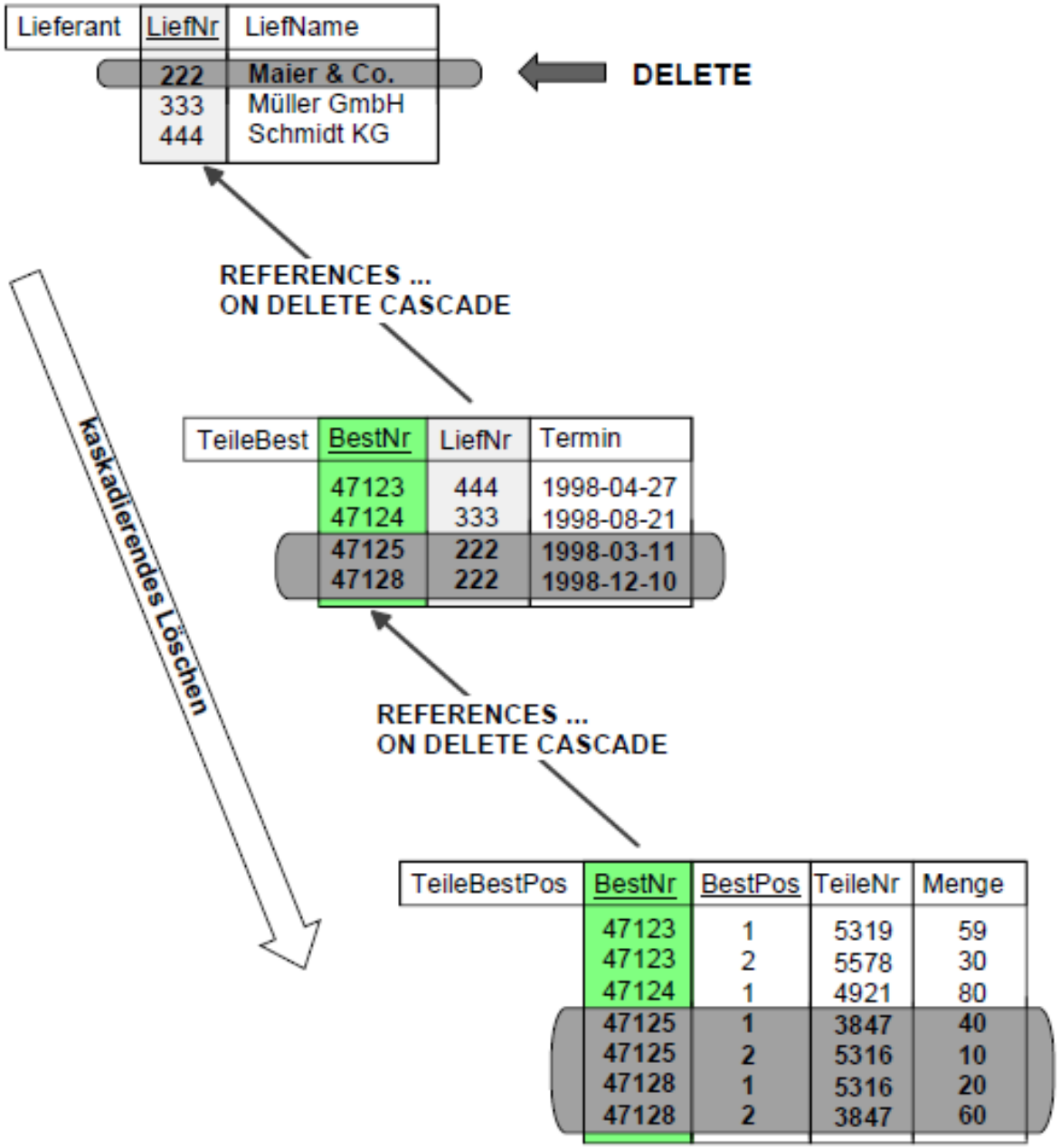
Kaskadierendes Löschen

- Angenommen, beim Löschen eines Kurs-Tupels sollen auch alle zughörigen Tupel in der Angebot-Tabelle mitgelöscht werden:

```
CREATE TABLE Angebot
( AngNr INTEGER NOT NULL,
  KursNr CHAR(3) NOT NULL,
  Datum DATE,
  Ort VARCHAR(20),
  PRIMARY KEY(AngNr, KursNr),
  FOREIGN KEY(KursNr) REFERENCES Kurs(KursNr) ON DELETE CASCADE
);
```

  - Anstelle von „... ON DELETE CASCADE“ kann z.B. auch „... ON DELETE SET NULL“ spezifiziert werden.

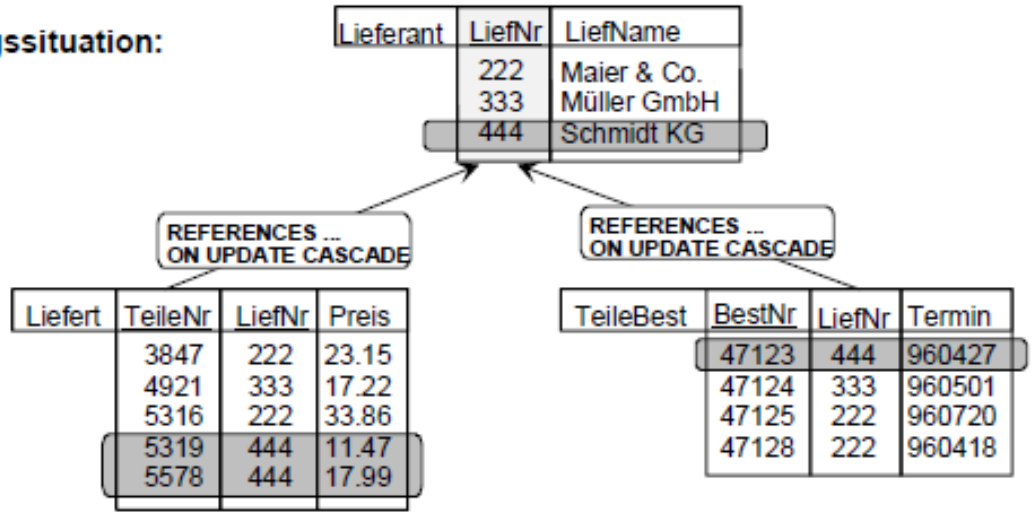
- Aufrechterhaltung konsistenter Fremdschlüssel-Beziehungen in realen DB-Anwendungen ist oft nicht trivial.
- Die Deklaration solcher referentieller Integritätsbedingungen (referential integrity constraints) (mit oder ohne kaskadierendes Löschen) ist das Mittel der Wahl zur Gewährleistung konsistenter Fremdschlüssel-Beziehungen in relationalen DBS.
- Das Löschen kann sich „kaskadierend“ über mehrere Tabellen hinweg auswirken, wenn zwischen diesen entsprechende Abhängigkeitsbeziehungen (via REFERENCES ... DELETE) definiert wurden (siehe nächste Folie).
- Das Löschen wird nur komplett oder gar nicht ausgeführt. Ist irgendwo in der Kette eine ... ON DELETE-Restrikt-Bedingung vorhanden, so wird die Änderungsoperation (komplett) zurückgewiesen.


Lieferant
LiefNr
LiefName
222
Maier & Co.
333
Müller GmbH
444
Schmidt KG
DELETE
REFERENCES ...
ON DELETE CASCADE
kaskadierendes Löschen
TeileBest
BestNr
LiefNr
Termin
47123
444
1998-04-27
47124
333
1998-08-21
47125
222
1998-03-11
47128
222
1998-12-10
REFERENCES ...
ON DELETE CASCADE
TeileBestPos
BestNr
BestPos
TeileNr
Menge
47123
1
5319
59
47123
2
5578
30
47124
1
4921
80
47125
1
3847
40
47125
2
5316
10
47128
1
5316
20
47128
2
3847
60

