Chapter 6’: SQL – Data Retrieval

Content:

• More features for data retrieval in SQL: Aggregation, grouping, more joins, case ..

Next:

• Physical data organization: indexing
Aggregates and Grouping

\[
\text{select } \text{avg(semester)} \text{ from students;}
\]

<table>
<thead>
<tr>
<th>StudNr</th>
<th>Name</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>24002</td>
<td>Xenokrates</td>
<td>18</td>
</tr>
<tr>
<td>25403</td>
<td>Jonas</td>
<td>12</td>
</tr>
<tr>
<td>26120</td>
<td>Fichte</td>
<td>10</td>
</tr>
<tr>
<td>26830</td>
<td>Aristoxenos</td>
<td>8</td>
</tr>
<tr>
<td>27550</td>
<td>Schopenhauer</td>
<td>6</td>
</tr>
<tr>
<td>28106</td>
<td>Carnap</td>
<td>3</td>
</tr>
<tr>
<td>29120</td>
<td>Theophrastos</td>
<td>2</td>
</tr>
<tr>
<td>29555</td>
<td>Feuerbach</td>
<td>2</td>
</tr>
</tbody>
</table>

Result

\[
1.0
\]

7.625
Aggregates and Grouping

```sql
select semester, min(birthdate) from students group by semester
```

<table>
<thead>
<tr>
<th>StudNr</th>
<th>Name</th>
<th>Semester</th>
<th>birthdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>24002</td>
<td>Xenokrates</td>
<td>12</td>
<td>1989-03-14</td>
</tr>
<tr>
<td>25403</td>
<td>Jonas</td>
<td>12</td>
<td>1994-01-14</td>
</tr>
<tr>
<td>26120</td>
<td>Fichte</td>
<td>10</td>
<td>1984-05-12</td>
</tr>
<tr>
<td>26830</td>
<td>Aristoxenos</td>
<td>3</td>
<td>1993-06-25</td>
</tr>
<tr>
<td>27550</td>
<td>Schopenhauer</td>
<td>3</td>
<td>1995-11-03</td>
</tr>
<tr>
<td>28106</td>
<td>Carnap</td>
<td>3</td>
<td>1990-08-30</td>
</tr>
<tr>
<td>29120</td>
<td>Theophrastos</td>
<td>2</td>
<td>1994-01-14</td>
</tr>
<tr>
<td>29555</td>
<td>Feuerbach</td>
<td>2</td>
<td>1993-07-01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester</td>
</tr>
<tr>
<td>1989-03-14</td>
</tr>
<tr>
<td>1984-05-12</td>
</tr>
<tr>
<td>1990-08-30</td>
</tr>
<tr>
<td>1993-07-01</td>
</tr>
</tbody>
</table>
Characteristics of Aggregates

- SQL creates one result tuple per group
- All attributes of the select-clause—except the aggregated—have to be listed in the group by-clause
- Thus SQL can make sure that the attribute value does not change within a group
- NULL value is an own group
- Aggregates `avg`, `max`, `min`, `count`, `sum`
Aggregates and Grouping

