Lecture 9

Database Usage (and Construction)

SQL Queries and Relational Algebra

Views

Summary so far

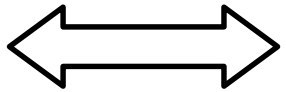
- SQL is based on relational algebra.
  - Operations over relations
- Operations for:
  - Selection of rows (σ)
  - Projection of columns (π)
  - Combining tables
    - Cartesian product (x)
    - Join, natural join ($\bowtie_C$, $\bowtie$)

SELECT-FROM-WHERE

- Basic structure of an SQL query:

```
SELECT attributes
FROM   tables
WHERE  tests over rows
```

```
SELECT X
FROM   T
WHERE  C
```



$\pi_X(\sigma_C(T))$

Example:

```
SELECT code, name, period
FROM   Courses, GivenCourses
WHERE  teacher = 'Niklas Broberg'
       AND code = course;
```

Courses

code	*name*
TDA357	Databases
TIN090	Algorithms

GivenCourses

course	*per*	*teacher*
TDA357	3	Niklas Broberg
TDA357	2	Graham Kemp
TIN090	1	Devdatt Dubhashi

$$\pi_{code,name,period}(\sigma_{teacher='Niklas\ Broberg'\ \&\ code\ =\ course}(Courses \times GivenCourses))$$

Example:

```
SELECT code, name, period
FROM Courses, GivenCourses
WHERE  teacher = 'Niklas Broberg'
       AND code = course;
```

code	*name*	*course*	*per*	*teacher*
TDA357	Databases	TDA357	3	Niklas Broberg
TDA357	Databases	TDA357	2	Graham Kemp
TDA357	Databases	TIN090	1	Devdatt Dubhashi
TIN090	Algorithms	TDA357	3	Niklas Broberg
TIN090	Algorithms	TDA357	2	Graham Kemp
TIN090	Algorithms	TIN090	1	Devdatt Dubhashi

$\pi_{code,name,period}(\sigma_{teacher='Niklas\ Broberg'\ \&\ code\ =\ course}(\textbf{Courses x GivenCourses}))$

Example:

```
SELECT code, name, period
FROM   Courses, GivenCourses
WHERE teacher = 'Niklas Broberg'
      AND code = course;
```

code	*name*	*course*	*per*	*Teacher*
TDA357	Databases	TDA357	3	Niklas Broberg
TDA357	Databases	TDA357	2	Graham Kemp
TDA357	Databa			
TIN090	Algorith			
TIN090	Algorithms	TDA357	2	Graham Kemp
TIN090	Algorithms	TIN090	1	Devdatt Dubhashi

code	*name*	*course*	*per*	*teacher*
TDA357	Databases	TDA357	3	Niklas Broberg

$$\Pi_{code,name,period}(\sigma_{teacher='Niklas\ Broberg'\ \&\ code=course}(\text{Courses x GivenCourses}))$$

Example:

```
SELECT code, name, period
FROM   Courses, GivenCourses
WHERE  teacher = 'Niklas Broberg'
       AND code = course;
```

code	*name*	*course*	*per*	*teacher*
TDA357	Databases	TDA357	3	Niklas Broberg

code	*name*	*per*
TDA357	Databases	3

$$\Pi_{code,name,period}(\sigma_{teacher='Niklas\ Broberg'\ \&\ code\ =\ course}(Courses \times GivenCourses))$$

Quiz!

What does the following relational algebra expression compute?

The expression is invalid, since the result after the projection will not have attributes teacher and course to test.

More complex expressions

- So far we have only examples of the same simple structure:

$$\pi_X(\sigma_C(T))$$

- We can of course combine the operands and operators of relational algebra in (almost) any way imaginable.

$$\sigma_C(R_3 \bowtie_D \pi_X(R_1 \times R_2))$$

```
SELECT *
FROM   R3 JOIN (SELECT X FROM R1,R2) ON D
WHERE  C
```

Subqueries

- Subqueries is a term referring to a query used inside another query:

```
SELECT teacher
FROM   GivenCourses NATURAL JOIN
         (SELECT course, period
          FROM   Lectures
          WHERE  weekday = 'Mon')
WHERE  period = 3;
```

- Beware the natural join!!
- ”List all teachers who have lectures on Mondays in period 3”
- SQL is a language where any query can be written in lots of different ways…

```
SELECT course, period
FROM   Lectures
WHERE  weekday = 'Mon'
```

course	*period*	*room*	*weekday*	*hour*
TDA357	3	HC1	Mon	13
TDA357	3	HC1	Thu	10
TDA357	2	VR	Tue	8
TDA357	2	HC1	Thu	13
TIN090	1	HA4	Mon	8
TIN090	1	HC3	Thu	13

```
SELECT course, period
FROM   Lectures
WHERE  weekday = 'Mon'
```

course	*period*	*room*	*weekday*	*hour*
TDA357	3	HC1	Mon	13
TIN090	1	HA4	Mon	8

```
SELECT teacher
FROM   GivenCourses NATURAL JOIN
         (SELECT course, period
          FROM   Lectures
          WHERE  weekday = 'Mon')
WHERE  period = 3;
```

course	*period*
TDA357	3
TIN090	1

course	*period*	*teacher*	*#students*
TDA357	3	Niklas Broberg	130
TDA357	2	Graham Kemp	135
TIN090	1	Devdatt Dubashi	95

```
SELECT teacher
FROM   GivenCourses NATURAL JOIN
          (SELECT course, period
           FROM   Lectures
           WHERE  weekday = 'Mon')
WHERE  period = 3;
```

course	*period*	*teacher*	*#students*
TDA357	3	Niklas Broberg	130
TIN090	1	Devdatt Dubashi	95

