IS5102
Database Management Systems
Lecture 9: Introduction to SQL

Alexander Konovalov
alexander.konovalov@st-andrews.ac.uk

(with thanks to Susmit Sarkar)

2021

Last time

Basic SQL

- Defining Table Structure
- Querying Tables

This week: More SQL

- Modifying Data
- Orderings and Aggregates
- Integrity Constraints
- Views and Authorisation

Modification of the Database

- Insertion of new tuples into a given relation
- Deletion of tuples from a given relation
- Updating of values in some tuples in a given relation

Deletion

- Delete all instructors

```
DELETE FROM instructor;
```

- Delete all instructors from the Finance department

```
DELETE FROM instructor
WHERE dept_id = 'FIN';
```

- Delete all tuples in the instructor relation for those instructors associated with a department located in the Bute building.

```
DELETE FROM instructor
WHERE dept_id IN
      (SELECT dept_id
         FROM department
        WHERE building = 'Bute');
```

- Add a new tuple to course

```
INSERT INTO course
VALUES ('IS5102', 'DBMS','CS', 15);
```

- or equivalently

```
INSERT INTO course (course_id, title, dept_id, credits)
VALUES ('IS5102', 'DBMS','CS', 15);
```

- Add a new tuple to student with `tot_creds` set to `null`

```
INSERT INTO student
VALUES ('65467', 'Emma', 'CS', NULL);
```

- Add all instructors to the student relation with `tot_creds` set to `0`

```
INSERT INTO student
SELECT instr_id, instr_name, dept_id, 0
  FROM instructor;
```

- The `SELECT FROM WHERE` statement is evaluated fully before any of its results are inserted into the relation. Otherwise queries like

```
INSERT INTO table1 SELECT * FROM table1;
```

would cause problems

Exercise: revert the previous update of `student`, relying on the ID of students and instructors being non-overlapping

Increase salaries of instructors whose salary is over £30,000 by 3%, and all others receive a 5% raise

Choice 1: Write two update statements:

```
UPDATE instructor
   SET salary = salary * 1.03
 WHERE salary > 30000;
UPDATE instructor
   SET salary = salary * 1.05
 WHERE salary <= 30000;
```

Case Statement for Conditional Updates

Choice 2: Same query as before but with case statement

```
UPDATE instructor
   SET salary =
       CASE
           WHEN salary <= 30000 THEN
               salary * 1.05
           ELSE
               salary * 1.03
       END;
```

NOTE: In MariaDB use

```
    END CASE
```

instead of

```
    END
```

SQL includes a string-matching operator for comparisons on character strings. The operator like uses patterns that are described using two special characters:

- percent (%): The % character matches any substring
- underscore (_): The _ character matches any character

- Find the names of all instructors whose name **includes the substring** "PH".

```
SELECT instr_name
  FROM instructor
 WHERE dept_id LIKE '%PH%';
```

- **Exercise**: check if the match is case-sensitive

- Match the string "100%"

```
LIKE '100\%' escape '\'
```

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

String Operations (Cont.)

- 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.

- SQL supports a variety of string operations such as
  - concatenation
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, etc.

- But standard syntax is not always supported!

Ordering the Display of Tuples

- List in alphabetic order the names of all instructors

```
SELECT DISTINCT instr_name
  FROM instructor
 ORDER BY instr_name;
```

- We may specify DESC for descending order or ASC for ascending order, for each attribute; ascending order is the default

```
SELECT DISTINCT instr_name, salary
  FROM instructor
 ORDER BY salary DESC;
```

- Can sort on multiple attributes

```
ORDER BY dept_name, name
```

Null Values

- It is possible for tuples to have a null value, denoted by `NULL`, for some of their attributes
- `NULL` signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving `NULL` is `NULL`

  Example: `5 + NULL` returns `NULL`
- Any comparison with `NULL` returns unknown

  Example: `5 < NULL` or `NULL <> NULL` or `NULL = NULL`
- The predicate `IS NULL` can be used to check for null values

  Example: Find all students with `tot_cred` is null.

```
SELECT name
  FROM student
 WHERE tot_cred IS NULL;
```

Aggregate Functions

These functions operate on the multi-set 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 examples

- Find the average salary of instructors in the Physics department

```
SELECT AVG (salary)
  FROM instructor
  WHERE dept_id = 'PHYS';
```

- Find the total number of instructors who teach at least one course

```
SELECT COUNT (DISTINCT instr_id)
  FROM teaches;
```

- Find the number of tuples in the course relation

```
SELECT COUNT (*)
  FROM course;
```

Find the average salary of instructors in each department

```
SELECT dept_id, AVG (salary) AS "Average Salary"
  FROM instructor
 GROUP BY dept_id;
```

Exercise: use `NATURAL JOIN` to output department name instead of ID

Attributes in select clause outside of aggregate functions must appear in group by list

The following does not produce meaningful result

```
SELECT dept_id, instr_name, AVG (salary)
  FROM instructor
 GROUP BY dept_id;
```

Find the names and average salaries of all departments whose average salary is greater than 30000

```
SELECT dept_id, AVG (salary)
  FROM instructor
 GROUP BY dept_id
HAVING AVG (salary) > 30000;
```

Note: predicates in the `HAVING` clause are applied **after** the formation of groups whereas predicates in the `WHERE` clause are applied **before** forming groups

Find the total sum of all annual salaries

```
SELECT SUM (salary)
  FROM instructor;
```

Above statement ignores NULL amounts

Result is NULL if there is no non-null amount

Exercise: find the total amount of salaries to be paid in a month

Reading and practice

- **[DBSC]** Chapters 4-5, Database System Concepts, Silberschatz, Korth and Sudarshan
- **[DBS]** Chapter 7, Database Systems, Connolly and Begg
- **[DBD]** Chapters 15-16, Database Design, Watt and Eng
- Useful sites
  - http://www.w3schools.com/sql/
  - http://sqlzoo.net/wiki/Main_Page