Database Usage (and Construction)

More SQL Queries and Relational Algebra

Tests on groups

- Aggregations can't be put in the WHERE clause
 – they're not functions on rows but on groups.
- Sometimes we want to perform tests on the result of an aggregation.
 - Example: List all teachers who have an average number of students of >100 in their courses.
- SQL allows us to put such tests in a special HAVING clause after GROUP BY.

Quiz!

List all teachers who have an average number of students of >100 in their courses.

SELECT teacher
FROM GivenCourses
GROUP BY teacher

HAVING AVG(nrStudents) > 100;

Example

SELECT teacher
FROM GivenCourses
GROUP BY teacher

HAVING AVG(nrStudents) > 100;

code	period	teacher	#students
TDA357	2	Niklas Broberg	130
TIN000	1	Devdatt Dubhashi	95
TDA357	4	Rogardt Heldal	135
TDA590	2	Rogardt Heldal	70

AVG(nrSt.) 130 95 102.5

Quiz!

- There is no correspondence in relational algebra to the HAVING clause of SQL. Why?
 - Because we can express it with an extra renaming and a selection. Example:

SELECT teacher
FROM GivenCourses
GROUP BY teacher

HAVING AVG(nrStudents) > 100;

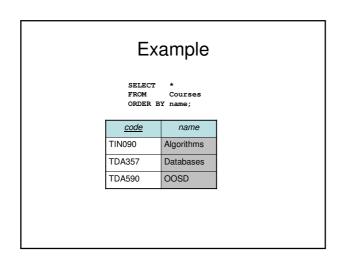
 $\pi_{\text{teacher}}(\sigma_{\text{avgSt}} > _{100}(\gamma_{\text{teacher}, \ \text{AVG(nrStudents)} \ \text{as avgSt}}(\text{GivenCourses})))$

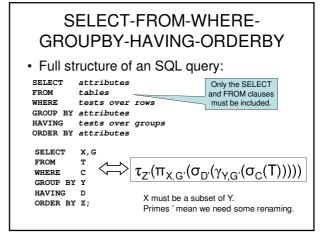
Sorting relations

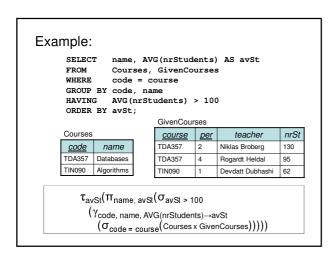
- · Relations are unordered by default.

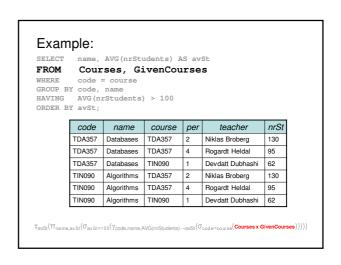
- Sort relation R on attributes X.

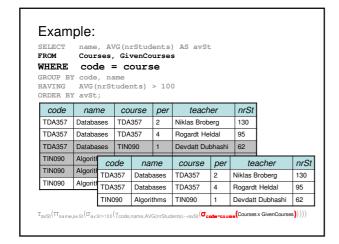
- Ordering only makes sense at the top level, or if only a given number of rows are sought, e.g. the top 5.
- Oracle: Use the implicit attribute rownum to limit how many rows should be used.
- τ = tau = greek letter t = sort (s is taken)

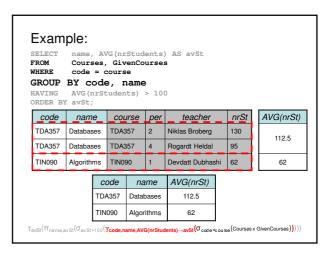


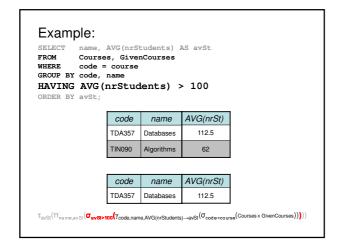


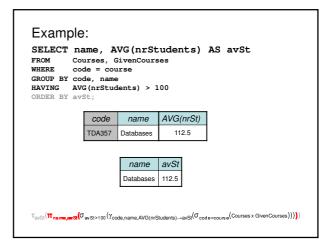


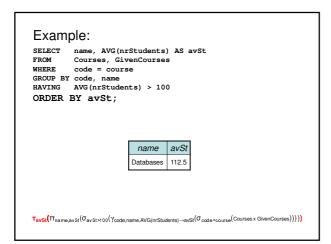












Relations as sets

- · Relations are sets of tuples.
- · Set theory has plenty to borrow from:
 - Some we've seen, like ∈ (IN).
 - More operators:
 - U (union)
 - ∩ (intersection)
 - \ (set difference)