Find C4 professors that work for at most 8 hours a week on teaching.
<table>
<thead>
<tr>
<th>PersNr</th>
<th>Name</th>
<th>Level</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>2127</td>
<td>Kopernikus</td>
<td>C3</td>
<td>310</td>
</tr>
<tr>
<td>2133</td>
<td>Popper</td>
<td>C3</td>
<td>52</td>
</tr>
<tr>
<td>2134</td>
<td>Augustinus</td>
<td>C3</td>
<td>309</td>
</tr>
<tr>
<td>2136</td>
<td>Curie</td>
<td>C4</td>
<td>36</td>
</tr>
<tr>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StudNr</th>
<th>Name</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>24002</td>
<td>Xenokrates</td>
<td>18</td>
</tr>
<tr>
<td>25403</td>
<td>Jonas</td>
<td>12</td>
</tr>
<tr>
<td>26120</td>
<td>Fichte</td>
<td>10</td>
</tr>
<tr>
<td>26830</td>
<td>Aristoxenos</td>
<td>8</td>
</tr>
<tr>
<td>27550</td>
<td>Schopenhauer</td>
<td>6</td>
</tr>
<tr>
<td>28106</td>
<td>Carnap</td>
<td>3</td>
</tr>
<tr>
<td>29120</td>
<td>Theophrastos</td>
<td>2</td>
</tr>
<tr>
<td>29555</td>
<td>Feuerbach</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture Nr</th>
<th>Title</th>
<th>Weekly Hours</th>
<th>Given by</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>Grundzüge</td>
<td>4</td>
<td>2137</td>
</tr>
<tr>
<td>5041</td>
<td>Ethik</td>
<td>4</td>
<td>2125</td>
</tr>
<tr>
<td>5043</td>
<td>Erkenntnistheorie</td>
<td>3</td>
<td>2126</td>
</tr>
<tr>
<td>5049</td>
<td>Mäeutik</td>
<td>2</td>
<td>2125</td>
</tr>
<tr>
<td>4052</td>
<td>Logik</td>
<td>4</td>
<td>2125</td>
</tr>
<tr>
<td>5052</td>
<td>Wissenschaftstheorie</td>
<td>3</td>
<td>2126</td>
</tr>
<tr>
<td>5216</td>
<td>Bioethik</td>
<td>2</td>
<td>2126</td>
</tr>
<tr>
<td>5259</td>
<td>Der Wiener Kreis</td>
<td>2</td>
<td>2133</td>
</tr>
<tr>
<td>5022</td>
<td>Glaube und Wissen</td>
<td>2</td>
<td>2134</td>
</tr>
<tr>
<td>4630</td>
<td>Die 3 Kritiken</td>
<td>4</td>
<td>2137</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StudNr</th>
<th>LectureNr</th>
<th>PersNr</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>28106</td>
<td>5001</td>
<td>2126</td>
<td>1</td>
</tr>
<tr>
<td>25403</td>
<td>5041</td>
<td>2125</td>
<td>2</td>
</tr>
<tr>
<td>27550</td>
<td>4630</td>
<td>2137</td>
<td>2</td>
</tr>
</tbody>
</table>
Aggregates and Grouping

Find C4 professors that work for at most 8 hours a week on teaching.

```sql
select PersNr, Name, sum (WeeklyHours) 
from Lectures, Professors
where Given_by = PersNr and Level = 'C4'
group by PersNr, Name
having sum (WeeklyHours) <= 8;
```
In the following slides we step through the (logical) execution of the following query:

\begin{verbatim}
select PersNr, Name, sum (WeeklyHours) 
from Lectures, Professors
where Given_by = PersNr and Level = 'C4'
group by PersNr, Name
having sum (WeeklyHours) <= 8;
\end{verbatim}
## 1. Cross Product

### Lectures x Professors

<table>
<thead>
<tr>
<th>Lecture Nr</th>
<th>Title</th>
<th>Weekly Hours</th>
<th>Given_by</th>
<th>PersNr</th>
<th>Name</th>
<th>Level</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>Grundzüge</td>
<td>4</td>
<td>2137</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>5041</td>
<td>Ethik</td>
<td>4</td>
<td>2125</td>
<td>2133</td>
<td>Popper</td>
<td>C3</td>
<td>52</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>4630</td>
<td>Die 3 Kritiken</td>
<td>4</td>
<td>2137</td>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
</tbody>
</table>

↓ Next: `where`-clause
2. Where Filtering

<table>
<thead>
<tr>
<th>Lecture Nr</th>
<th>Title</th>
<th>Weekly Hours</th>
<th>Given_by</th>
<th>PersNr</th>
<th>Name</th>
<th>Level</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>Grundzüge</td>
<td>4</td>
<td>2137</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>5041</td>
<td>Ethik</td>
<td>4</td>
<td>2125</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>4630</td>
<td>Die 3 Kritiken</td>
<td>4</td>
<td>2137</td>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ \text{Next: join condition} \]
### 3. Join Condition

