AALBORG
UNIVERSITY

Database System (SW5)

3. SQL

Tiantian Liu
Department of Computer Science
Aalborg University
Fall 2025

Motivation

AALBORG UNIVERSITY

- SQL is the query language
- SQL is very widely used
- SQL is well supported by relational database systems

Learning Goals

AALBORG UNIVERSITY

- Explain and use the SQL data model
- Create non-trivial database tables
- Modify non-trivial database tables
- Create non-trivial SQL statements

Agenda

- Overview of the SQL Query Language
- SQL Data Definition and Modification
- SQL Queries
- Additional Basic Operations
- Set Operations
- Aggregate Functions

AALBORG UNIVERSITY

SQL Parts

AALBORG UNIVERSITY

- **Data-definition language (DDL)** -- provides commands for defining relation schemas, deleting relations, and modifying relation schemas.
- **Data-manipulation language (DML)** -- provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.
- **Integrity** – the DDL includes commands for specifying integrity constraints.
- **View definition** -- the DDL includes commands for defining views.
- **Transaction control** –includes commands for specifying the beginning and ending of transactions.
- **Authorization** – includes commands for specifying access rights to relations and views.

Domain Types in SQL

AALBORG UNIVERSITY

- **char(n).** Fixed length character string, with user-specified length *n*.
- **varchar(n).** Variable length character strings, with user-specified maximum length *n*.
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point. (**numeric**(3,1), allows 44.5 to be stores exactly, but not 444.5 or 0.32)

Domain Types in SQL (Cont.)

AALBORG UNIVERSITY

- **date.** A calendar date containing a (four-digit) year, month, and day of the month.
- **time.** The time of day, in hours, minutes, and seconds.
- **timestamp.** A combination of date and time.
- Date and time values can be specified like this:
  - date '2018-04-25’
  - time '09:30:00’
  - timestamp '2018-04-25 10:29:01.45'

SQL Syntax

AALBORG UNIVERSITY

- Reserved words / Keywords
  - There is a set of **reserved words** that cannot be used as names for database objects.
  - **SELECT**, **FROM**, **WHERE**, **CREATE TABLE**, **UNIQUE**, etc.
- Case-insensitive
  - SQL is generally case-insensitive.
    - Example: **SELECT** is same to **select**
    - Exception: string constants. 'FRED' not the same as 'fred'.
  - Use **single quotes** for string constants:
    - 'abc' – Okay
    - “abc” – Not okay
- White-space is ignored
- All statements end with a semicolon (;)

Agenda

- Overview of the SQL Query Language
- SQL Data Definition and Modification
- SQL Queries
- Additional Basic Operations
- Set Operations
- Aggregate Functions

AALBORG UNIVERSITY

Create Table

AALBORG UNIVERSITY

- An SQL relation is defined using the create table command:

```
create table r
        (A1 D1,
         A2 D2,
         ...,
         An Dn,
      (integrity-constraint 1),
           ...,
      (integrity-constraint k));
```

  - *r* is the name of the relation
  - each A_i is an attribute name in the schema of relation *r*
  - D_i is the data type of values in the domain of attribute A_i

Create Table--Example

AALBORG UNIVERSITY

- Example:

```
create table instructor (
    ID          char(5),
    name        varchar(20),
    dept_name   varchar(20),
    salary      numeric(8,2));
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Integrity Constraints

AALBORG UNIVERSITY

- Types of integrity constraints
  - **primary key** (A_1, ..., A_n)
    - The primary key constraint includes the **not null** and **unique** constraints.
  - **foreign key** (A_m, ..., A_n) **references** *r*
    - Foreign keys can also reference candidate keys enforced with **unique** constraints
  - **not null**
  - **unique**
  - **default**
- SQL prevents any update to the database that violates an integrity constraint.

Primary Key

AALBORG UNIVERSITY

- Example:

```
create table instructor (
    ID          char(5),
    name        varchar(20)  not null,
    dept_name   varchar(20),
    salary      numeric(8,2),
    primary key (ID),
    foreign key (dept_name)
    references department
    );
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Primary Key (Cont.)