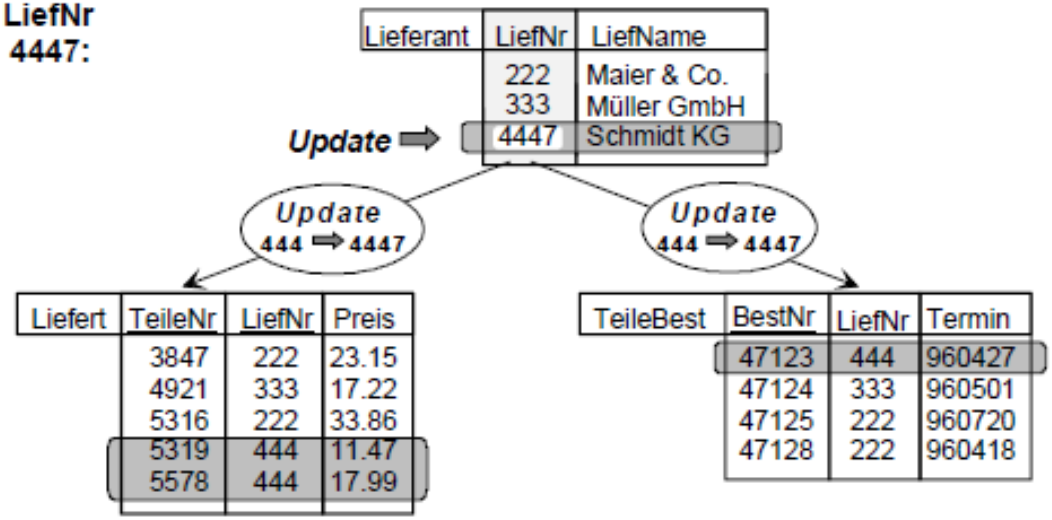
Kaskadierendes Ändern

- Analog zum kaskadierenden Löschen, sieht SQL92 auch das kaskadierende Ändern vor:
  - … ON UPDATE CASCADE
    - ON UPDATE NO ACTION
    - ON UPDATE RESTRICT
- Wirkung: Propagation des geänderten Attributwertes in die referenzierenden Fremdschlüssel der anderen Relationen

Ausgangssituation:



Update LiefNr
444 → 4447:



Resultat:

Lieferant	LiefNr	LiefName
	222	Maier & Co.
	333	Müller GmbH
	4447	Schmidt KG

Liefert	TeileNr	LiefNr	Preis
	3847	222	23.15
	4921	333	17.22
	5316	222	33.86
	5319	4447	11.47
	5578	4447	17.99

TeileBest	BestNr	LiefNr	Termin
	47123	4447	960427
	47124	333	960501
	47125	222	960720
	47128	222	960418

Default-Werte

- Das Schlüsselwort DEFAULT wird verwendet, um Standardwerte für Attribute anzugeben.
- Dieser Wert wird eingefügt, wenn beim Einfügen kein anderer angegeben wird.

```
CREATE TABLE Persons
( P_Id int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255) DEFAULT 'Brixen'
);
```

UNIQUE-Klausel

- Wirkung im Prinzip wie Primärschlüssel:
  - Stellt sicher, dass das angegebene Attribut bzw. die Attributkombination nur eindeutige Werte enthält.
  - Die „UNIQUE-Attribute“ können ebenfalls in REFERENCES Klauseln referenziert werden.
- UNIQUE-Klausel kann im Gegensatz zum Primärschlüssel in einer Relation mehrfach auftreten.
- Deklaration analog zum Primärschlüssel:

```
CREATE TABLE Kurs
( KursNr CHAR(3) NOT NULL UNIQUE,
  Titel VARCHAR(20)
);
```

alternativ:

```
CREATE TABLE Kurs
( KursNr CHAR(3) NOT NULL,
  Titel VARCHAR(20),
  UNIQUE(KursNr)
);
```

Übung

- Probiere folgenden SQL-Befehl aus:

```
CREATE TABLE Persons
( P_Id int NOT NULL UNIQUE,
  LastName varchar(255) NOT NULL UNIQUE,
  FirstName varchar(255) NOT NULL UNIQUE,
  Address varchar(255)
);
```

- Was passiert bei mehreren Unique Angaben?
  - Primary key?

Grundsätzliches

- Jede Relation sollte einen (als solchen ausgewiesenen) Primärschlüssel haben
- Referenzen sollten – wann immer sinnvoll – über den Primärschlüssel realisiert werden
- UNIQUE sollte primär zur Erzwingung von Eindeutigkeitsvorgaben verwendet werden.

Unbenamte und benamte Integritätsbedingungen

- Alle bisher dargestellten Integritätsbedingungen (integrity constraints) waren unbenamte Integeritätsbedingungen.
- Gezieltes Löschen bzw. (temporäres) Außerkraftsetzen solcher Integritätsbedingungen in der Regel nicht möglich.

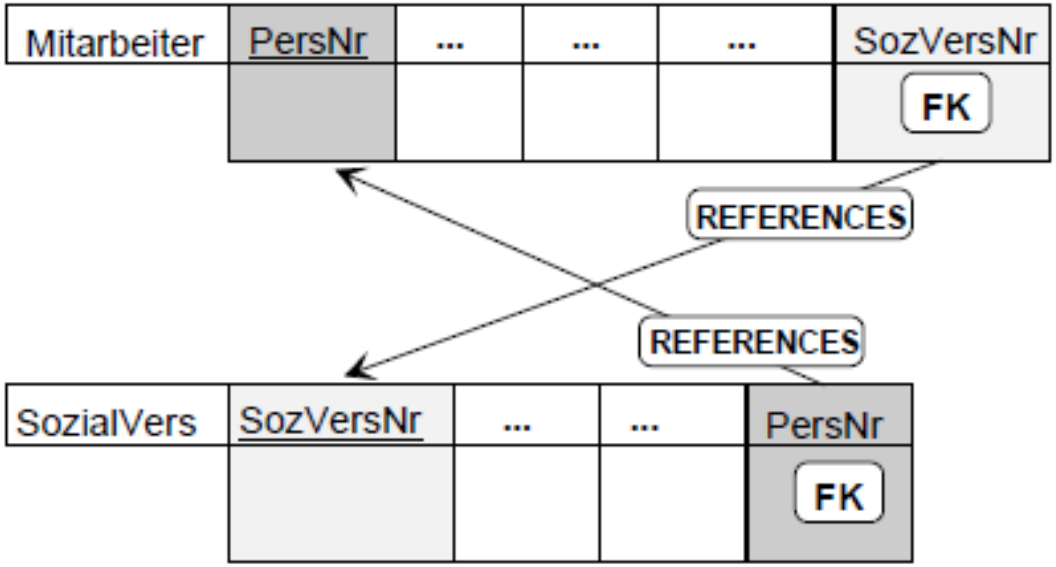
- Abhilfe: Deklaration als benamte Constraints:

```
CREATE TABLE TeileBestPos
( BestNr INTEGER NOT NULL,
  BestPos INTEGER NOT NULL,
  TeileNr INTEGER,
  Menge INTEGER,
  CONSTRAINT pk PRIMARY KEY(BestNr, BestPos),
  CONSTRAINT fk1 FOREIGN KEY(BestNr) REFERENCES TeileBest,
  CONSTRAINT fk2 FOREIGN KEY(TeileNr) REFERENCES Teil
);
```

- Constraint = Zwang, Beschränkung, Gezwungenheit

Aktivieren/Deaktivieren von (benamten) Constraints

- Probleme:
  - Ständige Überprüfung der Integritätsbedingungen bei „Massen-Änderungen“ u.U. ineffizient.
    - Besser wäre in solche Fällen eine Überprüfung auf einmal.
  - Wechselseitige Referenzierung:



Aktivieren/Deaktivieren von (benamten) Constraints

Lösungsmöglichkeit:

- Gezieltes Ein-/Ausschalten innerhalb einer Transaktion
  - Deferred Constraint Checking

```
SET CONSTRAINTS ALL|<constraint-name(n)> DEFERRED;
SQL-Anweisung 1;
SQL-Anweisung 2;
...
SQL-Anweisung n;
SET CONSTRAINTS ALL|<constraint-name(n)> IMMEDIATE;
```

  - Mehrere constraint-namen werden durch Beistriche getrennt

- Wirkungsweise:
  - Integritäts-Prüfungen werden temporär ausgesetzt und erst nach SET CONSTRAINTS ... IMMEDIATE durchgeführt.
  - Folgt auf „DEFERRED“ keine „IMMEDIATE“-Klausel, so werden die Integritäts-Prüfungen automatisch bei COMMIT WORK durchgeführt (später – Transaktionen)

Relationen löschen

- Löschen einer Relation

```
DROP TABLE <relation>;
```

- Wirkung:
  - Löscht die Tabelle Kurs samt Inhalt und entfernt den Katalogeintrag mit allen Verweisen und zugehörenden Hilfsdaten (z.B. Indexen)

Übung

- Erzeuge die entsprechenden Create Statements für das unten angegebene Datenbankschema des Olympia Informationssystems. Achte dabei bei der Erstellung auf eine sinnvolle Wahl der Datentypen. Versuche weiters möglichst restriktive Integritätsbedingungen festzulegen.