<table>
<thead>
<tr>
<th>Lecture Nr</th>
<th>Title</th>
<th>Weekly Hours</th>
<th>Given_by</th>
<th>PersNr</th>
<th>Name</th>
<th>Level</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>Grundzüge</td>
<td>4</td>
<td>2137</td>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
<tr>
<td>5041</td>
<td>Ethik</td>
<td>4</td>
<td>2125</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>5043</td>
<td>Erkenntnistheorie</td>
<td>3</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>5049</td>
<td>Mäeutik</td>
<td>2</td>
<td>2125</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>4052</td>
<td>Logik</td>
<td>4</td>
<td>2125</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>5052</td>
<td>Wissenschaftstheorie</td>
<td>3</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>5216</td>
<td>Bioethik</td>
<td>2</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>4630</td>
<td>Die 3 Kritiken</td>
<td>4</td>
<td>2137</td>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
</tbody>
</table>

**Next: group by**

10-Jan-19  
Database System Concepts for Non-Computer Scientists WS 2018/2019
4. Grouping

<table>
<thead>
<tr>
<th>Lecture Nr</th>
<th>Title</th>
<th>Weekly Hours</th>
<th>Given_by</th>
<th>PersNr</th>
<th>Name</th>
<th>Level</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>5041</td>
<td>Ethik</td>
<td>4</td>
<td>2125</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>5049</td>
<td>Mäeutik</td>
<td>2</td>
<td>2125</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>4052</td>
<td>Logik</td>
<td>4</td>
<td>2125</td>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>5043</td>
<td>Erkenntnistheorie</td>
<td>3</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>5052</td>
<td>Wissenschaftstheo.</td>
<td>3</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>5216</td>
<td>Bioethik</td>
<td>2</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>5001</td>
<td>Grundzüge</td>
<td>4</td>
<td>2137</td>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
<tr>
<td>4630</td>
<td>Die 3 Kritiken</td>
<td>4</td>
<td>2137</td>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
</tbody>
</table>

↓ Next: **having** filtering
### 5. Having Filtering

<table>
<thead>
<tr>
<th>Lecture Nr</th>
<th>Title</th>
<th>Weekly Hours</th>
<th>Given_by</th>
<th>PersNr</th>
<th>Name</th>
<th>Level</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>5043</td>
<td>Erkenntnistheorie</td>
<td>3</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>5052</td>
<td>Wissenschaftstheo.</td>
<td>3</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>5216</td>
<td>Bioethik</td>
<td>2</td>
<td>2126</td>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>5001</td>
<td>Grundzüge</td>
<td>4</td>
<td>2137</td>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
<tr>
<td>4630</td>
<td>Die 3 Kritiken</td>
<td>4</td>
<td>2137</td>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
</tbody>
</table>

↓ Next: Select aggregation (**sum**) and **projektion**
## Result

<table>
<thead>
<tr>
<th>PersNr</th>
<th>Name</th>
<th>\textbf{sum} (WeeklyHours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2126</td>
<td>Russel</td>
<td>8</td>
</tr>
<tr>
<td>2137</td>
<td>Kant</td>
<td>8</td>
</tr>
</tbody>
</table>
Maximum / Minimum

Student with the highest StudNr

```sql
select StudNr, Name
from Students
where StudNr =
  (select max(StudNr)
   from Student);
```

NOT

```sql
select Name, max(StudNr)
from Students;
```
Using the Result Set of a Sub-Query

```sql
select tmp.StudNr, tmp.Name, tmp.Number_of_Lectures
from (select s.StudNr, s.Name, count(*) as Number_of_Lectures
    from Students s, attend a
    where s.StudNr=a.StudNr
    group by s.StudNr, s.Name) tmp
where tmp. Number_of_Lectures > 2;
```

