Introduction to SQL

Dr. Shrutilipi Bhattacharjee, Assistant Professor, Dept. of IT, NIT Karnataka, India

Subqueries in the *From* Clause

- SQL allows a subquery expression to be used in the **from** clause

- Find the average instructors' salaries of those departments where the average salary is greater than $42,000"

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

- Note that we do not need to use the **having** clause

- Another way to write above query

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

With Clause

- The **with** clause provides a way of defining a temporary relation whose definition is available only to the query in which the **with** clause occurs

- Find all departments with the maximum budget

```
with max_budget (value) as
        (select max(budget)
        from department)
select department.name
from department, max_budget
where department.budget = max_budget.value;
```

Complex Queries using *With* Clause

- Find all departments where the total salary is greater than the average of the total salary at all departments

```
with dept _total (dept_name, value) as
        (select dept_name, sum(salary)
        from instructor
        group by dept_name),
dept_total_avg(value) as
        (select avg(value)
        from dept_total)
select dept_name
from dept_total, dept_total_avg
where dept_total.value > dept_total_avg.value;
```

Scalar Subquery

- Scalar subquery is one which is used where a single value is expected
- List all departments along with the number of instructors in each department

```
select dept_name,
(select count(*)
from instructor
where department.dept_name = instructor.dept_name)
as num_instructors
from department;
```

- Runtime error if subquery returns more than one result tuple

Modification of the Database

- Deletion of tuples from a given relation
- Insertion of new tuples into 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_name* = 'Finance';

- Example: Delete all tuples in the instructor relation for those instructors associated with a department located in the Watson building

delete from *instructor*
where *dept name* **in** (**select** *dept name*
from *department*
where *building* = 'Watson');

Deletion

- Delete all instructors whose salary is less than the average salary of instructors

  delete from *instructor*
  where *salary* < (**select avg**(*salary*)
  from *instructor*);

- Problem: As we delete tuples from *instructor*, the average salary changes

- Solution used in SQL:
  - First, compute **avg** (salary) and find all tuples to delete
  - Next, delete all tuples found above (without recomputing **avg** or retesting the tuples)

Insertion

- Add a new tuple to *course*

  insert into *course*
  values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);

- Or equivalently

  insert into *course* (*course_id*, *title*, *dept_name*, *credits*)
  values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);

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

  insert into *student*
  values ('3003', 'Green', 'Finance', *null*);

Insertion

- Add all instructors to the ***student*** relation with tot_cred set to 0

  insert into *student*
  select *ID, name, dept_name, 0*
  from *instructor*

- The **select from where** statement is evaluated fully before any of its results are inserted into the relation

- Otherwise queries like the following would cause problem

  insert into *table*1 **select** * **from** *table*1

Updates

- Give a 5% salary raise to all instructors

```
update instructor
set salary = salary * 1.05
```

- Give a 5% salary raise to those instructors who earn less than 70000

```
update instructor
set salary = salary * 1.05
where salary < 70000;
```

- Give a 5% salary raise to instructors whose salary is less than average

```
set salary = salary * 1.05
where salary < (select avg (salary)
                from instructor);
```

Updates

- Increase salaries of instructors whose salary is over $100,000 by 3%, and all others by a 5%

- Write two **update** statements:

```
update instructor
    set salary = salary * 1.03
    where salary > 100000;
update instructor
    set salary = salary * 1.05
    where salary <= 100000;
```

- The order is important

- Can be done better using the **case** statement

Case Statement for Conditional Updates

- Same query as before but with case statement

```
update instructor
set salary = case
        when salary <= 100000 then salary * 1.05
        else salary * 1.03
        end
```

Updates with Scalar Subqueries

- Recompute and update tot_creds value for all students

```
update student S
set tot_cred = (select sum(credits)
        from takes, course
        where takes.course_id = course.course_id and
                S.ID = takes.ID and
                takes.grade <> 'F' and
                takes.grade is not null);
```

- Sets *tot_creds* to null for students who have not taken any course
- Instead of **sum**(*credits*), use:

```
case
        when sum(credits) is not null then sum(credits)
        else 0
 end
```

Next Lecture

Intermediate SQL

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in

© Silberschatz, Korth and Sudarshan