Bewerb(Bewerbsid, Sportart, Bezeichnung, Geschlecht, Datum)
NimmtTeil(Bewerbsid, SportlerId, Startnr., ErgebnisId)
MannschaftNimmtTeil(Bewerbsid, ManschaftsId, Startnr., ErgebnisId)
Sportler(SportlerId, Vorname, Nachname, Nationalität, Geschlecht, Geburtsdatum)
Mannschaft(ManschaftsId, Nationalität, Mannschaftsbezeichnung)
TeilVon(ManschaftsId, SportlerId)
Ergebnis(ErgebnisId, Platzierung)

- **DML**

Der Datenmanipulationsteil von SQL

- Wir brauchen noch Befehle zur Manipulation
  - Einfüge-, Aktualisierung- und Löschoperationen

- INSERT
  - Hinzufügen neuer Zeilen zur Tabelle
  - Datenquelle:
    - mit VALUES direkt angeben
    - Oder die Werte stammen aus dem Ergebnis einer SELECT-Abfrage (später)

- UPDATE
  - Vorhandene Zeilen aktualisieren
  - Die zu verändernden Zeilen (Rowset) kann sowohl mit WHERE-Klauseln als auch über eine Verknüpfung mit anderen Tabellen ermittelt werden, ebenso können die neuen Werte aus anderen Tabellen stammen.

- DELETE
  - Entfernt Zeilen
  - Analog zu Update können die zu löschenden Zeilen durch WHERE eingeschränkt oder per JOIN-Verknüpfung über den Umweg anderer Tabellen spezifiziert werden.

Der Insert Befehl

- Einfügen neuer Datensätze in bestehende Tabellen laufender Applikationen
- Übernahme kompletter Datenbestände aus existierenden Datenquellen
  - Beispiel: Wiederherstellungsprozesses nach einem Systemausfall mit Datenverlust

- Die allgemeine Syntax des INSERT-Ausdruckes lautet:

```
INSERT INTO <Tabellenname>(<Spaltenname> [, weitere
Spaltennamen]) VALUES (<Wert für die erste Spalte> [,
weitere      Werte])
```

  - Einfache Form: Hinter VALUES werden alle einzufügenden Werte der Reihe nach aufgelistet.

- Es wird genau eine neue Zeile erzeugt

Beispiel 1

```
CREATE TABLE Customers
(id INT NOT NULL,
 name VARCHAR (20) NOT NULL,
 age INT NOT NULL,
 address CHAR (25),
 salary DECIMAL (18,2),
 PRIMARY KEY (id)
);

INSERT INTO Customers (id, name, age, address, salary)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

INSERT INTO Customers (id, name, age, address, salary) VALUES
(2, 'Khilan', 25, 'Delhi', 1500.00);

INSERT INTO Customers (id, name, age, address, salary) VALUES
(3, 'kaushik', 23, 'Kota', 2000.00);
```

Übung

- Probiere Beispiel 1 aus.
- Füge in eine Tabelle ein, indem du die Attributnamen weglässt und den Befehl „INSERT INTO <relation> VALUES” verwendest.
- Teste den Insert-Befehl in einer Tabelle mit einem Auto-Increment Primärschlüssel.

- Hinweis: Den Tabelleninhalt kannst du mit

  SELECT * FROM <relation>;

  ausgeben lassen (mehr dazu später).

Hinweise zu INSERT

- Alle nichtnumerischen Werte müssen in einfache oder doppelte Hochkommata eingeschlossen werden.
  - Hierunter fallen neben den Zeichenkettentypen (VARCHAR[], TEXT, CHAR[]…) auch alle Datumstypen (DATE, TIME…).

- Sonderstellung: NULL
  - Repräsentiert explizit fehlende Werte
  - Zur Abgrenzung von der Zeichenkette *NULL* nicht in Anführungszeichen eingeschlossen!

Übung

- Befülle mit dem INSERT-Befehl nur einige Attribute eines Datensatzes. Was passiert mit den restlichen Attributen?

- Füge in die Tabelle „CUSTOMERS“ folgende Werte in folgender Reihenfolge ein:

```
name = 'Chaitali'
id = 4
address = 'Mumbai'
age = 25
salary = 6500.00
```

- Wie lautet der entsprechende SQL-Befehl?
- Was passiert, wenn der Name weggelassen wird?

Bemerkung

- Auf dieser Ebene des Zugriffs gibt es keinen ‚Rückgängig-Button‘!

- Eine Datenbank stellt einen elementaren Datenspeicher dar, INSERT/UPDATE/DELETE verändert diesen Speicher direkt.
- Was eingefügt, geändert oder gelöscht wurde, ist unwiderruflich eingefügt, geändert und gelöscht.

Datensätze löschen

- Zur Löschung von Tupeln aus einer Tabelle existiert der DELETE-Befehl, der die betroffenen Datensätze ohne weite Nachfrage entfernt.

```
DELETE FROM <relation> [WHERE search_condition];
```

Beispiele

- Die einfachste Ausprägung der DELETE-Anweisung löscht alle Tupel einer Tabelle:

```
DELETE FROM <relation>;
```

- Durch Angabe der WHERE-Klausel können die zu löschenden Tupel eingegrenzt werden.

```
DELETE FROM Customers WHERE id = 6;
```

Der Update-Befehl

- UPDATE Befehl
  - gestattet es, frei wählbare Mengen von Tupel einer Tabelle zu modifizieren

- Die allgemeine Syntax des Befehls lautet:

```
UPDATE <relation> SET <col_name>=<expression>[,
...] [WHERE search_condition];
```

Beispiele

- `UPDATE Employee SET birthday=NULL;`
  - Setzt alle Geburtsdaten in der Tabelle Employee auf NULL.
- `UPDATE Customers SET address = 'Pune' WHERE id = 6;`
- `UPDATE Customers SET address = 'Pune', salary = 1000.00;`
- `UPDATE Customers SET salary = salary*1.1;`
  - Auch neue Inhalte aus den Bisherigen können errechnet werden: Das Gehalt aller Mitarbeiter wird um zehn Prozent erhöht.

Übung

- Gegeben sei folgendes Datenbankschema:

  Leiter(leiterEmail, leiterName)

  Kurs(kursNr, titel)

  Leitet(leiterEmail, kursNr)

- Erzeuge ein entsprechendes CREATE TABLE-Statement für die Tabellen.
- Beim Entfernen eines Kurses oder Leiters sollen die entsprechenden Einträge in der Tabelle Leitet gelöscht werden.
- Probiere aus, ob alles korrekt funktioniert.
  - Einfügen, Löschen, etc.

Übung

- Informiere dich im Internet darüber, welche Möglichkeiten man hat, um mehrere SQL-Befehle in einer Textdatei abzuspeichern und dann auf einmal abarbeiten zu lassen!

Übung

- Auf dem Klassenordner findest du eine Datei mit 500 Beispieldatensätzen in einer CSV-Datei. Schreibe ein Linux-Konsolen-Script, welches in einer Datenbank eine entsprechende Relation erstellt, wenn sie noch existiert und dann alle Datensätze darin einfügt.
- Informiere dich im Internet!

- Weiterführende DDL - Befehle

Ändern einer Tabelle

- Die Struktur einer Tabelle wird wie folgt geändert:

```
ALTER TABLE <Aufgabe> <Zusatzangaben>;
```

- Mit der Aufgabe ADD CONSTRAINT wird eine Constraint hinzugefügt:

```
ALTER TABLE Dienstwagen
  ADD CONSTRAINT Dienstwagen_PK PRIMARY KEY (ID);

ALTER TABLE Mitarbeiter ADD CONSTRAINT
Mitarbeiter_PersNr UNIQUE (Personalnummer);
```

Beispiele

```
ALTER TABLE Teilnehmer ADD PLZ DECIMAL(4) ADD Strasse
CHAR(30);
```

- *„Es existiere eine noch leere Tabelle Teile. Eine zusätzliche Spalte Preis soll hinzugefügt werden, die zwingend Werte enthalten soll.“*

```
ALTER TABLE Teile ADD Preis DECIMAL(8,2) NOT NULL
```

Ändern einer Tabelle

- Neues Attribut definieren:

  ```
  ALTER TABLE <relation> ADD <attribut> datentyp;
  ```

- Mit `ALTER ... DROP` können Attribute auch wieder gelöscht werden.

- Nachträgliche Änderungen von Attribut-Definitionen (Typ, „Größe“, NULL-Klausel) bei leerer Relation oder NULL-wertiger Spalte durch

  ```
  ALTER TABLE ... MODIFY
  ```

- Attribute umbenennen:

  ALTER TABLE <relation> CHANGE <alter_attributname> <neuer_attributname> datentyp;

- Und viele mehr…
  - Lohnt es sich vielleicht nicht doch, die Relation neu anzulegen?