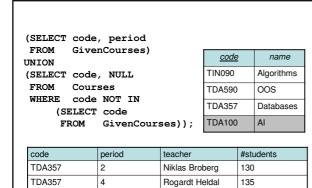
Set operations

- · Common set operations in SQL
 - UNION: Given two relations $\rm R_1$ and $\rm R_2,$ add them together to form one relation $\rm R_1$ U $\rm R_2.$
 - INTERSECT: Given two relations R_1 and R_2 , return all rows that appear in both of them, forming $R_1 \cap R_2$.
- All three operations require that R₁ and R₂ have (almost) the same schema.
 - Attribute names may vary, but number, order and types must be the same.

Quiz!

List all courses and the periods they are given in.

Courses that are not scheduled for any period should also be listed, but with NULL in the field for period. You must use a set operation.



Devdatt Dubhashi

Rogardt Heldal

95

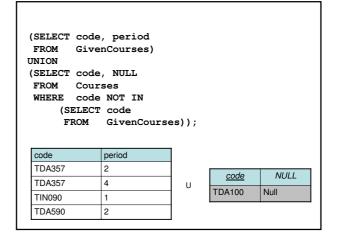
70

TIN090

TDA590

1

2



Result

code	period
TDA357	2
TDA357	4
TIN090	1
TDA590	2
TDA100	

Not sets but bags!

- In set theory, a set cannot contain duplicate values. Either a value is in the set, or it's not.
- In SQL, results of queries can contain the same tuples many times.
 - Done for efficiency, eliminating duplicates is costly.
- A set where duplicates may occur is called a *bag*, or *multiset*.

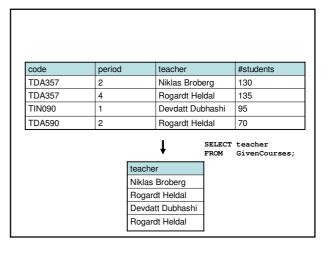
Controlling duplicates

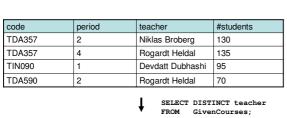
- Queries return bags by default. If it is important that no duplicates exist in the set, one can add the keyword DISTINCT.
 - Example:

SELECT DISTINCT teacher
FROM GivenCourses;

- DISTINCT can also be used with aggregation functions.
 - Example:

SELECT COUNT(DISTINCT teacher)
FROM GivenCourses;





	FROM	GivenCou	
teacher			
Niklas Brober	g		
Rogardt Held	al		
Devdatt Dubh	nashi		
•			

code	period	teacher	#students
TDA357	2	Niklas Broberg	130
TDA357	4	Rogardt Heldal	135
TIN090	1	Devdatt Dubhashi	95
TDA590	2	Rogardt Heldal	70
SELECT COUNT (teacher) FROM GivenCourses; SELECT COUNT(DISTINCT teacher) FROM GivenCourses;		COUNT(teacher) 4 COUNT(DISTINCT teacher) 3	

Duplicate elimination

 Duplicate elimination = Given relation R, remove all duplicate rows.

δ(R)

- Remove all duplicates from R.

SELECT DISTINCT X FROM R WHERE C;

 $\delta(\pi_X(\sigma_C(R)))$

• δ = delta = greek letter d = duplicate elimination

Retaining duplicates

- · Set operations eliminate duplicates by default.
 - For pragmatic reasons to compute either intersection or set difference efficiently, the relations need to be sorted, and then eliminating duplicates comes for free.
- If it is important that duplicates are considered, one can add the keyword ALL.

– Example:

Doesn't work in Oracle, there ALL only works for UNION.

(SELECT room
FROM Lectures)
EXCEPT ALL
(SELECT name
FROM Rooms);

All rooms appear once in Rooms. The set difference will remove each room once from the first set, thus leaving those rooms that have more than one lecture in them.

