SQL

Ziele

- Grundlagen von SQL
- Beziehung zur relationalen Algebra
- SELECT, FROM, WHERE
- Joins
- ORDER BY
- Aggregatfunktionen

Grundlagen

- Structured Query Language
  - Data Definition Language (DDL):
    - Definieren von Datenstrukturen
  - Data Manipulation Language (DML):
    - Verarbeiten von Daten
- Deklarative Sprache:
  - Entscheidend ist das „WAS“
  - Das „WIE“ übernimmt der Optimierer des DBMS

SELECT

- Syntax:
  - SELECT expression
  - FROM table-reference
  - WHERE search-condition
  - GROUP BY grouping-expression
  - HAVING search-condition
  - ORDER BY sort-key

SELECT – Bearbeitungsreihenfolge

SQL	Relationale Algebra
FROM	Kartesisches Produkt, Join
WHERE	Selektion
SELECT	Projektion
GROUP BY	„Gruppierung“
HAVING	„Filter“ für GROUP BY
INTERSECT	Durchschnitt
UNION	Vereinigung
EXCEPT	Differenz
ORDER BY	„Sortieren“

SELECT – Klausel

- Syntax:
  - SELECT [DISTINCT] expression
  - Projektion in der relationalen Algebra
- Beispiel ohne Aggregatfunktion
  - **SELECT** a.name, angestellter.name **FROM** abteilung a, angestellter
  - **SELECT DISTINCT** abteilung.name **FROM** abteilung → Duplikat-Elimination durch DISTINCT
  - **SELECT** preis+10 **FROM** angebot → Preise um 10 erhöht (Arithemetik)
  - **SELECT** 1 **FROM** person → viele Einsen…

FROM – Klausel 1

- Syntax:
  - FROM table-reference [AS correlation-name]
    {, table-reference [AS correlation-name]}
  - Kartesisches Produkt
- Beispiele:
  - **SELECT** * **FROM** Abteilung
  - **SELECT** * **FROM** Abteilung AS a, Mitarbeiter AS b;
  - → Kartesisches Produkt

Kartesisches Produkt

- Gefordert: Ein kartesisches Produkt zwischen R mit m Attributen und S mit n Attributen
- Ergebnis: Relation Q mit m+n Attributen

ID	Name
1	Max
2	Frank

ID	Name
1	BMW
2	Audi

A.ID	A.Name	B.ID	B.Name
1	Max	1	BMW
1	Max	2	Audi
2	Frank	1	BMW
2	Frank	2	Audi

FROM – Klausel 2

- Benutzen von join:
  - FROM table-reference
    [join|left outer join|right outer join|full outer join]
    ON join-attribut
- Beispiel:
  - **SELECT** * **FROM**
    mitarbeiter **AS** m **JOIN** abteilung **AS** a
    ON m.abt_id = a.id

Joins

- Join == Kartesisches Produkt mit Selektion
- **Natural/Inner Join**: join-attribute muss in beiden Relationen gefüllt sein und übereinstimmen
- **Left Outer Join**: alle Zeilen aus der linken Tabelle, ergänzt um die übereinstimmenden Zeilen der rechten Tabelle
- **Right Outer Join**: alle Zeilen aus der rechten Tabelle, ergänzt um die übereinstimmenden Zeilen der linken Tabelle
- **Full Outer Join**: alle Zeilen aus beiden Tabellen, wobei im Join-Kriterium übereinstimmende Zeilen „gejoint“ werden.

Inner / Natural Join

A

Name	Nummer
Katja	1
Sven	NULL
Ralf	5

B

ID	Nummer	Marke
50	1	BMW
51	2	Mercedes
53	NULL	Wartburg

A JOIN B ON A.Nummer = B.Nummer

Name	A.Nummer	ID	B.Nummer	Marke
Katja	1	50	1	BMW

Left Outer Join

A

Name	Nummer
Katja	1
Sven	NULL
Ralf	5

B

ID	Nummer	Marke
50	1	BMW
51	2	Mercedes
53	NULL	Wartburg

A LEFT [OUTER] JOIN B ON A.Nummer = B.Nummer

Name	A.Nummer	ID	B.Nummer	Marke
Katja	1	50	1	BMW
Sven	NULL	NULL	NULL	NULL
Ralf	5	NULL	NULL	NULL

Right Outer Join

A

Name	Nummer
Katja	1
Sven	NULL
Ralf	5

B

ID	Nummer	Marke
50	1	BMW
51	2	Mercedes
53	NULL	Wartburg

A RIGHT [OUTER] JOIN B ON A.Nummer = B.Nummer

Name	A.Nummer	ID	B.Nummer	Marke
Katja	1	50	1	BMW
NULL	NULL	51	2	Mercedes
NULL	NULL	52	NULL	Wartburg