Manipulation von Tabellen

- Manche DBMS erlauben das Ändern des Relationsnamens mittels der Anweisung:

```
RENAME TABLE <alter_Name> TO <neuer_Name>;
```

  - Alternativ:

```
ALTER TABLE <alter_Name> RENAME <neuer_Name>;
```

  - Funktioniert auch zum Verschieben in eine andere Datenbank:

```
RENAME TABLE <current_db.tbl_name> TO
<other_db.tbl_name>;
```

Anmerkungen

- Neue Spalten werden stets hinten angehängt.

- Sofern die Relation bereits Tupel enthält, werden die neuen Attribute mit Nullwerten gefüllt.
  - Was passiert mit den bestehenden Datensätzen, wenn die NOT NULL Klausel für neue Attribute angegeben wird?

- DRL – DATA RETRIEVAL LANGUAGE

Hintergrund: Relationen-Algebra und -Kalkül

- Eigenschaften
  - Deskriptiv
    - Im Gegensatz zu prozedural: man beschreibt was man haben möchte, nicht wie man dazu kommt (auf welchem Weg)
  - Mengenorientiert
    - Abgeschlossen
      - Operationen auf Relationen liefern immer wieder Relationen
    - orthogonal
      - Operationen lassen sich beliebig kombinieren
  - Mathematische Fundierung
    - Relationen-Algebra
      - 5 Basisoperatoren: Vereinigung, Differenz, Produkt, Selektion, Projektion
    - Relationen-Kalkül

- RELATIONEN-ALGEBRA

Operatoren der Relationen-Algebra

- Selektion
  - σ_F R
    - steht für eine Tupelauswahl angewandt auf Relation R unter Anwendung der Selektionsformel F
  - Beispiel

Teilnehmer	TnNr	Name	Ort
	143	Schmidt, M.	Bremen
	145	Huber, Chr.	Augsburg
	146	Abele, I.	Senden
	149	Kircher, B.	Bochum
	155	Meier, W.	Stuttgart
	171	Möller, H.	Ulm
	173	Schulze, B.	Stuttgart
	177	Mons, F.	Essen
	185	Meier, K.	Heidelberg
	187	Karstens, L.	Hamburg
	194	Gerstner, M.	Ulm

Anfrage: $\sigma_{TnNr > 155}$ Teilnehmer:

Ergebnis-Rel	TnNr	Name	Ort
	171	Möller, H.	Ulm
	173	Schulze, B.	Stuttgart
	177	Mons, F.	Essen
	185	Meier, K.	Heidelberg
	187	Karstens, L.	Hamburg
	194	Gerstner, M.	Ulm

Selektion

- Selektionsformel F:
  - Attributwerte
  - Konstanten
  - Vergleiche (<, >, >=, >=, <>)
  - logische Verknüpfungen (∨, ∧, ¬)

Anfrage: $\sigma_{(TnNr <= 146) \vee (Ort = 'Ulm')}$ Teilnehmer

Ergebnis-Rel	TnNr	Name	Ort
	143	Schmidt, M.	Bremen
	145	Huber, Chr.	Augsburg
	146	Abele, I.	Senden
	171	Möller, H.	Ulm
	194	Gerstner, M.	Ulm

Selektion

Anfrage: $\sigma_{(TnNr < 180) \wedge (Ort = 'Ulm')}$ Teilnehmer :

Ergebnis-Rel	TnNr	Name	Ort
	171	Möller, H.	Ulm

- Formal:
  - $sch(\sigma_F R) = sch(R)$
  - $val(\sigma_F R) = \{ t \in val(R) \mid F(t) \}$

Operatoren der Relationen-Algebra

- Projektion
  - π_{Attr} R
    - steht für eine Projektion angewandt auf Relation R, wobei Attr die Teilmenge der Attribute ist, auf die R abgebildet (projiziert) wird.
  - Beispiel

Anfrage: $\pi_{\{TnNr,Name\}}$ Teilnehmer :

Teilnehmer	TnNr	Name	Ort
	143	Schmidt, M.	Bremen
	145	Huber, Chr.	Augsburg
	146	Abele, I.	Senden
	149	Kircher, B.	Bochum
	155	Meier, W.	Stuttgart
	171	Möller, H.	Ulm
	173	Schulze, B.	Stuttgart
	177	Mons, F.	Essen
	185	Meier, K.	Heidelberg
	187	Karstens, L.	Hamburg
	194	Gerstner, M.	Ulm

Ergebnis-Rel	TnNr	Name
	143	Schmidt, M.
	145	Huber, Chr.
	146	Abele, I.
	149	Kircher, B.
	155	Meier, W.
	171	Möller, H.
	173	Schulze, B.
	177	Mons, F.
	185	Meier, K.
	187	Karstens, L.
	194	Gerstner, M.

Projektion

Anfrage: $\pi_{\{Ort\}}$ Teilnehmer :

Ergebnis-Rel	Ort
	Bremen
	Augsburg
	Senden
	Bochum
	Stuttgart
	Ulm
	Essen
	Heidelberg
	Hamburg

- Formal:
  - $sch(\pi_L R) = L \qquad L \subseteq sch(R)$
  - $val(\pi_L R) = \{ t(L) \mid \exists\, t' \in val(R) \wedge t = t'(L) \}$

Zusammengesetzte Anfragen

- Beispiel: Zusammengesetzte Anfrage
  - *„Gib (nur) Teilnehmer-Nummer und -Name aller Kurs-Teilnehmer aus Ulm aus“*
    - Anfrage: $\pi_{\{TnNr,Name\}}\ \sigma_{Ort = 'Ulm'}$ Teilnehmer
    - (logische !) Abarbeitungsreihenfolge:
      - 1. Schritt: Berechnung des σ-Ausdrucks ⇒ Rel1
      - 2. Schritt: Anwendung des π-Operators auf Rel1 ⇒ Erg
    - Die alternative Formulierung
      - $\sigma_{Ort = 'Ulm'}\ \pi_{\{TnNr,Name\}}$Teilnehmer
    - wäre nicht korrekt - warum?

Attribut-Umbenennung

- Es ist sinnvoll, den Projektions-Operator mit der Möglichkeit zur Attribut-Umbenennung zu versehen.
- Notation:
  - $\pi_{NeuerName:alterName}\ R$
- Beispiel
  - „Gib die Teilnehmer-Relation aus, aber benenne ‘Name’ in ‘Nachname’ um“:
    - $\pi_{\{TnNr,Nachname:Name\}}$ Teilnehmer
    - oder falls „Name“ nicht eindeutig wäre:
      - $\pi_{\{TnNr,Nachname:Teilnehmer.Name\}}$ Teilnehmer

Operatoren der Relationen-Algebra

- Vereinigung
  - R U S
    - steht für die Vereinigung (union) der Relationen R und S
    - R und S müssen strukturgleich sein
      - Es muss gelten: sch(R) = sch(S)
  - Kurzschreibweise: $U_{i=1,2,..k}\ R_i = R_1\ U\ R_2\ U\ ...\ U\ R_k$
  - Formal:
    - sch(R ∪ S) = sch(R) = sch(S)
    - val(R ∪ S) = val(R) ∪ val(S)

Operatoren der Relationen-Algebra

- Differenz
  - R – S
    - steht für die Differenz der Relationen R und S
    - alle Tupel von R, die auch in S vorkommen, werden aus R entfernt.
    - Auch hier muss gelten: sch(R) = sch(S)
  - Formal:
    - sch(R - S) = sch(R) = sch(S)
    - val(R – S) = val(R) – val(S)

Operatoren der Relationen-Algebra

- Kartesisches Produkt
  - R × S
    - steht für das kartesische (relationale) Produkt der Relationen R und S
  - Beispiel

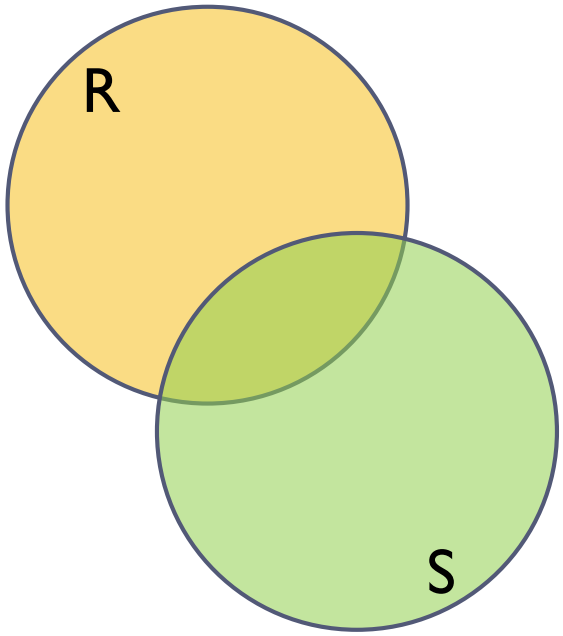
R	A	B
	a_1	b_1
	a_2	b_2

S	C	D
	c_1	d_1
	c_2	d_2
	c_3	d_3

R × S	A	B	C	D
	a_1	b_1	c_1	d_1
	a_1	b_1	c_2	d_2
	a_1	b_1	c_3	d_3
	a_2	b_2	c_1	d_1
	a_2	b_2	c_2	d_2
	a_2	b_2	c_3	d_3

Operatoren der Relationen-Algebra

- Durchschnitt:
  - $R \cap S = R - (R - S)$
    - ableitbar!