AALBORG UNIVERSITY

- Example:

```
create table instructor (
    ID          char(5) primary key,
    name        varchar(20) not null,
    dept_name   varchar(20)
    references department,
    salary      numeric(8,2)
    );
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Foreign Key

AALBORG UNIVERSITY

- *country*(*ID*, *name*)
- *city*(*ID*, *name*, *country_ID*)
- Example:

```
create table country (
        ID              char(5),
        name            varchar(20)   unique not null,
        primary key (ID));
```

```
create table city (
        ID              char(5),
        name            varchar(20)   not null,
        country_ID      char(5),
        primary key (ID),
        foreign key (country_ID) references country (ID));
```

Default

AALBORG UNIVERSITY

- Example:

```
create table t1 (
        i       int default -1,
        c       varchar(10) default ' ',
        price double(16,2) default 0.00);
```

Exercise 0 (5 minutes)

AALBORG UNIVERSITY

Create a database

1. Go to the link https://db-book.com/university-lab-dir/sample_tables-dir/index.html (can be found in Moodle: *In_class_Schema_Instances*)

2. Download

    a. DDL/DDL with drop table

    b. SQL code for creating small relations

3. Copy and paste the code, run it in PostgreSQL (see *Instruction_PostgreSQL* for reference)

4. If you have done that, try to understand the code and be familiar with the tables

Updates to tables

AALBORG UNIVERSITY

- Insert

```
insert into instructor values
('10211', 'Smith', 'Biology',
66000);
```

- Delete
  - Remove all tuples from the *instructor* relation

```
delete from instructor;
```

- Drop Table

```
drop table instructor;
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Updates to tables-- Alter

AALBORG UNIVERSITY

- **alter table** *r* **add** *A D;*
  - where *A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A*.
  - All exiting tuples in the relation are assigned *null* as the value for the new attribute.
- **alter table** *r* **drop** *A;*
  - where *A* is the name of an attribute of relation *r*
  - Dropping of attributes not supported by many databases.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Exercise 1 (5 minutes)

AALBORG UNIVERSITY

Write the SQL statements

- Add an attribute to the table
  - gender: the type could be char(1)
- Insert an instructor into the table.
  - ID: 10001
  - name: Tiantian
  - dept_name: Comp. Sci.
  - salary: 70000
  - gender: F
- Drop the attribute: gender

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Agenda

- Overview of the SQL Query Language
- SQL Data Definition and Modification
- SQL Queries
- Additional Basic Operations
- Set Operations
- Aggregate Functions

AALBORG UNIVERSITY

Basic Query Structure

AALBORG UNIVERSITY

- A typical SQL query has the form:

select $A_1, A_2, \ldots, A_n$
from $r_1, r_2, \ldots, r_m$
where P;

  - A_i represents an attribute
  - r_i represents a relation
  - P is a predicate
- The result of a SQL query is a relation
- Mapping to Relational Algebra

$$\Pi_{A1,A2,A3\ldots.An}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

The select Clause

AALBORG UNIVERSITY

- The **select** clause lists the attributes desired in the result of a query
  - corresponds to the projection operation of the relational algebra
- Example: find the names of all instructors:

```
select name
from instructor;
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

The select Clause (Cont.)

AALBORG UNIVERSITY

- Find the department names of all instructors, and remove duplicates

```
select distinct dept_name
from instructor;
```

dept_name
Comp. Sci.
Finance
Music
Physics
History
Biology
Elec. Eng.

- The keyword **all** specifies that duplicates should not be removed.

```
select all dept_name
from instructor;
```

dept_name
Comp. Sci.
Finance
Music
Physics
History
Physics
Comp. Sci.
History
Finance
Biology
Comp. Sci.
Elec. Eng.

The select Clause (Cont.)

AALBORG UNIVERSITY

- The **select** clause can contain arithmetic expressions involving the operation, +, –, *, and /, and operating on constants or attributes of tuples.
- The query:

```
select ID, name, salary/12
from instructor;
```

  - would return a relation that is the same as the instructor relation, except that the value of the attribute salary is divided by 12.
- Can rename "*salary/12*" using the **as** clause:

```
select ID, name, salary/12 as monthly_salary
From instructor;
```

The select Clause (Cont.)

AALBORG UNIVERSITY

```
select ID, name, salary * 1.2 as new_salary
From instructor;
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	EI Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

ID	*name*	*new_salary*
10101	Srinivasan	78000
12121	Wu	108000
15151	Mozart	48000
22222	Einstein	114000
32343	EI Said	72000
33456	Gold	104400
45565	Katz	90000
58583	Califieri	74400
76543	Singh	96000
76766	Crick	86400
83821	Brandt	110400
98345	Kim	96000

instructor

The select Clause (Cont.)

