SQL 2

Ziele

- Fortgeschrittene SQL-Konstrukte
  - group by
  - having
  - union / intersect / except
- Aggregatfunktionen revisited
- Subqueries
- Korrelierte Subqueries

Group-By-Klausel Syntax

- Syntax:

  GROUP BY grouping-expression {, grouping-expression}

  – Tupel mit gleichen Werten in den Gruppierungsausdrücken werden zu Gruppen zusammengefasst
  – Die SELECT-Klausel referenziert die gruppierten Ausdrücke

FILM

TITEL	GENRE	MIETPREIS
Hängt ihn höher	WESTERN	1,50
Krieg der Sterne	SCI-FI	2,-
1984	SCI-FI	4,-
Django	WESTERN	1,-

GROUP BY GENRE

Hängt ihn höher	WESTERN	1,50
Django	WESTERN	1,-
Krieg der Sterne	SCI-FI	2,-
1984	SCI-FI	4,-

Group-By-Klausel Beispiele

- Durchschnittspreise für jedes Film-Genre

  SELECT genre, AVG(mietpreis) FROM FILM GROUP BY genre

- Anzahl der Filme je Genre

  SELECT genre, count(*) FROM FILM GROUP BY genre
  → *count(*) bezieht sich auf jede Gruppe*
  → *Achtung: count(*) vs. count(name)*

Having-Klausel

- Syntax:

  HAVING search-condition

- Logischer Ausdruck, NOT/AND/OR/Klammern
- Filtert Gruppen, die mit GROUP-BY definiert wurden
- HAVING ohne GROUP-BY-Klausel
  - Das Ergebnis des vorherigen SELECT wird als eine Gruppe angesehen – ohne Gruppierungsattribute.
  - Aggregatfunktion in SELECT-Klausel notwendig.

Having-Klausel Beispiele

- Durchschnittspreise für Western und teure Filme:

  SELECT genre, AVG(mietpreis) FROM FILM GROUP BY genre
  HAVING (genre=´WESTERN´ or AVG(mietpreis) > 5)

- Der (Gesamt-)Durchschnittspreis wird nur ausgegeben, wenn er größer als 5 ist.

  SELECT AVG(mietpreis) FROM film
  HAVING AVG(mietpreis) > 5

Having-Klausel Beispiele 2

- Preis der billigsten Filme für die Genre „Western“ und „Krimi“, wenn der Durchschnittspreis des jeweiligen Genres größer 3 ist. Es werden nur Filme nach 1994 berücksichtigt.

```
SELECT
  genre, MIN(mietpreis)
FROM film
WHERE jahr > 1994
GROUP BY genre
HAVING
  (genre=´Western´ OR genre=´Krimi´)
  AND AVG(mietpreis) > 3
```


Union – Klausel

- Syntax:

  subselect | (fullselect) UNION [ALL] subselect | (fullselect)

- UNION vereinigt das Ergebnis zweier SELECT-Anfragen
- UNION führt Duplikatelimination durch, UNION ALL nicht
- Alle Tabellen im UNION müssen gleiche Spaltenanzahl haben

- Beispiel:

```
SELECT titel, mietpreis, genre FROM film
UNION
SELECT titel, mietpreis, type FROM musik
```

Intersect – Klausel

- Syntax:

  subselect | (fullselect) INTERSECT [ALL] subselect | (fullselect)

- INTERSECT bildet den Durchschnitt zweier SELECT-Anfragen, d.h. gibt die Tupel aus, die in beiden Relationen erscheinen
- INTERSECT führt Duplikatelimination durch, INTERSECT ALL nicht
- Alle Tabellen beim INTERTSECT müssen gleiche Spaltenanzahl haben
- Beispiel:

  SELECT titel FROM film
  INTERSECT
  SELECT titel FROM musik

Except – Klausel

- Syntax:

  subselect | (fullselect) EXCEPT [ALL] subselect | (fullselect)

- EXCEPT bildet die Differenz zweier SELECT-Anfragen, d.h. alle Tupel der linken Relation werden ausgewählt, es sei denn, sie erscheinen in der rechten Relation
- Alle Tabellen müssen gleiche Spaltenanzahl haben

Except – Klausel Beispiel

(SELECT titel FROM film UNION ALL SELECT titel FROM film)
EXCEPT ALL
SELECT titel FROM film
→ Ergebnis entspricht SELECT titel FROM film

(SELECT titel FROM film UNION ALL SELECT titel FROM film)
EXCEPT
SELECT titel FROM film
→ Ergebnis: kein Tupel!