Beispiel

- Kombination von × und σ
  - gleichzeitig Beispiel für die Verknüpfung von Tabellen

- Anfrage:
  - „Welche Kurse (Ausgabe: Kurs-Nummer und -Titel) sind für welche anderen Kurse (Ausgabe: Kurs-Nummer) Voraussetzung?“
- Ausgangs-Relationen:

Kurs	KursNr	Titel
	G08	Grundlagen I
	G10	Grundlagen II
	P13	C-Programmierung
	I09	Datenbanken

Vorauss	VorNr	KursNr
	G08	P13
	G10	P13
	G08	I09
	G10	I09
	P13	I09

Kurs	KursNr	Titel
	G08	Grundlagen I
	G10	Grundlagen II
	P13	C-Programmierung
	I09	Datenbanken

Vorauss	VorNr	KursNr
	G08	P13
	G10	P13
	G08	I09
	G10	I09
	P13	I09

- Anfrage in Relationen-Algebra:

$$\pi_{\text{VorNr,Titel,VorausFür:Vorauss.KursNr}}\ \sigma_{\text{VorNr = Kurs.KursNr}}\ (\text{Vorauss} \times \text{Kurs})$$

$\pi_{VorNr,Titel,VorausFür:Vorauss.KursNr}\ \sigma_{VorNr = Kurs.KursNr}\ (Vorauss \times Kurs)$

- Abarbeitungsreihenfolge:
  - 1. Schritt: Berechnung Vorauss × Kurs → Rel1

Rel_1	VorNr	Vorauss.KursNr	Kurs.KursNr	Titel
	G08	P13	G08	Grundlagen I
	G08	P13	G10	Grundlagen II
	...	...	...	...
	G08	P13	I09	Datenbanken
	G10	P13	G08	Grundlagen I
	...	...	...	...
	G10	P13	I09	Datenbanken
	...	...	...	...
	P13	I09	I09	Datenbanken

$\pi_{VorNr,Titel,VorausFür:Vorauss.KursNr}\ \sigma_{VorNr = Kurs.KursNr}\ (Vorauss \times Kurs)$

- 2. Schritt: Anwendung von $\sigma_{VorNr=Kurs.KursNr}$ auf Rel1

Erg-Rel	VorNr	Vorauss.KursNr	Kurs.KursNr	Titel
	G08	P13	G08	Grundlagen I
	G10	P13	G10	Grundlagen II
	G08	I09	G08	Grundlagen I
	G10	I09	G10	Grundlagen I
	P13	I09	P13	C-Programmierung

$\pi_{VorNr,Titel,VorausFür:Vorauss.KursNr}\ \sigma_{VorNr = Kurs.KursNr}\ (Vorauss \times Kurs)$

- Mögliche Umbenennung von Attributen + Änderung der Attribut-Reihenfolge mittels (erweitertem) π-Operator:
  - $\pi_{\{VorNr,\ Titel,\ Vorauss\_für:\ Vorauss.KursNr\}}$ Erg-Rel
- bzw.
  - $\pi_{\{VorNr,\ Titel,\ Vorauss\_für:\ Vorauss.KursNr\}}\ \sigma_{VorNr = Kurs.KursNr}\ (Vorauss \times Kurs)$

Erg-Rel	VorNr	Titel	Vorauss_für
	G08	Grundlagen I	P13
	...	...	...
	P13	C-Programmierung	I09

Spezielle Operationen

- Verknüpfung von Tabellen aufgrund von Attributwert-Beziehungen häufige Operation im relationalen Datenmodell
  - daher durch spezielle Verbund-Operation explizit unterstützt

- Verbund (Join)
  - $R \bowtie_F S$
    - steht für den Verbund der Relationen R und S unter Verwendung der Verbund-Bedingung F
- $R \bowtie_F S$ ist semantisch äquivalent zu $\sigma_F (R \times S)$

Verbund - Join

Kurs	KursNr	Titel
	G08	Grundlagen I
	G10	Grundlagen II
	P13	C-Programmierung
	I09	Datenbanken

Vorauss	VorNr	KursNr
	G08	P13
	G10	P13
	G08	I09
	G10	I09
	P13	I09

- Beispiel
  - $\sigma_{\text{VorNr = Kurs.KursNr}}$ (Vorauss × Kurs)
    - führt zum selben Resultat wie:
  - Vorauss $\bowtie_{\text{VorNr = KursNr}}$ Kurs

- Ausführungslogik:

```
for each Tupel x in Vorauss do
   for each Tupel y in Kurs do
      if x.VorNr = y.KursNr then
          erzeuge ResultatTupel;
      fi
   done
done
```

Beispiel: Verbund - Join

- Anfrage:
  - „Gib (nur) KursNr und Titel aller Kurse sowie den Namen des Kursleiters aus, die von dem Kursleiter mit der Personalnummer 27183 durchgeführt werden“

Kursleiter	PersNr	Name	Gehalt
	27183	Meier, I.	4300.50
	29594	Schulze, H.	3890.20
	38197	Huber, L.	4200.10
	43325	Müller, K.	3400.80

Kurs	KursNr	Titel
	G08	Grundlagen I
	G10	Grundlagen II
	P13	C-Programmierung
	I09	Datenbanken

Fuehrt_durch	AngNr	KursNr	PersNr
	1	G08	38197
	2	G08	38197
	1	G10	43325
	2	G10	29594
	1	P13	27183
	2	P13	27183
	1	I09	29594
	2	I09	29594
	2	I09	29594

Beispiel: Verbund - Join

- Mögliche Anfrageformulierung:

$\pi_{\{Kurs.KursNr,\ Titel,\ Name\}}\ \sigma_{PersNr\ =\ 27183}$
(Kurs $\bowtie_{Kurs.KursNr=Fuehrt\_durch.KursNr}$
(Kursleiter $\bowtie_{Kursleiter.PersNr=Fuehrt\_durch.PersNr}$ Fuehrt_durch)
)

- Anmerkung:
  - Die „Join-Bedingung“ kann auch mittels <, >, >=, >= sowie V, $\wedge$,¬ formuliert werden.

Spezielle Joins

- Equi-Join, Theta-Join
  - Joins über eine “=“-Bedingung heißen: Equijoin
  - Joins über “<,>,≤,≥“ etc. heißen: θ-Join (Theta-Join)

- Natural-Join:
  - Joins werden oft zwischen Attributen gleichen Namens formuliert.
  - Joins mit "="-Bedingung sind am häufigsten.
  - Hierauf abgestellte Join-Variante: **Natural Join**

Natural Join

- Natural Join
  - $R \bowtie S$
    - steht für natürlichen Verbund (natural join) der Relationen R und S.
  - Alle in R und S auftretenden Attribute gleichen Namens werden mittels "="-Bedingung verknüpft.
  - Haben R und S keine gemeinsamen Attribute, so hat $R \bowtie S$ die gleiche Wirkung wie $R \times S$.