<table>
<thead>
<tr>
<th>StudNr</th>
<th>Name</th>
<th>Number_of_Lectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>28106</td>
<td>Carnap</td>
<td>4</td>
</tr>
<tr>
<td>29120</td>
<td>Theophrastos</td>
<td>3</td>
</tr>
</tbody>
</table>
... or alternatively

```sql
select tmp.StudNr, tmp.Name, tmp.Number_of_Lectures
from (select s.StudNr, s.Name, count(*) as Number_of_Lectures
    from Students s, attend a
    where s.StudNr = a.StudNr
    group by s.StudNr, s.Name
    having count(*) > 2) tmp;
```
**Decision-Support-Query with Nested Sub-Queries**

```sql
SELECT n.LectureNr, n.NumPerLect, t.TotalNum, 
    n.NumPerLect / t.TotalNum AS MarketShare 
FROM (SELECT LectureNr, COUNT(*) AS NumPerLect 
    FROM attend 
    GROUP BY LectureNr ) n, 
( SELECT COUNT(*) AS TotalNum 
    FROM Students) t;
```
### Result of the Query ?!

<table>
<thead>
<tr>
<th>LectureNr</th>
<th>NumPerLect</th>
<th>TotalNum</th>
<th>MarketShare</th>
</tr>
</thead>
<tbody>
<tr>
<td>4052</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>5001</td>
<td>3</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>5022</td>
<td>2</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Decision-Support-Query with Nested Sub-Queris

```sql
select n.LectureNr, n.NumPerLect, t.TotalNum,
    cast(n.NumPerLect as decimal(6,2)) / t.TotalNum as MarketShare
from ( select LectureNr, count(*) as NumPerLect
    from attend
    group by LectureNr ) n,
( select count(*) as TotalNum
    from Students ) t;
```
### Professors

<table>
<thead>
<tr>
<th>PersNr</th>
<th>Name</th>
<th>Level</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>2125</td>
<td>Sokrates</td>
<td>C4</td>
<td>226</td>
</tr>
<tr>
<td>2126</td>
<td>Russel</td>
<td>C4</td>
<td>232</td>
</tr>
<tr>
<td>2127</td>
<td>Kopernikus</td>
<td>C3</td>
<td>310</td>
</tr>
<tr>
<td>2133</td>
<td>Popper</td>
<td>C3</td>
<td>52</td>
</tr>
<tr>
<td>2134</td>
<td>Augustinus</td>
<td>C3</td>
<td>309</td>
</tr>
<tr>
<td>2136</td>
<td>Curie</td>
<td>C4</td>
<td>36</td>
</tr>
<tr>
<td>2137</td>
<td>Kant</td>
<td>C4</td>
<td>7</td>
</tr>
</tbody>
</table>

### Students

<table>
<thead>
<tr>
<th>StudNr</th>
<th>Name</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>24002</td>
<td>Xenokrates</td>
<td>18</td>
</tr>
<tr>
<td>25403</td>
<td>Jonas</td>
<td>12</td>
</tr>
<tr>
<td>26120</td>
<td>Fichte</td>
<td>10</td>
</tr>
<tr>
<td>26830</td>
<td>Aristoxenos</td>
<td>8</td>
</tr>
<tr>
<td>27550</td>
<td>Schopenhauer</td>
<td>6</td>
</tr>
<tr>
<td>28106</td>
<td>Carnap</td>
<td>3</td>
</tr>
<tr>
<td>29120</td>
<td>Theophrastos</td>
<td>2</td>
</tr>
<tr>
<td>29555</td>
<td>Feuerbach</td>
<td>2</td>
</tr>
</tbody>
</table>

### Lectures

<table>
<thead>
<tr>
<th>Lecture Nr</th>
<th>Title</th>
<th>Weekly Hours</th>
<th>Given_by</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>Grundzüge</td>
<td>4</td>
<td>2137</td>
</tr>
<tr>
<td>5041</td>
<td>Ethik</td>
<td>4</td>
<td>2125</td>
</tr>
<tr>
<td>5043</td>
<td>Erkenntnistheorie</td>
<td>3</td>
<td>2126</td>
</tr>
<tr>
<td>5049</td>
<td>Mäeutik</td>
<td>2</td>
<td>2125</td>
</tr>
<tr>
<td>4052</td>
<td>Logik</td>
<td>4</td>
<td>2125</td>
</tr>
<tr>
<td>5052</td>
<td>Wissenschaftstheorie</td>
<td>3</td>
<td>2126</td>
</tr>
<tr>
<td>5216</td>
<td>Bioethik</td>
<td>2</td>
<td>2126</td>
</tr>
<tr>
<td>5259</td>
<td>Der Wiener Kreis</td>
<td>2</td>
<td>2133</td>
</tr>
<tr>
<td>5022</td>
<td>Glaube und Wissen</td>
<td>2</td>
<td>2134</td>
</tr>
<tr>
<td>4630</td>
<td>Die 3 Kritiken</td>
<td>4</td>
<td>2137</td>
</tr>
</tbody>
</table>