AALBORG UNIVERSITY

- An asterisk in the select clause denotes “all attributes”

```
select *
from instructor;
```

The where Clause

AALBORG UNIVERSITY

- The **where** clause specifies conditions that the result must satisfy
  - Corresponds to the selection predicate of the relational algebra.
- To find names of all instructors in Comp. Sci. dept

```
select name
from instructor
where dept_name = 'Comp. Sci.';
```

- SQL allows the use of the logical connectives **and**, **or**, and **not**
- The operands of the logical connectives can be expressions involving the comparison operators <, <=, >, >=, =, and <>.

The where Clause (Cont.)

AALBORG UNIVERSITY

- To find names of all instructors in Comp. Sci. dept with *salary* > 70000

```
select name
from instructor
where dept_name = 'Comp. Sci.' and salary > 70000;
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

name
Katz
Brandt

Where Clause Predicates

AALBORG UNIVERSITY

- SQL includes a **between** comparison operator
- Example: Find the names of all instructors with salary between 90,000 and 100,000 (that is, >= 90,000 and <= 100,000)

```
select name
from instructor
where salary between 90000 and 100000;
```

The from Clause

AALBORG UNIVERSITY

- The **from** clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra.
- Find the Cartesian product *instructor X teaches*

```
select *
from instructor, teaches;
```

  - generates every possible instructor – teaches pair, with all attributes from both relations.
- For common attributes (e.g., *ID*), the attributes in the resulting table are renamed using the relation name (e.g., *instructor.ID*)
- Cartesian product not very useful directly, but useful combined with **where**-clause condition (selection operation in relational algebra).

The from Clause--Example

AALBORG UNIVERSITY

```
select *
from instructor, teaches;
```

instructor.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

The from Clause--Example

AALBORG UNIVERSITY

- Find the names of all instructors who have taught some course, and the course_id

```
select name, course_id
from instructor , teaches
where instructor.ID = teaches.ID;
```

name	*course_id*
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

Sequence of Steps

AALBORG UNIVERSITY

select $A_1, A_2, \ldots, A_n$
from $r_1, r_2, \ldots, r_m$
where P;

- **Step 1:** Generate a Cartesian product of the relations listed in the **from** clause.
- **Step 2:** Apply the predicates specified in the **where** clause on the result of Step 1.
- **Step 3:** For each tuple in the result of Step 2, output the attributes (or results of expressions) specified in the **select** clause.
- Note: This sequence of steps helps make clear what **the result of an SQL query** should be, not how it should be executed.

Exercise 2

AALBORG UNIVERSITY

Write the SQL statements

- Find ID and name of all instructors in Physics dept with *salary* > 90000

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Agenda

- Overview of the SQL Query Language
- SQL Data Definition and Modification
- SQL Queries
- Additional Basic Operations
- Set Operations
- Aggregate Functions

AALBORG UNIVERSITY

The Rename Operation

AALBORG UNIVERSITY

- The SQL allows renaming relations and attributes using the **as** clause:

old-name **as** *new-name*

- Example

```
select s.ID, d.ID
from department as d, student as s
where d.ID = s.dept_ID;
```

Rename relations

The Rename Operation (Cont.)

AALBORG UNIVERSITY

- Example

```
select s.ID as stuent_ID, d.ID as department_ID,
from department as d, student as s
where d.ID= s.dept_ID;
```

Rename attributes and relations

- Keyword **as** is optional and may be omitted

  department **as** *d* ≡ *department d*

String Operations

AALBORG UNIVERSITY

- The SQL standard specifies that the equality operation on strings is **case sensitive**;
  - 'comp. sci.' = 'Comp. Sci.' evaluates to **false**
- **upper**(s): converting string s to uppercase
- **lower**(s): converting string s to lowercase
- **trim**(s): removing spaces at the end of the string

String Operations (Cont.)

AALBORG UNIVERSITY

- The operator **like** uses patterns that are described using two special characters:
  - percent (%). The % character matches any substring.
  - underscore (_). The _ character matches any character.
- Pattern matching examples:
  - 'Intro%' matches any string beginning with “Intro”.
  - '%Comp%' matches any string containing “Comp” as a substring.
  - '_ _ _' matches any string of exactly three characters.
  - '_ _ _ %' matches any string of at least three characters.

String Operations (Cont.)

AALBORG UNIVERSITY

- Example
  - Find the names of all instructors whose name includes the substring "dar".