- Beispiel
  - Vorhin:
    - $\pi_{\{Kurs.KursNr, Titel, Name\}} \sigma_{PersNr = 27183} (Kurs \bowtie_{Kurs.KursNr=Fuehrt\_durch.KursNr} (Kursleiter \bowtie_{Kursleiter.PersNr=Fuehrt\_durch.PersNr} Fuehrt\_durch))$
  - Jetzt:
    - $\pi_{\{KursNr, Titel\}} Kurs \bowtie (\sigma_{PersNr = 27183} (Kursleiter) \bowtie Fuehrt\_durch)$

- Unterschied zum Equi-Join:
  - Attribute gleichen Namens treten in der Ergebnis-Relation (sowie in allen Zwischenergebnis-Relationen) natürlich jeweils nur einmal auf.
    - $sch(R \bowtie S) = sch(R) \cup sch(S)$

Beispiele

Gegeben:

R	A	B	C
	1	a	d
	3	c	c
	4	d	f
	5	d	b
	6	e	f

S	B	D
	a	100
	b	300
	c	400
	d	200
	e	150

T	B	D
	a	100
	d	200
	f	400
	g	120

Anfragen und Ergebnisse:

$\sigma_{D<300}$ S	B	D
	a	100
	d	200
	e	150

$\pi_{\{A,C\}}$ R	A	C
	1	d
	3	c
	4	f
	5	b
	6	f

$\pi_{\{C\}}$ R	C
	d
	c
	f
	b

Gegeben:

R	A	B	C
	1	a	d
	3	c	c
	4	d	f
	5	d	b
	6	e	f

S	B	D
	a	100
	b	300
	c	400
	d	200
	e	150

T	B	D
	a	100
	d	200
	f	400
	g	120

Anfragen und Ergebnisse:

S ∪ T	B	D
	a	100
	b	300
	c	400
	d	200
	e	150
	f	400
	g	120

S – T	B	D
	b	300
	c	400
	e	150

T – S	B	D
	f	400
	g	120

$R \bowtie_{R.B=S.B} S$	A	R.B	C	S.B	D
	1	a	d	a	100
	3	c	c	c	400
	4	d	f	d	200
	5	d	b	d	200
	6	e	f	e	150

R × T	A	R.B	C	T.B	D
	1	a	d	a	100
	1	a	d	d	200
	1	a	d	f	400
	1	a	d	g	120
	3	c	c	a	100
	3	c	c	d	200
	3	c	c	f	400
	3	c	c	g	120
	4	d	f	a	100
	4	d	f	d	200
	4	d	f	f	400
	4	d	f	g	120
	5	d	b	a	100
	5	d	b	d	200
	5	d	b	f	400
	5	d	b	g	120
	6	e	f	a	100
	6	e	f	d	200
	6	e	f	f	400
	6	e	f	g	120

S ⋈ T	B	D
	a	100
	d	200

R ⋈ T	A	B	C	D
	1	a	d	100
	4	d	f	200
	5	d	b	200

$R \bowtie_{A*100=D} S$	A	R.B	C	S.B	D
	1	a	d	a	100
	4	d	f	c	400
	3	c	c	b	300

Übung (1)

- Gegeben ist folgendes Beispielschema:
  - STUDENT(MatrikelNr,Name,Wohnort)
  - MITARBEITER(BearbeiterNr,PersonalNr,Name,Wohnort)
  - VORLESUNG(VorlesungsNr,Bezeichnung)
  - DOZENT(BearbeiterNr,PersonalNr,VorlesungsNr)
  - HOERT(MatrikelNr,VorlesungsNr,Wiederholung)
  - FINDETSTATT(VorlesungsNr,Zeit,RaumNr)
  - RAUM(RaumNr,Bezeichnung)

Übung (2)

- Erstelle für das Schema die folgenden Anfragen mit Hilfe der relationalen Algebra:
  - Gib die Bezeichnung der Vorlesung ’080104’ aus.
  - Gib die Namen aller Studenten, die die Veranstaltung ’080104’ hören, zusammen mit den Namen aller Dozenten der Veranstaltung ’080104’ aus.
  - Angenommen, der Name identifiziert eine Person eindeutig. Gib die Personen, die Dozent einer Veranstaltung sind und sich parallel dazu für diese Veranstaltung als Student angemeldet haben aus.

Übung

- Gegeben sei das folgende Relationenschema eines sozialen Netzwerkes:
  - Profil(Email, Passwort, Vorname, Nachname)
  - Schulabschluss(Email, Schule, Jahrgang)
  - IstBefreundet(EmailDesAnfragenden, EmailDesAngefragten, AnfrageDatum, Annahme-
  - Datum)
  - Album(AlbumId, Email, Bezeichnung, ErzeugungsDatum)
  - Bild(BildID, Beschreibung, AlbumId)
  - Markierung(BildID, Email, PositionX, PositionY, DimensionX, DimensionY)

- Erstelle für dieses Schema die folgenden Anfragen mit Hilfe der relationalen Algebra
  a) Finde die Email-Adresse aller Benutzer, die mit dir zusammen die Schule absolviert haben.
  b) Erstelle eine Abfrage, welche alle Bilder mit Album-Bezeichnung und Bild-Beschreibung, auf denen du markiert wurdest, auflistet.
  c) Um dem Benutzer “Chris Freeman“ neue Freunde vorzuschlagen, sollen ihm die Profile von Personen, mit welchen er zusammen auf Bildern markiert wurde und mit welchen er noch nicht befreundet ist, als mögliche Freunde vorgeschlagen werden.
  d) Gesucht sind alle Profile, die innerhalb der letzten 5 Monaten keine Freundschaftsanfragen angenommen haben.

Übung

- Gegeben sei das folgende Schema:
  - Mitarbeiter(MitNr, Vorname, Nachname, GebDatum)
  - Standort(PLZ, Ort, GeschäftsführerID)
    - GeschäftsführerID ist ein Fremdschlüssel auf MitNr

- Gesucht sind alle Mitarbeiter (Vorname, Nachname, Geburtsdatum), die an dem selben Standort angestellt sind, an dem Wolfgang Maier Geschäftsführer ist.

Übung

- Gegeben sei folgendes Datenbankschema, in dem Informationen über Orte, Filme und das aktuelle Programm gespeichert sind:
  - Kinos(KinoName, Adresse, Stadt, PLZ, Telefon)
  - Schauspieler(SchauspID, Vorname, Nachname)
  - Filme(FilmID, Titel, Regie)
  - SpieltIn(FilmID, SchauspID)
  - Programm(KinoName, FilmID, Datum, Zeit)

- In welchen Filmen (Titel) spielt Philip Seymour Hoffman mit?
- Welche Schauspieler (Vorname, Nachname) haben schon mal mit Scarlett Johansson gearbeitet?
- Welche Schauspieler (Vorname, Nachname) spielen in Filmen, die aktuell laufen?
- In welchen Kinos (Name und Adresse und Stadt) läuft Slumdog Millionaire?
- In welchen Kinos (Name, Adresse und Stadt) laufen Filme mit Natalie Portman?
- Welche Filme laufen nicht mehr aktuell im Kino?

Übung

- Gegeben sei das folgende Schema einer Personalverwaltungsdatenbank.
  - Person(PersonalId, Vorname, Nachname, aktuellesGehalt)
  - Gehaltserhöhung(PersonalId, Datum, altesGehalt, neuesGehalt)
- Gesucht ist eine Anfrage, welche Vorname, Nachname und Gehalt aller Mitarbeiter zurückliefert, die noch keine Gehaltserhöhung erhalten haben.

Übung

- Es seien folgende Relationen in Tabellenform gegeben:
  - Tabelle 1: Mitarbeiter

Name	Wohnort
Max Müller	Kiel
Tina Schmidt	Lübeck
Klaus Meyer	Kiel

  - Tabelle 2: Studenten

Name	Wohnort
Max Müller	Kiel
Andre Petersen	Hamburg
Thomas Ebert	Berlin
Klaus Meyer	Lübeck

- Bilde den Durchschnitt!
- Bilde die Vereinigung!
- Bestimme alle Mitarbeiter, die keine Studenten sind!
- Welche Operationen sind notwendig, um alle Mitarbeiter zu erhalten, die in Kiel wohnen?
- Welche Operationen sind notwendig, um eine Liste aller Namen der Mitarbeiter oder Studenten zu erhalten?

Übung (1)

- Betrachte die folgende Datenbasis, die ein Prüfungsverwaltungssystem beschreibt:
  - Studierende haben eine Matrikel-Nummer und einen Namen.
  - Semester werden mit 21 = Sommersemester 2002, 22 = Wintersemester 2002/03, 31 = Sommersemester 2003 bezeichnet. (Auf diese Weise: zeitlichen Abfolge (41 < 42 < 51, Sommersemester 2004 früher als Wintersemester 2004/05, früher als Sommersemester 2005).