### Assistants

<table>
<thead>
<tr>
<th>PersNr</th>
<th>Name</th>
<th>Area</th>
<th>Boss</th>
</tr>
</thead>
<tbody>
<tr>
<td>3002</td>
<td>Platon</td>
<td>Ideenlehre</td>
<td>2125</td>
</tr>
<tr>
<td>3003</td>
<td>Aristoteles</td>
<td>Syllogistik</td>
<td>2125</td>
</tr>
<tr>
<td>3004</td>
<td>Wittgenstein</td>
<td>Sprachtheorie</td>
<td>2126</td>
</tr>
<tr>
<td>3005</td>
<td>Rhetikus</td>
<td>Planetenbewegung</td>
<td>2127</td>
</tr>
<tr>
<td>3006</td>
<td>Newton</td>
<td>Keplersche Gesetze</td>
<td>2127</td>
</tr>
<tr>
<td>3007</td>
<td>Spinoza</td>
<td>Gott und Natur</td>
<td>2126</td>
</tr>
</tbody>
</table>

### Test

<table>
<thead>
<tr>
<th>StudNr</th>
<th>LectureNr</th>
<th>PersNr</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>28106</td>
<td>5001</td>
<td>2126</td>
<td>1</td>
</tr>
<tr>
<td>25403</td>
<td>5041</td>
<td>2125</td>
<td>2</td>
</tr>
<tr>
<td>27550</td>
<td>4630</td>
<td>2137</td>
<td>2</td>
</tr>
</tbody>
</table>
## Result of the query

<table>
<thead>
<tr>
<th>LectureNr</th>
<th>NumPerLect</th>
<th>TotalNum</th>
<th>MarketShare</th>
</tr>
</thead>
<tbody>
<tr>
<td>4052</td>
<td>1</td>
<td>8</td>
<td>.125</td>
</tr>
<tr>
<td>5001</td>
<td>3</td>
<td>8</td>
<td>.375</td>
</tr>
<tr>
<td>5022</td>
<td>2</td>
<td>8</td>
<td>.25</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Further Queries with Sub-Queries

```sql
select Name
from Professors
where PersNr not in ( select Given_by
                      from Lectures );

select Name
from Students
where Semester >= all ( select Semester
                         from Students );
```
Case Construct

```sql
select StudNr, ( case when Grade < 1.5 then ´very good´
when Grade < 2.5 then ´good´
when Grade < 3.5 then ´satisfactory´
when Grade < 4.0 then ´sufficient´
else ´failed´ end)
from test;
```

First qualifying when-clause is executed
Joins in SQL-92

- **cross join**: full cartesian product (not in all DBS!)
- **natural join**: equality test on all attributes with the same names, output of all attributes, those with the same names only once (not in all DBS!)

<table>
<thead>
<tr>
<th>Name</th>
<th>StudNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carnap</td>
<td>1</td>
</tr>
<tr>
<td>Fichte</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StudNr</th>
<th>LectureNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Databases</td>
</tr>
<tr>
<td>2</td>
<td>Math</td>
</tr>
</tbody>
</table>
Joins in SQL-92

- **join** also called **inner join**: Theta Join (any conditions), equi-join (only equal condition)
- **left, right or full outer join**: keeps tuples with no join partner

