Einführung in die relationale Datenbanksprache SQL[1]

In diesem Script sollen einige grundlegende Elemente der Sprache SQL dargestellt werden. Es wird auf die Beispielrelationen PRO1, IMMA, und NOTEN zurückgegriffen.

1. Projektion und Selektion bezüglich einer Tabelle

Typisch für in SQL formulierte Anfrageanweisungen ist die `SELECT-FROM-WHERE`-Klausel.

```
SELECT matnr,lst
FROM noten
WHERE db < 2;
```

2. Ausdrucksmöglichkeiten der `WHERE`-Klausel

allgemein: `WHERE <search condition>` (`true` oder `false`)
Vergleichsoperatoren: =; <>; <; <=; >; >=
Prädikate können mit `AND`, `OR` und `NOT` zu komplexeren Bedingungen verknüpft werden, beachte Priorität: `NOT` → `AND` → `OR` oder Klammerung.

`IN`-Prädikat: → vereinfachte `ODER`-Verknüpfung

```
SELECT matnr,name FROM imma WHERE ort IN
('Berlin','Halle');

SELECT matnr,lst FROM noten WHERE matnr
IN
(SELECT matnr FROM noten WHERE db < 2);
```

(...) entspricht einer subquery, diese enthält eine Menge von Ergebnistupeln, der in der Klammer stehenden Anfrage.

`BETWEEN`-Prädikat:

entspricht einer Verknüpfung >= `AND` <=

```
SELECT matnr FROM noten WHERE db
BETWEEN 1 AND 2;
```

`LIKE`-Prädikat:

nur für Datentyp `CHAR`, es werden Teilzeichenketten auf Gleichheit oder Ungleichheit geprüft.

```
SELECT name FROM pro1 WHERE name LIKE '%ei_';
```

Ergebnis: Klein, da _ - genau ein Zeichen und % - n>=0 Zeichen (ACCESS ? = _ bzw. * = %)

[1] entnommen aus: Keller, Bernd: Script zur Vorlesung *Theorie und Praxis relationaler Datenbanken*, TU Dresden, SS 2002

3. Duplikate in Ergebnistabellen und Sortierung

```
SELECT name
FROM imma
WHERE name = 'Meier';
```
→ Name Meier kommt mehrfach vor

```
SELECT DISTINCT name
FROM imma
WHERE name = 'Meier';
```
→ Mehrfach vorkommende Feldeinträge werden gestrichen
→ die Ergebnisrelation enthält nur einmal den Namen Meier

Sortierung:

```
ORDER BY <column name > [ ASC | DESC]
```

ASC → aufsteigende Sortierung
DESC → absteigende Sortierung

4. Verbundanfragen (`JOIN`) mit Projektion und Selektion

4.1 Innerer Verbund, d.h. mit Verbundbedingung (`INNER JOIN`)

Beispiel: Von den Studenten, die am Projekt1 mitarbeiten, werden die Matrikelnummer, der Name, der Heimatort und der Lehrstuhl gesucht.

```
SELECT pro1.matnr, pro1.name, ort, lehrst
FROM pro1 INNER JOIN imma ON pro1.matnr = imma.matnr
```

→ entspricht direktem Abbild des Relationenalgebra-Joins in SQL

4.2 Äußerer Verbund (`OUTER JOIN`)

Ein 'normaler' Join liefert nur solche Ergebniszeilen, für die beide Operanden der Join-Bedingung nicht `NULL` sind. Dies bedeutet, dass so genannte ‚dangling tuples' (d.h. solche, die beim Verknüpfen der Tabellen keinen Partner finden) nicht angezeigt werden.
Aber: Falls ein Tupel des Verbundes sinnvoll ist, obwohl nur eine Seite der Join-Bedingung einen Wert `IS NOT NULL` hat, ist der outer-join anwendbar.

Beispiel: In der Notentabelle `NOTEN` fehlt aus irgendwelchen Gründen der Student mit Matrikelnummer `OUZT9`. Dann sind folgende Ergebnisse möglich:

```
SELECT imma.matnr,name,ort,lst,db
FROM imma INNER JOIN noten ON
imma.matnr = noten.matnr;
```

aber(!)

```
SELECT imma.matnr,name,ort,lst,db
FROM imma LEFT JOIN noten ON
imma.matnr = noten.matnr;
```

5. Gruppenfunktionen (Built-In-Funktionen, Aggregatfunktionen)

Mit Ausnahme bei der Verwendung von `DISTINCT` sind folgende Ausdrücke möglich:
`SUM, MIN, MAX, AVG, COUNT`

5.1 Die gesamte Tabelle als Gruppe

```
SELECT MAX(db) FROM noten;
```

5.2 Zerlegung einer Tabelle in n >= 1 Gruppen

```
SELECT lst,avg(db) FROM noten GROUP BY lst;
```

In der SELECT-Klausel dürfen außer Merkmalen, die in der GROUP BY-Klausel definiert sind, nur aggregierende Funktionen bezüglich dieser Gruppen stehen.

5.3 Einschränkung der Menge der auszuwertenden Gruppen

```
SELECT lst,avg(db),count(matnr)
FROM noten
GROUP BY lst
HAVING count(matnr) > 1;
```

Die `HAVING`-Klausel ist als '`WHERE`-Klausel' für Gruppen von Zeilen zu verstehen.

6. Verbundanfragen mit Gruppenfunktionen

```
SELECT lehrstuhl.lst,lname,count(db)
FROM lehrstuhl INNER JOIN noten ON lehrstuhl.lst=noten.lst
GROUP BY lehrstuhl.lst,lname;
```

Abarbeitungsreihenfolge:
`FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY`

7. Nullwerte

Durch den Indikator `NULL` werden Merkmalswerte mit der Semantik
(a) unbekannt
(b) noch nicht bekannt
(c) nicht relevant codiert.

8. Arithmetische Ausdrücke

gebildet mit den Operatoren +, -, *, /
Arithmetische Ausdrücke können in der `<select list>`, der `<where clause>`, der `<having clause>` stehen. Speziell können sie auch in Vergleichs-, `BETWEEN`-und `IN`-Prädikaten vorkommen.