SQL Summary

Definitions

- **1. SQL definition**

SQL is a declarative query language

- **2. Components**

DRL: Data Retrieval Language

DML: Data Manipulation Language

DDL: Data Definition Language

DCL: Data Control Language

TCL: Transaction Control Language

DDL EXAMPLE

```
CREATE TABLE Lectures
(LectureNr INTEGER NOT NULL,
Title VARCHAR(30) NOT NULL UNIQUE,
WeeklyHours INTEGER CHECK(WeeklyHours >0 AND
WeeklyHours<20),
Given_by INTEGER,
PRIMARY KEY (LectureNr),
CONSTRAINT givenby_fk FOREIGN KEY (Given_by)
REFERENCES Professors (PersNr)
(Given_by INTEGER REFERENCES Professors)
On DELETE SET NULL/ ON DELETE CASCADE);

 CREATE TABLE Professors (
 PersNr  INTEGER NOT NULL,
 Name VARCHAR(30) NOT NULL
 PRIMARY KEY (PersNr) );
```

SQL Implementation

- **1. Skeleton SQL Query**

Select <> f**rom** <> **where** <>

group by <> / **having** <> /**order by** <>

	SQL expression
Duplicate Elimination	Select distinct
Where clause	And/or/not/between/is NULL
wildcard	Like '…%'
Select all	*
Name collision solution	s.name, p.name from student s, prof p
Aggregation	Min, avg, max, count, sum, Group by
Sorting	Order by
Keep/not keep tuples without join partners	Inner join/left outer join/right outer join/full outer join

- 2. Subqueries /nested queries

```
select s.*
from Students s
where exists
        (select p.*
        from Professors p
        where p.Birthdate > s.Birthdate);
```

```
select s.*
from Students s
where s.Birthdate <
        (select max (p.Birthdate)
         from Professors p);
```

Exam question 1

- Search the student with the best grade and give the name, the title of the lecture, and the professor of this lecture.

Professors

PersNr	Name	Level	Room
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Students

StudNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Lectures

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

attend

StudNr	LectureNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
25403	5022
29555	5022
29555	5001

require

Predecessor	Successor
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

test

StudNr	LectureNr	PersNr	Grade
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Assistants

PersNr	Name	Area	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

Exam question 1

```
select s.name as studentname, l.title,
p.name as professorname, t.grade
from professors p, students s, lectures l, test t
where  t.studNr=s.StudNr and
t.LectureNr=l.LectureNr and
t.PersNr=p.PersNr and t.grade=
        (select min(t.Grade)from test t)
```

Exam question 2

- All students with their names and number of lectures they attend , who attend lectures which also Fichte attends

Professors

PersNr	Name	Level	Room
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Students

StudNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Lectures

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

attend

StudNr	LectureNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
25403	5022
29555	5022
29555	5001

require

Predecessor	Successor
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

Assistants

PersNr	Name	Area	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

test

StudNr	LectureNr	PersNr	Grade
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Exam question 2

```
select tmp.name, count(a.LectureNr)
from attend a,
        (select a.StudNr, s.name
        from Students s, attend a
        where s.Name!='Fichte' and s.StudNr=a.StudNr
        and a.LectureNr =
                (select a.LectureNr
                from Students s, attend a
                where s.Name='Fichte' and
                s.StudNr=a.StudNr))tmp
where tmp.StudNr=a.StudNr
group by tmp.name
```

Exam question 2

```
select s.name, count(a.LectureNr)
from attend a, Students s
where s.StudNr=a.StudNr and a.StudNr in
        (select a.StudNr
        from Students s, attend a
        where s.Name!='Fichte' and s.StudNr=a.StudNr
        and  a.LectureNr =
                (select a.LectureNr
                from Students s, attend a
                where s.Name='Fichte' and
                s.StudNr=a.StudNr))
group by s.name
```

Supplementary

- **1. DDL: Views**
- Syntax**: create view** <*name*> **as** <select-statement>
- Query stored, but not the result
- + simplify and restrict access, model generalization;
  - View can‘t be modified

- Views are used like relations
- Names of all professors who give lectures with credits > average of credits and with > 3 assistants

Select Name
From Professors
Where PersNr **in** (**select** Given_by **from** *AboveAverageWeeklyHours*) **and** PersNr **in** (**select** Boss **from** *ManyAssistants*);

Supplementary

- **2. Variants of SQL**

• Database cannot only be used interactively

• SQL can be embedded in other programming languages (embedded SQL)

• Problem : SQL is set oriented, most programming languages aren‘t

Supplementary

- **3. Relational tuple calculus**
- Syntax: {t | P(t)} where t: Tuple variable, P: predicate
- Procedural description
- Students who attend at least one lecture of Curie
  {s | s ∈ Students
∧ ∃ h ∈ attend (s.StudNr =h.StudNr
∧ ∃ v ∈ Lectures (h.LectureNr=v.LectureNr
∧ ∃ p ∈ Professors (p. PersNr=v.Given_by
∧ p.Name = 'Curie'))))}

- ∧ = and; ∈ = element of; ∃ = exist; ∪ = Union; ∩ = Intersection; ρ = renaming; σ = selection; Π = Projection; x= Cartesian Product;