<table>
<thead>
<tr>
<th>Name</th>
<th>StudNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carnap</td>
<td>1</td>
</tr>
<tr>
<td>Fichte</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StudNr</th>
<th>LectureNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Databases</td>
</tr>
<tr>
<td>2</td>
<td>Math</td>
</tr>
</tbody>
</table>
Joins in SQL-92

- **semi-join**: no operator in SQL, expressed with `exists` or `in`
(Inner) Join

\[
\text{select } * \\
\text{from } R_1, R_2 \\
\text{where } R_1.A = R_2.B;
\]

or alternatively

\[
\text{select } * \\
\text{from } R_1 \text{ join } R_2 \text{ on } R_1.A = R_2.B;
\]
Outer Joins (left)

```
from (Professors p
    left outer join test t
    left outer join Students s
        on t.StudNr = s.StudNr;
```
## Result

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2126</td>
<td>Russel</td>
<td>2126</td>
<td>1</td>
<td>28106</td>
<td>28106</td>
<td>Carnap</td>
</tr>
<tr>
<td>2125</td>
<td>Sokrates</td>
<td>2125</td>
<td>2</td>
<td>25403</td>
<td>25403</td>
<td>Jonas</td>
</tr>
<tr>
<td>2137</td>
<td>Kant</td>
<td>2137</td>
<td>2</td>
<td>27550</td>
<td>27550</td>
<td>Schopenhauer</td>
</tr>
<tr>
<td>2136</td>
<td>Curie</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outer Joins (right)

```sql
from (Professors p
  right outer join test t
  right outer join Students s
    on t.StudNr = s.StudNr;
```
Result

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2126</td>
<td>Russel</td>
<td>2126</td>
<td>1</td>
<td>28106</td>
<td>28106</td>
<td>Carnap</td>
</tr>
<tr>
<td>2125</td>
<td>Sokrates</td>
<td>2125</td>
<td>2</td>
<td>25403</td>
<td>25403</td>
<td>Jonas</td>
</tr>
<tr>
<td>2137</td>
<td>Kant</td>
<td>2137</td>
<td>2</td>
<td>27550</td>
<td>27550</td>
<td>Schopenhauer</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>26120</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

null

null

null

null

null

null

26120

Fichte
Outer Joins (full)

```
from (Professors p
    full outer join test t
    full outer join Students s
    on t.StudNr = s.StudNr;
```
Null values

- In SQL there is a special value **NULL**
- This value exists for all data types and represents values which are
  - unknown or
  - *not available* or
  - *not applicable*
- Null values can also emerge from query evaluation

- Test for NULL  →  **is NULL**

Example:
```
select *  
from Professors  
where Room is NULL;
```
Null values cont.

• Null values are passed through in arithmetic expressions: at least one operand NULL $\rightarrow$ result is NULL as well
• Sometimes surprising query results, if Null values occur, e.g.:

```sql
select count (*)
from Students
where Semester < 13 or Semester >= 13
```

• If there are students whose attribute value semester is a NULL value these are not counted
• The reason is three-valued logic with inclusion of NULL values
Null values cont.

\[ \text{sum(semester)} = 14 \]
\[ \text{count(*)} = 3 \]
\[ \text{count(semester)} = 2 \]
\[ \text{avg(semester)} = 7 \]
Exercise SQL

Average Grade of the student Schopenhauer.

Name and number of given lectures for all professors.
Exercise SQL (cont.)

```sql
select name, avg(grade), count(*)
from students s left outer join test t
    on s.studnr = t.studnr
where name = 'Schopenhauer'
group by name;

select persNr, name, count(lectureNr) as numLectures
from Professors p left outer join Lectures l
    on p.persNr = l.given_by
where level = 'C4'
group by p.persNr, p.name
order by numLectures desc
```
Data Manipulation Language
DML
Insert of tuples by explicitly giving values:

```
insert into Students (StudNr, Name)
values (28121, 'Archimedes'), (4711, 'Pythagoras');
```
Changes in the database: Insert

Insert of tuples via a query

```
insert into attend
    select StudNr, LectureNr
from Students, Lectures
where Title= 'Logik';

(Mandatory registration of all students for 'Logik')
```
Insert of tuples from a file