Summary – relations as sets

- · Set operations can be used on relations
 - Requires the operands to have the same arity (number of attributes) and types must match.
 - UNION
 - INTERSECT
 - EXCEPT (MINUS)
- Relations are treated as bags in most queries, but as sets in the result of a set operation.
 - To eliminate duplicates, use DISTINCT.
 - To retain duplicates for set operations, use ALL.

Common idiom

List all courses and the periods they are given in.

Courses that are not scheduled for any period should also be listed, but with NULL in the field for period. You must use a set operation.

(SELECT code, period
FROM Courses, GivenCourses that fit in the join,
UNION then union with
(SELECT code, NULL those that don't.
FROM Courses
WHERE code NOT IN
(SELECT course FROM GivenCourses));

Summary SQL and Relational Algebra

- SQL is based on relational algebra.
- Operations over relations
- SELECT-FROM-WHERE-GROUPBY-HAVING-ORDERBY
- Operations for:
 - Selection of rows (σ)
 - Projection of columns (π)
 - Combining tables

 - Cartesian product (x) Join, natural join, outer join (M_C, M, M)
- Grouping and aggregation
 - Grouping (γ)
 - SUM, AVG, MIN, MAX, COUNT
- Set operations
 - Union (U)
 - Intersect (n) • Set difference (\)
- Miscellaneous
 - Renaming (ρ) Duplicate elimination (δ)
 - Sorting (τ)
- Subqueries
 - Sequencing

 - (Views)

Course Objectives - Usage

When the course is through, you should

- Know how to query a database for relevant data using SQL
- Know how to change the contents of a database using SQL

"Add a course 'Databases' with course code 'TDA357', given by ...

"Give me all info regarding the course 'TDA357"

Exam – Relational Algebra

"Here is a schema for a database over persons and their employments....

- What does this relational-algebraic expression compute? ...
- Translate this relational-algebraic expression to SQL.
- Write a relational-algebraic expression that computes
- Translate this SQL query to a relational-algebraic expression.

Exam - SQL DML

"The grocery store wants your help in getting proper information from their database. ...

- Write a query that finds the total value of the entire inventory of the store.
- List all products with their current price, i.e. the discount price where such exists, otherwise the base price.

Database Construction (and Usage)

More on Modifications and Table Creation Assertions **Triggers**

Summary - Modifications

- · Modifying the contents of a database:
 - Insertions

INSERT INTO tablename VALUES tuple

- Deletions

DELETE FROM tablename WHERE test over rows

- Updates

UPDATE tablename

SET attribute = value

WHERE test over rows

Insertions with queries

 The values to be inserted could be taken from the result of a query:

```
INSERT INTO tablename (query)
- Example:
INSERT INTO GivenCourses
(SELECT course, period + 2, teacher, NULL
FROM GivenCourses
WHERE period <= 2);</pre>
```

All courses that are given in periods one and two are also scheduled to be given two periods later, with the same teacher.

Explicit attribute lists

- Attribute order could be given explicit when inserting.
 - Example:

```
INSERT INTO
GivenCourses(course, period, teacher, nrStudents)
(SELECT course, period + 2, teacher, NULL
FROM GivenCourses
WHERE period <= 2);</pre>
```

Perhaps the teacher and nrStudents attributes were listed in the other order in the definition of the table? Doesn't matter anymore since they are explicitly listed.

Quiz

What will the following insertion result in?

```
INSERT INTO
GivenCourses(course, period, teacher)
VALUES ('TDA357', 3, 'Niklas Broberg');
```

 Attribute lists can be partial. Any attributes not mentioned will be given the value a default value, which by default is NULL.

Default values

- · Attributes can be given default values.
 - Specified when a table is defined using the DEFAULT keyword.

- Default default value is NULL.

Insertion with default values

 Leaving out an attribute in an insertion with explicitly named attributes gives that row the default value for that attribute:

```
INSERT INTO
GivenCourses(course, period, teacher)
VALUES ('TDA357', 3, 'Niklas Broberg');
```

 When no attribute list is given, the same effect can be achieved using the DEFAULT keyword:

```
INSERT INTO GivenCourses
VALUES ('TDA357', 3, 'Niklas Broberg', DEFAULT);
```