Stud

MatNo	Name
0900	Werner Brösel
4242	Karl Napf
:	:

Doz

Name	Titel
Hogrefe	Prof
Dix	Prof
Richter	Prof
May	Prof
Ebner	Dr
Behrends	DiplInf
:	:

LehrVeranst (LV)

Name	ECTS
Informatik I	9
Informatik II	9
Informatik II	9
Programmierprakt	9
Datenbanken	6
SQL-Prakt	9
:	:

Prf

MatNo	LV	Semester	Note
0900	Informatik I	22	4.0
0900	Informatik II	31	3.3
0900	Programmierprakt	31	4.0
0900	Datenbanken	32	5.0
0900	Datenbanken	42	4.0
0900	SQL-Prakt	51	2.3
:	:	:	:

VorlVerz (VV)

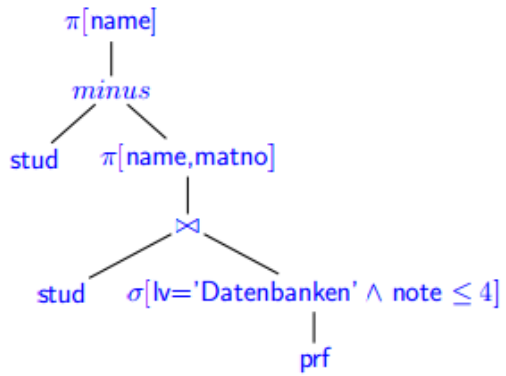
LV	Dozent	Semester
Informatik I	May	22
Informatik I	May	32
Informatik II	Fu	31
Informatik II	Ebner	41
Programmierprakt	Werner	41
Datenbanken	May	42
Datenbanken	May	52
SQL-Prakt	Behrends	51
:	:	:

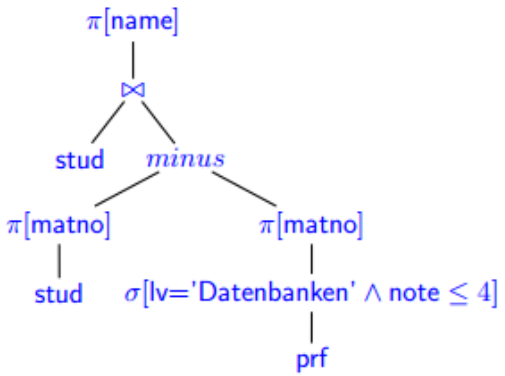
setztVoraus

SetztVoraus	wirdVorausgesetzt
Informatik II	Informatik I
Datenbanken	Informatik II
SQL-Prakt	Datenbanken
SQL-Prakt	Programmierprakt
:	:

Übung (2)

- Gib Sie einen Ausdruck der relationalen Algebra an, der das das folgende tut:
  - Ausgabe aller Paare (Student(Name), Lehrveranstaltung), so dass der Student eine Prüfung über die LV nicht bestanden hat (Noten 1 bis 4.0 sind bestanden, 5.0 ist nicht bestanden).
  - Ausgabe aller Paare (Student(Name), Dozent), so dass der Student eine Prüfung bei diesem Dozenten bestanden hat.
  - Ausgabe aller Namen aller Studierenden, die “Datenbanken” noch nicht erfolgreich absolviert haben.





Äquivalenz-Umformungen

Seien R, S und T drei jeweils geeignet definierte Relationen

- Klassifikation algebraischer Umformungen:
  - ***Kommutativität*** von unären Operationen (σ, π):
    - $\mathbf{U_1}\ \mathbf{U_2}\ R \rightarrow \mathbf{U_2}\ \mathbf{U_1}\ R$
  - ***Kommutativität*** von binären Operationen (U, − , ×,):
    - $R\ \mathbf{B}\ S \rightarrow S\ \mathbf{B}\ R$
  - ***Assoziativität*** von binären Operationen (U, − , ×,):
    - $R\ \mathbf{B}\ (S\ \mathbf{B}\ T) \rightarrow (R\ \mathbf{B}\ S)\ \mathbf{B}\ T$
  - ***Idempotenz*** (bzw. Zusammenfassung) von unären Operationen (σ, π):
    - $\mathbf{U}\ R \rightarrow \mathbf{U}\ \mathbf{U}\ R$
  - ***Distributivität*** von unären Operationen in Bezug auf binäre Operationen:
    - $\mathbf{U}\ (R\ \mathbf{B}\ S) \rightarrow (\mathbf{U}\ R)\ \mathbf{B}\ (\mathbf{U}\ S)$

Äquivalenz-Umformungen von Relationen-Algebra-Ausdrücken

Nr	Regel	Nebenbedingung
1	$\sigma_{F1}(\sigma_{F2} R) \equiv \sigma_{F2}(\sigma_{F1} R)$	—
2	$\sigma_{F}(\pi_{A} R) \equiv \pi_{A}(\sigma_{F} R)$	$\leftarrow$: $Attr(F) \subseteq A$
3	$R \cup S \equiv S \cup R$	—
4	$R \times S \equiv S \times R$	—
5	$R \bowtie_{F} S \equiv S \bowtie_{F} R$	—
6	$(R \cup S) \cup T \equiv R \cup (S \cup T)$	—
7	$(R \times S) \times T \equiv R \times (S \times T)$	—
8	$(R \bowtie_{F1} S) \bowtie_{F2} T \equiv R \bowtie_{F1} (S \bowtie_{F2} T)$	$\rightarrow$: $Attr(F2) \subseteq (Attr(S) \cup Attr(T))$ $\leftarrow$: $Attr(F1) \subseteq (Attr(R) \cup Attr(S))$
9	$\pi_{A1} R \equiv \pi_{A1} \pi_{A2} R$	$A1 \subseteq A2 \subseteq Attr(R)$
10	$\sigma_{F} R \equiv \sigma_{F1} \sigma_{F2} R$	$F = F1 \wedge F2$

Äquivalenz-Umformungen von Relationen-Algebra-Ausdrücken

11	$\sigma_F (R \cup S) \equiv (\sigma_F R) \cup (\sigma_F S)$	—
12	$\sigma_F (R - S) \equiv (\sigma_F R) - (\sigma_F S)$	—
13	$\sigma_F(R \times S) \equiv (\sigma_{F1} R) \times (\sigma_{F2} S)$	$\rightarrow$: (F = F1 $\wedge$ F2) $\wedge$ Attr(F1) $\subseteq$ Attr(R)) $\wedge$ (Attr(F2) $\subseteq$ Attr(S)) $\leftarrow$: F = F1 $\wedge$ F2
14	$\sigma_F(R \bowtie_{F3} S) \equiv (\sigma_{F1} R) \bowtie_{F3} (\sigma_{F2} S)$	$\rightarrow$: (F = F1 $\wedge$ F2) $\wedge$ Attr(F1) $\subseteq$ Attr(R)) $\wedge$ (Attr(F2) $\subseteq$ Attr(S)) $\leftarrow$: F = F1 $\wedge$ F2
15	$\pi_A (R \cup S) \equiv (\pi_A R) \cup (\pi_A S)$	—
16	$\pi_A (R \times S) \equiv (\pi_{A1} R) \times (\pi_{A2} S)$	$\rightarrow$: (A1 = A $\cap$ Attr(R)) $\wedge$ (A2 = A $\cap$ Attr(S)) $\leftarrow$: A = A1 $\cup$ A2

Äquivalenz-Umformungen von Relationen-Algebra-Ausdrücken

17	$\pi_A (R \bowtie_F S) \equiv (\pi_{A1} R) \bowtie_F (\pi_{A2} S)$	$\rightarrow$: $(Attr(F) \subseteq A)$ $\wedge (A1 = A - Attr(S))$ $\wedge (A2 = A - Attr(R))$ $\leftarrow$: $A = A1 \cup A2$
18	$R \bowtie R \equiv R$	—
19	$R \cup R \equiv R$	—
20	$R - R \equiv \emptyset$	—
21	$R \bowtie (\sigma_F R) \equiv \sigma_F R$	—
22	$R \cup \sigma_F R \equiv R$	—
23	$R - \sigma_F R \equiv \sigma_{\neg F} R$	—
24	$(\sigma_{F1} R) \bowtie (\sigma_{F2} R) \equiv \sigma_{(F1 \wedge F2)} R$	—
25	$(\sigma_{F1} R) \cup (\sigma_{F2} R) \equiv \sigma_{(F1 \vee F2)} R$	—
26	$(\sigma_{F1} R) - (\sigma_{F2} R) \equiv \sigma_{(F1 \wedge \neg F2)} R$	—

Beispiele und Erläuterungen zu den Äquivalenz-Umformungen:

- Gegeben seien die Relationen R(A,B,C), R1(A,B,C), R2(A,B,C), S(D,E) und T(F,G,H) wobei die Wertebereiche der Attribute A bis H jeweils Teilmengen der natürlichen Zahlen sein sollen.