Database system specific programs, e.g. DB2:

- **Import:**
  
  \[
  \begin{align*}
  &\text{IMPORT FROM studis.tbl OF DEL} \\
  &\text{INSERT INTO Students;}
  \end{align*}
  \]

  Analogously:

  \[
  \begin{align*}
  &\text{EXPORT TO studis.tbl OF DEL} \\
  &\text{SELECT * FROM Students;}
  \end{align*}
  \]

- **Load:**

  High-Performance alternative to import

  Oracle: Load, Datapump, ...
Changes in the Database: delete, update

delete from Students

where Semester > 13;

Note: delete from Students deletes all tuples from the relation

update Students

set Semester = Semester + 1;
Changes in two phase

1. Candidates for changes are determined and marked

2. Changes are performed at the marked tuples

Otherwise changes can depend on the order of the tuples.

```sql
delete from Require
  where predecessor in (select successor
                         from Require);
```
Example

delete from Require
where predecessor in (select successor
from Require);

A
Info

DBMS
B
Logic
C
Math
Example 2

display from Require
where predecessor in (select successor
from Require);

<table>
<thead>
<tr>
<th>predecessor</th>
<th>successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>5041</td>
</tr>
<tr>
<td>5001</td>
<td>5043</td>
</tr>
<tr>
<td>5001</td>
<td>5049</td>
</tr>
<tr>
<td>5041</td>
<td>5216</td>
</tr>
<tr>
<td>5043</td>
<td>5052</td>
</tr>
<tr>
<td>5041</td>
<td>5052</td>
</tr>
<tr>
<td>5052</td>
<td>5229</td>
</tr>
</tbody>
</table>

Execution in order of the example instance would (also) contain tuple (5052, 5229) erroneously as before all tuples with 5052 as **Successor** were deleted.
Data Definition Language
DDL
Changes to the schema

- **drop table** `<Table name>`
- **alter table** `<Table name>`
  - `drop` | `add column` `<Attribute name>` `<Data type>`
  - `alter column` `<Attribute name>` **set default** `<default>`
  - ...

Further commands vendor specific, e.g. Oracle:

- **alter table** `<Table name>`
  - **modify** | `add column` `<Attribute name>` `<Data type>`
  - `drop column` `<Attribute name>`
  - `add` | `drop` | `enable` | `disable` `<constraint clause>`
Views ...

- Belong to DDL: `create view <view name> as <select-statement>`
- Often used to design queries more clear
- Are kind of a "virtual relation"
- Show an excerpt of the database

Advantages
- Simplify the access for certain user groups
- Can be used to restrict the access to the data

Disadvantages
- Not all (mostly none) views can be modified
Remember

```sql
select tmp.StudNr, tmp.Name, tmp.Number_of_Lectures
from (select s.StudNr, s.Name, count(*) as Number_of_Lectures
    from Students s, attend a
    where s.StudNr = a.StudNr
    group by s.StudNr, s.Name
    having count(*) > 2) tmp;
```
... alternatively

```sql
create view tmp (StudNr, Name, Number_of_Lectures) as
(select s.StudNr, s.Name, count(*)
    from Students s, attend a
    where s.StudNr = a.StudNr
    group by s.StudNr, s.Name)

select * from tmp where Number_of_Lectures > 2;

drop view tmp;
```
... alternatively

```sql
with tmp (StudNr, Name, Number_of_Lectures) as
(select s.StudNr, s.Name, count(*)
from Students s, attend a
where s.StudNr=a.StudNr
group by s.StudNr, s.Name)

select *
from tmp

where Number_of_Lectures > 2;

→ With creates a temporary table, only valid within the query
```
Simplifying Queries with Views

**Complex query**: Names of all professors who give a lecture with more weekly hours than the average weekly hours per lecture and with more than three assistants.