Full Outer Join

A

Name	Nummer
Katja	1
Sven	NULL
Ralf	5

B

ID	Nummer	Marke
50	1	BMW
51	2	Mercedes
53	NULL	Wartburg

A FULL [OUTER] JOIN B ON A.Nummer = B.Nummer

Name	A.Nummer	ID	B.Nummer	Marke
Katja	1	50	1	BMW
NULL	NULL	51	2	Mercedes
NULL	NULL	52	NULL	Wartburg
Sven	NULL	NULL	NULL	NULL
Ralf	5	NULL	NULL	NULL

WHERE-Klausel

- Syntax:
  - WHERE search-condition
  - Logischer Ausdruck, bestehend aus Prädikaten, die mit NOT, AND, OR, (und) kombiniert werden können
  - Filtert das Ergebnis der FROM-Klausel (Selektion)
  - Kann join-Expression simulieren
- Beispiel:
  - **SELECT** * **FROM** angestellter **WHERE** gehalt < 350
  - **SELECT** * **FROM** a,b **WHERE** a.id = b.id

WHERE-Klausel 2

- BETWEEN-Prädikat
  - WHERE c BETWEEN a AND b:
  - A <= C <= B
- LIKE-Prädikat
  - WHERE c LIKE pattern
  - Pattern beinhaltet Wildcards
    - _ für ein Zeichen
    - % für beliebige (auch leere) Zeichenkette
  - **WHERE** name **LIKE** ´%arazin´
- NULL-Prädikat
  - WHERE expression IS [NOT] NULL
  - **SELECT** * **FROM** angestellter **WHERE** telefon is **NOT NULL**

ORDER BY – Klausel

- Syntax:
  - ORDER BY sort-key [ASC | DESC] {, sort-key [ASC|DESC] }
    → ordnet das Ergebnis der SELECT-Klausel bezüglich der Attributliste
- Beispiele
  - **SELECT** * **FROM** angestellter **AS** a
    ORDER BY name
  - **SELECT** name, ort, strasse **FROM** person
    ORDER BY 1

Aggregatfunktionen

- AVG, COUNT, MAX, MIN, SUM, STDDEV, VARIANCE

- Syntax:
  - SELECT aggrFunc([DISTINCT] expression) ...
  - SELECT COUNT([DISTINCT] expression | *)
- I.d.R. mit GROUP BY verwendet
- Beispiel:
  - **SELECT COUNT**(*) **FROM** PERSON;
  - **SELECT AVG**(gehalt) **FROM** angestellter;

Beispiele SELECT

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

Vorlesung (prof_id, name, zeit)

- Geben Sie alle weiblichen ledigen Professoren aus!
- Alle ledigen Professoren über 60 werden gekürzt. Wer ist das?
- Geben Sie die Namen und Vornamen der Informatikprofessoren an, die außer Prof. Freytag noch registriert sind.
- Alle Professoren über 60 werden gekürzt. Wie viele sind das?
- Wie viele WiMis gibt es?
- Welche Vorlesungen entfallen dank der gekürzten Professoren (alle über 60)?

Beispiele SELECT

Person (ID, name)

Student (person_id, matrikel_no, fachsemester)

- Geben Sie Namen und Matrikel-Nummern aller Studenten aus. Benutzen Sie join.
- Geben Sie alle Personen aus und bei Studenten zusätzlich die Matrikel-Nummer!
- Finden sie die Namen aller Langzeitstudenten (fachsemester>10) heraus.

UPDATE

- Syntax
  - UPDATE table-name SET attribut1=wert1 {,attribut2=wert2} [WHERE where-Klausel]
- Beispiel
  - **UPDATE** student **SET** kontostand=kontostand-500, fachsemester=fachsemester+1 **WHERE** fachsemester>10

DELETE

- Syntax:
  - DELETE FROM table-name [WHERE where-condition]
- Beispiel
  - **DELETE FROM** politiker_privilegien
  - **DELETE FROM** unis **WHERE** city=´Berlin´

Beispiele DELETE, UPDATE

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

- Löschen Sie alle Professoren der Landwirtschaftlichen Fakultät.
- Wir befinden uns im Jahr 2009. Löschen Sie alle Professoren über 56!

- Reduzieren Sie die Anzahl aller WiMis auf die Hälfte.
- „Verheiraten“ Sie alle Professoren über 60!