```
select name
from instructor
where name like '%dar%';
```

- Match the string "100%"

```
like '100\%' escape  '\'
```

in that above we use backslash (\) as the escape character.

Ordering the Display of Tuples

AALBORG UNIVERSITY

- List in alphabetic order the names of all instructors

```
select name
from   instructor
order by name;
```

- We may specify **desc** for descending order or **asc** for ascending order, for each attribute; ascending order is the default.
  - Example: **order by** *name* **desc**
- Can sort on multiple attributes
  - Example: **order by** *dept_name, name*

Order By--Example

AALBORG UNIVERSITY

```
select name, salary
from   instructor
order by salary DESC;
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	65000
22222	Einstein	Physics	95000
83821	Brandt	Comp. Sci.	65000
33456	Gold	Physics	87000

instructor

name	*salary*
Einstein	95000
Wu	90000
Gold	87000
Srinivasan	65000
Mozart	65000
Brandt	65000

Order by *salary* **DESC**

Order By--Example

AALBORG UNIVERSITY

```
select name, salary
from   instructor
order by salary DESC, name;
```

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	65000
22222	Einstein	Physics	95000
83821	Brandt	Comp. Sci.	65000
33456	Gold	Physics	87000

instructor

name	*salary*
Einstein	95000
Wu	90000
Gold	87000
Srinivasan	65000
Mozart	65000
Brandt	65000

Order by *salary* **DESC**

name	*salary*
Einstein	95000
Wu	90000
Gold	87000
Brandt	65000
Mozart	65000
Srinivasan	65000

Order by *salary* **DESC**, *name*

Exercise 3

AALBORG UNIVERSITY

Write the SQL statements

- Retrieve ID and name of all instructors whose name starts with the letter "C".
- Retrieve the name, dept_name, and salary of all instructors whose salary exceeds 70,000. Display the results in descending order based on salary.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Agenda

- Overview of the SQL Query Language
- SQL Data Definition and Modification
- SQL Queries
- Additional Basic Operations
- Set Operations
- Aggregate Functions

AALBORG UNIVERSITY

Set Operations

AALBORG UNIVERSITY

- Set operations **union**, **intersect**, and **except**
  - Each of the above operations automatically eliminates duplicates
- To retain all duplicates using the following operations
  - **union all**
  - **intersect all**
  - **except all**

Set Operations (Cont.)

AALBORG UNIVERSITY

- Example: Find courses that ran in Fall 2017 or in Spring 2018

```
(select course_id  from section where semester = 'Fall' and year = 2017)
union
(select course_id  from section where semester = 'Spring' and year = 2018);
```

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

section

Result

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

Set Operations (Cont.)

AALBORG UNIVERSITY

- Example: Find courses that ran in Fall 2017 or in Spring 2018

```
(select course_id  from section where semester = 'Fall' and year = 2017)
union all
(select course_id  from section where semester = 'Spring' and year = 2018);
```

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

section

Result

course_id
CS-101
CS-101
CS-315
CS-319
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

Set Operations (Cont.)

AALBORG UNIVERSITY

- Example: Find courses that ran in Fall 2017 and in Spring 2018

```
(select course_id  from section where semester = 'Fall' and year = 2017)
intersect
(select course_id  from section where semester = 'Spring' and year = 2018);
```

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

section

Result

course_id
CS-101

Set Operations (Cont.)

AALBORG UNIVERSITY

- Example: Find courses that ran in Fall 2017 but not in Spring 2018

```
(select course_id  from section where semester = 'Fall' and year = 2017)
except
(select course_id  from section where semester = 'Spring' and year = 2018);
```

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

section

Result

course_id
CS-347
PHY-101

Set Operations (Cont.)

AALBORG UNIVERSITY

- Set operations require union compatibility
  - Same number of attributes
  - Compatible domains: identical domains, domains based on characters, or domains based on numerical values
- Result schema: column names of the first table

```
select ID, name from instructor
union
select stud_ID, name from student;
```

ID	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	65000

instructor

stud_ID	*name*	*dept_name*
1001	Anna	Physics
1002	Christian	Comp. Sci.
1003	David	Physics

student

ID	*name*
10101	Srinivasan
12121	Wu
15151	Mozart
1001	Anna
1002	Christian
1003	David

Agenda

- Overview of the SQL Query Language
- SQL Data Definition and Modification
- SQL Queries
- Additional Basic Operations
- Set Operations
- Aggregate Functions