- Not all at once → divide into smaller more concise parts
- These parts can be realized by using views or or named intermediate results (‘with’)

Database System Concepts for Non-Computer Scientists WS 2018/2019
1. All professors with weekly hours more than the average of weekly hours:

```sql
create view AboveAverageWeeklyHours as
select given_by
from Lectures
where WeeklyHours >
  (select avg (WeeklyHours) from Lectures);
```
Simplification

2. All professors with more than three assistants:

```sql
create view ManyAssistants as
  select Boss
  from Assistants
  group by Boss
  having count(*) > 3;
```
Simplification

• Combine
• Views can be used like common relations

```sql
select Name
from Professors
where PersNr in
  (select given_by
     from AboveAverageWeeklyHours)
  and PersNr in
  (select Boss
     from ManyAssistants);
```
Expanding when executed

```sql
select Name
from Professors
where PersNr in
  (select Given_by
   from (select Given_by
       from Lectures
       where WeeklyHours >
         (select avg (WeeklyHours)
          from Lectures))
   and
   PersNr in
   (select Boss
    from (select Boss
        from Assistants
        group by Boss
        having count(*) > 3));
```

AboveAverageWeeklyHours

ManyAssistants
Views ...

For data privacy

create view testView as
  select StudNr, LectureNr, PersNr
  from test

For statistics

create view TestQual(Name, QualLevel) as
  (select p.Name, avg(t.Grade)
   from Professors p join test t on
   group by p.Name, p.PersNr
   having count(*) > 50)
Relational Modelling of the Generalization

Assistants

Professors

Employees

Area

Room

Level

PersNr

Name

Employees: \{[PersNr, Name]\}

Professors: \{[PersNr, Level, Room]\}

Assistants: \{[PersNr, Area]\}
**Table Definition**

```sql
create table Employees
    (PersNr integer not null,
     Name varchar(30) not null);

create table ProfData
    (PersNr integer not null,
     Level character(2),
     Room integer);

create table AssData
    (PersNr integer not null,
     Area varchar(30));
```
Views to model generalization

create view Professors as
    select *
    from employees e, ProfData p
    where e.PersNr=p.PersNr;

create view Assistants as
    select *
    from Employees e, AssData d
    where e.PersNr=d.PersNr;

➤ subtypes as view
Table Definition

**create table** Professors

(PersNr integer not null,
Name varchar (30) not null,
Level character (2),
Room integer);

**create table** Assistants

(PersNr integer not null,
Name varchar (30) not null,
Area varchar (30) );

**create table** OtherEmployees

(PersNr integer not null,
Name varchar (30) not null);
Views to Model Generalization

```sql
create view Employees as
    (select PersNr, Name
     from Professors)
    union
    (select PersNr, Name
     from Assistants)
    union
    (select *
     from OtherEmployees);

→ supertype as view
```
Views to guarantee data independency

- Logical data independency
- Physical data independency
Modifiability of views

In SQL

- Only one base relation
- Key must be part of
- No aggregation, grouping, duplicate elimination

(all views)

-theoretical modifiable views

-in SQL modifiable views
Quiz

Table Airplane:

<table>
<thead>
<tr>
<th>Producer</th>
<th>Type</th>
<th>NumberSeats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing</td>
<td>B747-400</td>
<td>550</td>
</tr>
<tr>
<td>Boeing</td>
<td>B737-300</td>
<td>380</td>
</tr>
<tr>
<td>Airbus</td>
<td>A340-600</td>
<td>380</td>
</tr>
<tr>
<td>Airbus</td>
<td>A320-200</td>
<td>179</td>
</tr>
<tr>
<td>Airbus</td>
<td>A380</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Every producer together with its type of airplane with the most seats

Result:

<table>
<thead>
<tr>
<th>Producer</th>
<th>Type</th>
<th>SeatsMax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing</td>
<td>B747-400</td>
<td>550</td>
</tr>
<tr>
<td>Airbus</td>
<td>A340-600</td>
<td>380</td>
</tr>
</tbody>
</table>
Quiz: Solution

with GroupProducer (Producer, SeatsMax) as

(select Producer, max (NumberSeats) from Airplane group by Producer)

select A.Producer, Type, SeatsMax
from Airplane A, GroupProducer G
where A.Producer = G.Producer and A.Seats = G.SeatsMax