Basic SQL DML (no joins)

followup

CS383 Jan 31

Homework 1

- You created and distributed public and private keys.
- Now: ssh db

Local Machine		Remote machine
UNIX> ssh remote send ID along with initial ping (clear)		
		find public key from local using ID Encrypt something using public key
Receive encrypted thing, decrypt re-encrypt (plus some extra) using private key		
		decrypt -- check if is what you sent (plus) Yes: generate a key for a symmetric algorithm encrypt symm key using public NO: send back NO
receive response: if yes, decrypt symm key		
Send and receive material using symm key		Send and receive material using symm key

Another look at how select works

- You can think of select as defining, building and refining a relation based on other relations.
- After each stage of select, there is a valid relation
  - Unlikely that this is actually how the query executed internally ... Why? Do I care?

```
SELECT selection_list  # Define what the columns in the relation will be

FROM table_list        # fill in the columns from the listed tables
                       # Does cross product if there are multiple tables

WHERE constraint       # Select the rows in the temp table after FROM completes
                       # such that the rows match the given constraint

GROUP BY columns       # groups the remaining rows by the given columns
    HAVING group constraints      # select the grouped rows by the constraint

ORDER BY sorting_cols  # Order the remaining rows by the given columns

LIMIT count;           # Limit on results
```

Using alaised columns

in the univ database

- Recall aliased columns
  - select salary as s, name from instructor;
  - ~~select salary as s, name from instructor where s>100000~~; /// DOES NOT WORK!!!!
    - cannot use alias within the query, at least in this way.
  - select salary as s, name from instructor where salary>100000;

  - aliases in the output relations are not available, Aliases in subrelations are
    - select salary, name from instructor, (select avg(salary) as av from instructor) as ie where salary>ie.av;
    - select salary, name from instructor where salary> (select avg(salary) as av from instructor);
    - select * from (select salary as s, name from instructor) as inn, (select avg(salary) as av from instructor) as ie where inn.s>ie.av;

Formatting Numbers

and column names

- select s as "hello kitty", name,av::numeric(8,2) as "the average" from (select salary as s, name from instructor) as inn, (select avg(salary) as av from instructor) as ie where inn.s>ie.av;
  - for reals, cast into numeric and specify format
  - In column name alias use double quotes to get spaces
    - Postgres: double quotes only on top level column names

Grouping problems

- get list of all people who earn max salary in their department

  select max(salary), dept_name, name from instructor group by dept_name;
  ERROR: column "instructor.name" must appear in the GROUP BY clause or be used in an aggregate function
  LINE 1: select avg(salary), dept_name, name from instructor group by...

  - this query does not work
    - WHY?

Grouping 2

temporary relations the WITH clause

- get list of all people who earn max salary in their department
- with aaa as (select max(salary) as maxx, dept_name from instructor group by dept_name) select * from aaa, instructor where aaa.dept_name=instructor.dept_name and salary=maxx;

- With is often unneeded
  - select * from (select max(salary) as maxx, dept_name from instructor group by dept_name) as aaa, instructor where aaa.dept_name=instructor.dept_name and salary=maxx;
- The more complex the query the more I like with

With and Having

Or having is just feels weird ...

- Show all departments whose average salary is greater than the university average
- University average: select avg(salary) from instructor;
- select avg(salary), dept_name from instructor group by dept_name having avg(salary) > (select avg(salary) from instructor);

- Use with to avoid having
  - alternately, to precisely explain having.
- with aaa as (select avg(salary) as avg, dept_name from instructor group by dept_name) select * from aaa where avg>(select avg(salary) from instructor);

From the sakila DB

- Find the names of all movies in which an actor with the last name GUINESS appeared. Show the first and last names of the actor.
  - You will need the movie, actor and film_actor relations

- Find the names of all movies in which an actor with the last name GUINESS appeared with an actor last name DAVIS. (show first and last names of both actors
  - 2 versions
    - A: NOT using Join
    - B: using join
  - You will need the movie, actor and film_actor relations