Result

teacher
Niklas Broberg

Renaming attributes

- Sometimes we want to give new names to attributes in the result of a query.
  - To better understand what the result models
  - In some cases, to simplify queries

```
SELECT *
FROM   Courses NATURAL JOIN
          (SELECT course AS code, period, teacher
           FROM   GivenCourses);
```

Renaming relations

- Name the result of a subquery to be able to refer to the attributes in it.
- Alias existing relations (tables) to make referring to it simpler, or to disambiguate.

```
SELECT L.course, weekday, hour, room
FROM   Lectures L, GivenCourses G, Rooms
WHERE  L.course = G.course
       AND L.period = G.period
       AND room = name
       AND nrSeats < nrStudents;
```

What does this query mean?

Renaming relations

- Name the result of a subquery to be able to refer to the attributes in it.
- Alias existing relations (tables) to make referring to it simpler, or to disambiguate.

```
SELECT L.course, weekday, hour, room
FROM   Lectures L, GivenCourses G, Rooms
WHERE  L.course = G.course
       AND L.period = G.period
       AND room = name
       AND nrSeats < nrStudents;
```

List all lectures that are scheduled in rooms with too few seats.

Renaming in Relational Algebra

- Renaming = Given a relation, give a new name to it, and (possibly) to its attributes

$$\rho_{A(X)}(R)$$

  - Rename R to A, and the attributes of R to the names specified by X (must match the number of attributes).
  - Leaving out X means attribute names stay the same.
  - Renaming the relation is only necessary for subqueries.
  - **ρ** = rho = greek letter **r** = **r**ename

Sequencing

- Easier to handle subqueries separately when queries become complicated.
  - Example: $\pi_X(R_1 \bowtie_C R_2)$ could be written as

$$R_3 := R_1 \times R_2$$
$$R_4 := \sigma_C(R_3)$$
$$R := \pi_X(R_4)$$

  - In SQL:

```
WITH
 R3 AS (SELECT * FROM R1, R2),
 R4 AS (SELECT * FROM R3 WHERE C)
SELECT X FROM R4;
```

- Example:

```
WITH DBLectures AS
   (SELECT room, hour, weekday
    FROM   Lectures
    WHERE  course = 'TDA357'
           AND period = 3)
SELECT weekday
FROM   DBLectures
WHERE  room = 'HC1';
```

What does this query mean?

- Example:

```
WITH DBLectures AS
   (SELECT room, hour, weekday
    FROM   Lectures
    WHERE  course = 'TDA357'
           AND period = 3)
SELECT weekday
FROM   DBLectures
WHERE  room = 'HC1';
```

Lists the days when the Databases course has lectures in room HC1 during period 3.

Creating views

- A *view* is a ”virtual table”, or ”persistent query” – a relation defined in the database using data contained in other tables.

```
CREATE VIEW viewname AS query
```

- For purposes of querying, a view works just like a table.
- Depending on your DBMS, a view can be read-only, or allow modifications to the underlying table.

Example:

```
CREATE VIEW DBLectures AS
 SELECT room, hour, weekday
 FROM   Lectures
 WHERE  course = 'TDA357'
        AND period = 3;


SELECT weekday
FROM   DBLectures
WHERE  room = 'HC1';
```

The WHERE clause

- Specify conditions *over rows*.
- Can involve
  - constants
  - attributes in the row
  - simple value functions (e.g. ABS, UPPER)
  - subqueries

- Lots of nice tests to make…

Testing for membership

- Test whether or not a tuple is a member of some relation.

```
tuple [NOT] IN subquery {or literal set}
```

```
SELECT course
FROM   GivenCourses
WHERE  period IN (1,4);
```

List all courses that take place in the first or fourth periods.

Quiz!

List all courses given by a teacher who also gives the Databases course (TDA357). (You must use IN…)

```
SELECT course
FROM   GivenCourses
WHERE  teacher IN
          (SELECT teacher
           FROM   GivenCourses
           WHERE  course = 'TDA357');
```

Testing for existance

- Test whether or not a relation is empty.

```
[NOT] EXISTS subquery
```

e.g. List all courses that have lectures.

```
SELECT code
FROM   Courses
WHERE  EXISTS
         (SELECT *
          FROM   Lectures
          WHERE  course = code);
```

Note that code is in scope here since it is an attribute in the row being tested in the outer ”WHERE” clause. This is called a correlated query.

Quiz!

List all courses that are not given in the third period. (You must use EXISTS…)

```
SELECT code
FROM   Courses
WHERE  NOT EXISTS
          (SELECT *
           FROM   GivenCourses
           WHERE  course = code
                  AND period = 3);
```