SOME / ANY / ALL

- Syntax:

  WHERE expression = | <> | < | > | <= | >=
  SOME | *ANY* | *ALL (fullselect)*

- SOME ist äquivalent zu ANY.
- SOME liefert TRUE, sobald der Vergleich für einen der Werte im Fullselect gilt
- ALL liefert TRUE, sobald der Vergleich für alle Werte im Fullselect gilt
- Das Fullselect muss so viele Spalten liefern, wie *expression* beinhaltet

SOME / ANY / ALL Beispiel

- Filme, deren Mietpreis teurer ist als der Mietpreis irgendwelcher Stücke der Band „Proclaimers“:

```
SELECT titel FROM film WHERE mietpreis > SOME
  (SELECT mietpreis FROM musik
    WHERE interpret=´Proclaimers´)
```

- Alle Filme, die mindestens so teuer sind wie die Musik von B. Streisand:

```
SELECT titel FROM film WHERE mietpreis >= ALL
  (SELECT mietpreis FROM musik
    WHERE interpret=´Streisand´)
```


IN

- Syntax:

```
WHERE expression [NOT] IN (fullselect)
```

- Beispiele:

```
SELECT titel FROM film
1) WHERE titel IN
     (SELECT titel from filmstars where name=´Fonda´)

2) WHERE titel NOT IN
     (´Lethal Weapon´, ´Matrix 3´)

3) WHERE (name, flaeche) IN
     (VALUES (´Tanasee´, 3630), (´Orisee´,200))
```

EXISTS

- Syntax:

  WHERE EXISTS (fullselect)

- Testet, ob das fullselect überhaupt Tupel produziert, ohne den Inhalt anzusehen

- Beispiel:

  SELECT 1 FROM one_row_table WHERE EXISTS
  (SELECT titel FROM film WHERE name=´Easy Rider´)

Korrelierte Subqueries

- Ein Attribut der äußeren Query wird in der Subquery benutzt
- Beispiel:

  SELECT titel FROM film f WHERE titel IN
  (SELECT titel FROM filmstars s
  WHERE f.darsteller = s.name
  AND s.name LIKE ´Fonda´)
- Auswertung:
  - Für jeden Wert, den das äußere Referenz-Attribut (f.darsteller) annehmen kann, wird die Subquery ausgeführt.
  - Kehrt das IN-Prädikat mit TRUE zurück, wird das entsprechende Tupel aus *film* ausgegeben

Korrelierte Subqueries

- Jede Subquery mit ´=´ oder ´IN´ kann immer durch eine flache Query ausgedrückt werden
- Beispiel

```
SELECT titel FROM film f WHERE titel IN
  (SELECT titel FROM filmstars s
   WHERE f.darsteller = s.name
   AND s.name LIKE ´Fonda´)

SELECT titel FROM film f, filmstars s
WHERE f.titel = s.titel
  AND f.darsteller = s.name
  AND s.name LIKE ´Fonda´)
```


Beispiele DELETE, UPDATE (reviseted)

Professor (ID, name, vorname, alter, anz_wimis, fakultät, familienstand,geschlecht)
Vorlesung (prof_id, name, zeit)

- Löschen Sie alle Professoren der Landwirtschaftlichen Fakultät.
  - Löschen sie alle zunächst alle Vorlesungen dieser Professoren

Subqueries in der FROM-Klausel

- Syntax:

```
FROM [TABLE] (fullselect)
  AS correlation-name (column-name {, column-name}
```

- Beispiel

```
SELECT blume.name FROM TABLE
  (SELECT art_code, pflanzenname FROM pflanzen
   WHERE sorte = ´Blume´) AS blume (art_nr, name)
WHERE blume.name <> ´Distel´
```


Skalare Subqueries

- Ein SELECT, welches nur ein Tupel mit einem Attribut liefert, kann als Skalar verwendet werden.

- Beispiel:

```
SELECT betellnummer,
        betrag – (SELECT avg(betrag) FROM bestellungen)
FROM bestellungen
```

Fancy Stuff

SELECT name FROM person
UNION
SELECT NULL FROM person

- Geht nicht, da NULL typfrei. Muss für UNION gecastet werden (siehe DB-Anleitung). DB2:

SELECT name FROM person
UNION
SELECT CAST (NULL AS CHAR) FROM person

Beispiel

- Fluss (Name, Länge)
- See (Name, Grösse)
- fliesst_durch (Fluss, Stadt)
- liegt_in (See, Stadt)
- Stadt (Name)
- Gesucht:
  - Anzahl der Städte, die am Nil liegen
  - Alle Flüsse, die nicht durch Berlin fließen und alle Seen, die nicht in Berlin liegen. Der Typ des Gewässers soll mit ausgegeben werden.

Beispiel

- Land (Name)
- Stadt (Name, Land, Bevölkerung)
- Gesucht:
  - Alle Länder, die lt. Städte-Statistik mehr als 50 Mio. EW haben
  - Alle Städte, die in Ländern liegen, die lt. Städte-Statistik mehr als 50 Mio. EW haben