Creation of views

- views correspond to external DB schemas
- In relational DBS views are regarded as derived relations which are defined by queries.
- **create view** `<view name> [(<attribute name> [, <attribute name>]*)] as <subquery>`
  - example:
    ```
    create view major_students as
      select * from students where sem > 4
    ```
    The keyword “*” is a shortcut for the complete attribute list of those relations placed after `from`.

Deletion of views

- **drop view** `<view name>`
5.3 Data Manipulation Language (DML)

select-from-where clause

- simple form:
  - `select distinct A_1, A_2, ..., A_n`
  - `from R_1, R_2, ..., R_m`
  - `where F`
  - `A_1, A_2, ..., A_n` attribute names, `R_1, R_2, ..., R_m` relation names, predicate `F`

- equivalent to the following relational algebra expression:
  \( \pi_{A_1, A_2, ..., A_n}(\sigma_F(R_1 \times R_2 \times ... \times R_m)) \)

- The `select` clause corresponds to the projection operation of the relational algebra and not to the selection operation! The `from` clause corresponds to the Cartesian product and the `where`-clause to the selection operation of the relational algebra.

- The predicate `F` after the `where` clause contains
  - comparison operators `=, <, <>, <=, >, >=`
  - boolean operators `and, or, not`
  - set operations `in, not in, any, some, all`
If the **where** clause is omitted, \( F = true \) holds.

The result of an SQL query can contain the same tuple multiple times (multiset!).

If different relations have attributes with equal names, these are distinguished by the relation name.

**Transfer of the operations of the relational algebra into SQL**

- **relation \( R \)**
  - `select * from R`
  - The declaration "*" in the `select`-clause indicates that all attributes of the relation \( R \) after the `from`-clause belong to the output.

- **projection \( \pi_{A,B}(R) \)**
  - `select distinct A, B from R`
  - Without the keyword **distinct** the result is a multiset (multi-relation).

- **selection \( \sigma_F(R) \)**
  - `select distinct * from R where F`

- **Cartesian product \( R \times S \)**
  - `select * from R, S`
theta join \( R \bowtie_F S \) on relations \( R(A, B) \) and \( S(C, D) \)

\[
\text{select} \ * \ \text{from} \ R, S \ \text{where} \ F
\]

union \( R \cup S \) of the relations \( R(A, B) \) and \( S(A, B) \)

\[
\text{select} \ * \ \text{from} \ R \ \text{union} \ \text{select} \ * \ \text{from} \ S
\]

difference \( R - S \) of the relations \( R(A, B) \) and \( S(A, B) \)

\[
\text{select} \ * \ \text{from} \ R \ \text{minus} \ \text{select} \ * \ \text{from} \ S
\]

Duplicates and duplicate elimination

- The usual select clause does not eliminate duplicates in the result relation, which therefore is a multiset (multi-relation). But this can be done by using the keyword distinct so that a relation is created as output.

- The minus operation on two multisets corresponds to the semantics of the extended relational algebra. Keyword minus is only used by Oracle. In SQL92 the keyword except is used instead.

- The union operation defined on relations automatically eliminates duplicates. If duplicates are not to be eliminated, the keyword all has to follow the keyword union.
Examples for SQL queries

- Find all personell ids and names of C4 professors.

  ```sql
  select pers-id, name from professors where rank = "C4"
  ```

<table>
<thead>
<tr>
<th>pers-id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2125</td>
<td>Sokrates</td>
</tr>
<tr>
<td>2126</td>
<td>Russel</td>
</tr>
<tr>
<td>2136</td>
<td>Curie</td>
</tr>
<tr>
<td>2137</td>
<td>Kant</td>
</tr>
</tbody>
</table>

- A strength of SQL is based on the fact that it is near to a natural language formulation of a command.
- Determine the different ranks of professors.

```sql
select distinct rank from professors
```

<table>
<thead>
<tr>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>C3</td>
</tr>
<tr>
<td>C4</td>
</tr>
</tbody>
</table>

- elimination of duplicates in a table is not automatically executed for efficiency reasons (sorting necessary)

- keyword `distinct` for explicit duplicate elimination

- Determine the names of professors who hold the lecture titled “maieutics”.

```sql
select name, title
from professors, lectures
where pers-id = held_by and title = "maieutics"
```

<table>
<thead>
<tr>
<th>name</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sokrates</td>
<td>maieutics</td>
</tr>
</tbody>
</table>
Which students attend which lecture? Output student names and lecture titles.

```sql
select name, title
from students, attends, lectures
where students.reg-id = attends.reg-id and
      attends.id = lectures.id
```

alternative formulation using tuple variables that are associated to relations:

```sql
select s.name, l.title
from students as s, attends as a, lectures as l
where s.reg-id = a.reg-id and
      a.id = l.id
```

relationship to the tuple relational calculus observable: a variable is bound to tuples of a relation

Determine the names of all university employees, i.e., the names of all professors and all assistants.

```
(select name
 from assistants)
union
(select name
 from professors)
```