AALBORG UNIVERSITY

Aggregate Functions

AALBORG UNIVERSITY

- These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

Aggregate Functions--Example

AALBORG UNIVERSITY

- Find the average salary of instructors in the Computer Science department

```
select avg (salary)
from instructor
where dept_name = 'Comp. Sci.';
```

(65000 + 75000 + 70000) / 3 = 70000

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	70000
98345	Kim	Elec. Eng.	80000

instructor

Aggregate Functions--Example

AALBORG UNIVERSITY

- Find the number of tuples in the *instructor* relation

```
select count (*) as number
from instructor;
```

number
12

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	EI Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Aggregate Functions

AALBORG UNIVERSITY

- The argument columns can optionally be accompanied by the keyword **distinct** and **all** (except in case of **count**(*))
- **distinct**: duplicates are removed before evaluating the aggregate function
- **all**: duplicates are considered for evaluation (default)
- **null** values are removed before evaluation (except in case of **count**(*))

Aggregate Functions--Example

AALBORG UNIVERSITY

- Find the number of different departments

```
select count(distinct dept_name) as number
from instructor;
```

number
7

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	EI Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Aggregate Functions--Example

AALBORG UNIVERSITY

```
select count(distinct dept_name) as number
from instructor;
```

number
6

```
select count(dept_name) as number
from instructor;
```

number
10

```
select count(*) as number
from instructor;
```

number
12

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	EI Said	History	60000
33456	Gold	null	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	null	80000

instructor

Aggregate Functions-- Group By

AALBORG UNIVERSITY

- Find the average salary of instructors in each department

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name;
```

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregate Functions-- Having Clause

AALBORG UNIVERSITY

- Find the names and average salaries of all departments whose average salary is greater than 42000

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name
having avg (salary) > 42000;
```

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
~~Music~~	~~40000~~
Physics	91000

Sequence of operations

AALBORG UNIVERSITY

- **Step 1**: **from** clause
- **Step 2**: **where** clause
- **Step 3**: **group by** clause
- **Step 4**: **having** clause
- **Step 5**: **select** clause

Aggregate Functions-- Example

AALBORG UNIVERSITY

```
select dept_name, avg (salary) as avg_salary
from instructor
Where salary > 70000
group by dept_name
having avg (salary) > 75000;
```

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

ID	name	dept_name	salary
12121	Wu	Finance	90000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Aggregate Functions--Example

AALBORG UNIVERSITY

```
select dept_name, avg (salary) as avg_salary
from instructor
Where salary > 70000
group by dept_name
having avg (salary) > 75000;
```

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

ID	name	dept_name	salary
12121	Wu	Finance	90000
76543	Singh	Finance	80000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
76766	Crick	Biology	72000
98345	Kim	Elec. Eng.	80000

Aggregate Functions--Example

AALBORG UNIVERSITY

```
select dept_name, avg (salary) as avg_salary
from instructor
Where salary > 70000
group by dept_name
having avg (salary) > 75000;
```

ID	*name*	*dept_name*	*salary*
12121	Wu	Finance	90000
76543	Singh	Finance	80000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
76766	Crick	Biology	72000
98345	Kim	Elec. Eng.	80000

avg (*salary*)	**avg** (*salary*) > 75000
85000	true
91000	true
88500	true
72000	flase
80000	true

Aggregate Functions--Example

AALBORG UNIVERSITY

```
select dept_name, avg (salary) as avg_salary
from instructor
Where salary > 70000
group by dept_name
having avg (salary) > 75000;
```

ID	*name*	*dept_name*	*salary*
12121	Wu	Finance	90000
76543	Singh	Finance	80000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
76766	Crick	Biology	72000
98345	Kim	Elec. Eng.	80000

dept_name	*avg_salary*
Finance	85000
Physics	91000
Comp. Sci.	88500
Elec. Eng.	80000

Exercise 4

AALBORG UNIVERSITY

Write the SQL statements

- Retrieve the dept_name and instructor number of all departments whose instructor number is greater than 2

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor

Summary

AALBORG UNIVERSITY

- Overview of the SQL Query Language
- SQL Data Definition and Modification
  - Create table, update table
- SQL Queries
  - Select, from, where
- Additional Basic Operations
  - Rename, String, order by
- Set Operations
  - Union, intersect, except
- Aggregate Functions
  - Avg, min, max, sum, count
  - Group by

Next Lecture

- SQL—Part 2
  - Explain and use the SQL data model
  - Create non-trivial database tables
  - Modify non-trivial database tables
  - Create non-trivial SQL statements