- Zu 1: $\sigma_{A<10 \wedge C>8}(\sigma_{B>8} R)$ kann in $\sigma_{B>8}(\sigma_{A<10 \wedge C>8} R)$ transformiert werden und umgekehrt.
- Zu 2: $\sigma_{B<200}(\pi_{\{A,B\}} R)$ kann in $\pi_{\{A,B\}}(\sigma_{B<200} R)$ transformiert werden. Geht man von der rechten Seite aus, d.h. von $\pi_{\{A,B\}}(\sigma_{B<200} R)$, so kann die Transformation nur dann durchgeführt werden, wenn sich die Selektions-Bedingung lediglich auf Attribute bezieht, die auch in der Projektion spezifiert sind (siehe NB zu Regel 2). Im vorliegenden Fall entspricht Attr(F) der Attributmenge {B} und A entspricht der Attributmenge {A,B}. Die Nebenbedingung ist also erfüllt. D.h. die Transformation in die "←"-Richtung ist ebenfalls möglich.
- Zu 8: Gegeben sei $(R \bowtie_{A=D} S) \bowtie_{E=F} T$. Für die "→"-Transformation muss folgende Nebenbedingung erfüllt sein: $\{E,F\} \subseteq (\{D,E\} \cup \{F,G,H\})$. Dies ist hier der Fall. Der Ausdruck kann somit in $R \bowtie_{A=D} (S \bowtie_{E=F} T)$ transformiert werden. Bei $(R \bowtie_{A=D} S) \bowtie_{A=F} T$ ginge dies z.B. hingegen nicht.
- Zu 9: $\pi_{\{A\}} R$ kann in $\pi_{\{A\}} \pi_{\{A,B\}} R$ transformiert werden und umgekehrt.

- Zu 10: $\sigma_{A<100 \wedge B>30}$ R kann zerlegt werden in $\sigma_{A<100}$ $\sigma_{B>30}$ R. Umgekehrt kann man $\sigma_{A<100}$ $\sigma_{B>30}$ R durch Zusammenfassen der Selektionsbedingungen (UND-Verknüpfung) in $\sigma_{A<100 \wedge B>30}$ R transformieren.
- Zu 11: $\sigma_{B<300}$ (R1 ∪ R2) kann in ($\sigma_{B<300}$ R1) ∪ ($\sigma_{B<300}$ R2) transformiert werden und umgekehrt. Anmerkung: Hier ist keine Nebenbedingung erforderlich, da die Vereinigung nur zwischen Relationen gleichen Typs definiert ist.
- Zu 12: $\sigma_{B<300}$ (R1 − R2) kann in ($\sigma_{B<300}$ R1) − ($\sigma_{B<300}$ R2) transformiert werden und umgekehrt. (σ bei R2 kann auch entfallen!)
- Zu 13: Gegeben sei $\sigma_{A<300 \wedge D>50}$ (R × S). Dieser Ausdruck kann in zwei Selektionen überführt werden, indem man die Selektions-Bedingung geeignet „aufspaltet“. In diesem Fall wäre etwa ($\sigma_{A<300}$ R) × ($\sigma_{D>50}$ S) eine mögliche Aufspaltung.
- Zu 14: Da ein Verbund R $\bowtie_F$ S äquivalent zu σ_F (R × S) ist, gilt das im vorangegangenen Beispiel Gesagte – analog übertragen – auch hier.

- Zu 15: Ein Ausdruck der Art $\pi_{\{A,B\}}(R1 \cup R2)$ kann stets „ausmultipliziert“ werden zu $(\pi_{\{A,B\}} R1) \cup (\pi_{\{A,B\}} R2)$.
- Zu 16: Die Nebenbedingung für die "→"-Richtung besagt, dass nach dem „Ausmultiplizieren“ die Projektions-Attribute so gewählt werden müssen, dass sie für die jeweilige Relation auch definiert sind. Im Falle des Ausdruckes $\pi_{\{A,R.B,D\}} (R \times S)$ wäre z.B. $(\pi_{\{A,R.B\}} R) \times (\pi_{\{D\}} S)$ eine korrekte Transformation, $(\pi_{\{A\}} R) \times (\pi_{\{R.B,\ D\}} S)$ hingegen nicht.
- Zu 17: Die Nebenbedingung für die "→"-Richtung kann wie folgt interpretiert werden: Sind die in der Verbund-Bedingung angegebenen Attribute eine Teilmenge der in der Projektion angegebenen Attribute (d.h. gilt $Attr(F) \subseteq A$), dann kann der Ausdruck „ausmultipliziert“ werden, wobei wieder zu beachten ist, dass die in der Projektion auf der linken Seite angegebenen Attribute wieder korrekt auf ihre Relationen „verteilt“ werden. Der Ausdruck $\pi_{\{A,B,E\}}(R \bowtie_{B<E} S)$ kann z.B. transformiert werden in $(\pi_{\{A,B\}} R) \bowtie_{B<E} (\pi_{\{E\}} S)$. $\pi_{\{A,B,D\}} (R \bowtie_{B<E} S)$ hingegen wäre nicht transformierbar, da $\{B,E\} \not\subset \{A,B,D\}$. Die umgekehrte Richtung ("←") ist trivial. Hier sind lediglich die Projektions-Attribute zusammenzufassen.

- Zu 23: $R - \sigma_{B<200} R$ kann in $\sigma_{B \geq 200} R$ transformiert werden und umgekehrt.
- Zu 26: $(\sigma_{A<10} R) - (\sigma_{B>100} R)$ kann in $\sigma_{A<10 \wedge B \leq 100} R$ transformiert werden und umgekehrt.

Weitere (abgeleitete) Operatoren

- Durchschnitt (hatten wir schon)
- R ÷ S Division (andere Schreibweise: R \ S)
  - Sei D = R ÷ S, dann muss gelten:
  - sch(S) ⊂ sch(R)
  - sch(D) = sch(R) - sch(S)
  - t ∈ D ⇔ ∀ s ∈ val(S): < t,s > ∈ val(R)

Beispiel:

R	A	B
	a1	b1
	a2	b1
	a3	b1
	a4	b1
	a1	b2
	a3	b2
	a2	b3
	a3	b3
	a4	b3
	a1	b4
	a2	b4
	a3	b4

S	A
	a1
	a2
	a3

R ÷ S	B
	b1
	b4

Weitere (abgeleitete) Operatoren

- Left Outer Join: (Einseitiger) „Außen-Verbund“
  - Alle Tupel der linken Relation (d.h. alle R-Tupel) sind im Ergebnis enthalten. Gibt es für ein Tupel $r_i \in R$ kein S-Tupel, das F erfüllt, so werden die S-Attribute im Ergebnis-Tupel mit Nullwerten aufgefüllt.

R	A	B
	a1	b1
	a2	b2
	a3	b4
	a4	b2
	a4	b5
	a5	b1

S	B	C
	b1	c1
	b1	c2
	b2	c3
	b3	c6

$R \overset{\subset}{\bowtie}_{R.B=S.B} S$	R.A	R.B	S.B	S.C	
	a1	b1	b1	c1	
	a1	b1	b1	c2	
	a2	b2	b2	c3	
	a3	b4	--	--	⇐ **Nullwert!**
	a4	b2	b2	c3	
	a4	b5	--	--	⇐ **Nullwert!**
	a5	b1	b1	c1	
	a5	b1	b1	c2	

Weitere (abgeleitete) Operatoren

- Analog:
  - Natural Left Outer Join 

  - Natural Right Outer Join 

  - Natural Full Outer Join 

- RELATIONEN-KALKÜL

- Relationen-Algebra
  - „Konstruktion“ der Ergebnisrelation durch sukzessive (geschachtelte) Anwendung von Algebra-Operatoren auf die Ausgangsrelationen (→ prozedurale Vorgehensweise)
- Relationen-Kalkül
  - Ganz andere Philosophie:
    - Beschreibung, welche Bedingungen (Prädikate) die Tupel der Ergebnisrelation erfüllen müssen (→ deklarative Vorgehensweise)

- Beispiel:
  - Gegeben seien die folgenden Relationen:
    - Vorauss(VorNr, KursNr)
    - Kurs(KursNr, Titel)
    - Angebot(AngNr, KursNr, Datum, Ort)
  - Zu beantworten sei folgende Anfrage:
    - „Gib für alle Kurse, die zwischen dem 1.1.96 und 31.3.96 stattfanden und den Kurs G08 als Voraussetzung haben, die KursNr, den Titel, das Datum und den Ort aus.“

- Algebra:
  - $\pi_{\{KursNr, Titel, Datum, Ort\}}$ ((Kurs $\bowtie$ $\sigma_{1.1.96 < Datum < 31.3.96}$ Angebot) $\bowtie$ ($\sigma_{VorNr = 'G08'}$ Vorauss))
- Tupel-Relationenkalkül (prädikative Beschreibung der Ergebnismenge):
  - { t I (∃ a (∃ v (∃ k (Kurs (k) ∧ Angebot (a) ∧ Vorauss (v) ∧ k(KursNr) = a(KursNr) ∧ k(KursNr) = v(KursNr) ∧ a(Datum) < 31.3.96 ∧ a(Datum) > 1.1.96 ∧ v(VorNr) = 'G08' ∧ t(KursNr) = k(KursNr) ∧ t(Titel) = k(Titel) ∧ t(Datum) = a(Datum) ∧ t(Ort) = a(Ort))))) }