Principles of Database Systems

Introduction to SQL(5)-嵌套子查询

Nested Subqueries
(嵌套子查询)

Nested Subqueries

- SQL provides a mechanism for the nesting of subqueries.
- A **subquery** is a select-from-where expression that is nested within another query. (子查询是嵌套在另一个查询中的select-from-where表达式)
- A common use of subqueries is to perform tests for **set membership**, **set comparisons**, **and set cardinality**. (子查询通常被用来对集合成员资格、集合的比较以及集合的基数进行检查)

Set Membership(集合的成员资格)

- SQL allows testing tuples for membership in a relation. The **in** connective tests for set membership, where the set is a collection of values produced by a **select clause**. The **not in** connective tests for the absence of set membership. (SQL允许测试元组在关系中的成员资格。连接词in测试元组是否是集合中的成员，集合是由select子句产生的一组值构成的。连接词not in则测试元组是否不是集合中的成员)

Set Membership

- The **in** and **not in** operators can also be used on enumerated(枚举) sets. (in和not in操作符也能用于枚举集合)

```
select distinct name
from instructor
where name not in ('Mozart', 'Einstein');
```

Set Membership

- "Find all the courses taught in the both the Fall 2009 and Spring 2010 semesters."

- Step 1

  (**select** course id
  from section
  where semester = 'Spring' **and** year= 2010)

- Step 2

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
        course_id in (select course_id
                      from section
                      where semester = 'Spring' and year= 2010);
```

Set Membership

- "Find all the courses taught in the both the Fall 2009 but not in Spring 2010 semesters."

```
select distinct course_id
from section
where semester = 'Fall'
      and year= 2009
      and course_id  not in (select course_id
                             from section
                             where semester = 'Spring'
                                and year= 2010);
```

Set Comparison(集合的比较)

- Recall the query “Find names of instructors with salary greater than that of some (at least one) instructor in the Biology ”

```
select distinct T.name
from instructor as T, instructor as S
where T.salary > S.salary and S.dept_name='Biology';
```

Set Comparison

- Alternative style for the query

```
select name
from instructor
where salary > some (select salary
                     from instructor
                     where dept_name='Biology');
```

Note: The keyword any is synonymous(同义的) to some in SQL

Definition of Some Clause

- F <comp> **some** $r \Leftrightarrow \exists\, t \in r$ such that (F <comp> t)
  Where <comp> can be: $<,\ \leq,\ >,\ =,\ \neq$

(5 < **some** | 0 / 5 / 6 |) = true (read: 5 < some tuple in the relation)

(5 < **some** | 0 / 5 |) = false

(5 = **some** | 0 / 5 |) = true

(5 ≠ **some** | 0 / 5 |) = true (since 0 ≠ 5)

(= **some**) ≡ **in**
However, (≠ **some**) ≢ **not in**

Set Comparison

- Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.

```
select name
from instructor
where salary > all (select salary
                    from instructor
                    where dept_name = 'Biology');
```

Definition of all Clause

- F <comp> **all** $r \Leftrightarrow \forall\ t \in r$ (F <comp> t)

(5 < **all** | 0, 5, 6 |) = false

(5 < **all** | 6, 10 |) = true

(5 = **all** | 4, 5 |) = false

(5 ≠ **all** | 4, 6 |) = true (since 5 ≠ 4 and 5 ≠ 6)

(≠ **all**) ≡ **not in**

However, (= **all**) ≢ **in**

Set Comparison

- In many cases, using subqueries with **some**, **any** or **all** is logical equivalent to equality comparison which uses the subqueries containing certain aggregation.(在很多情况下，使用some,any或all的子查询逻辑等价于对使用聚合的子查询进行相等比较)

Test for Empty Relations

- Yet another way of specifying the query "Find all courses taught in both the Fall 2009 semester and in the Spring 2010 semester"

```
select course_id
from section as S
where semester = 'Fall' and year= 2009 and
        exists (select *
                  from section as T
                 where semester = 'Spring' and year= 2010
                       and S.course_id= T.course_id);
```

- **Correlated subquery相关子查询**

Correlated Subquery

- A subquery that uses a correlation name from an **outer query** is called a **correlated subquery**

- Observe and explain the following query:

```
select ID,name
from student
where (select COUNT(*)
            from takes
            where takes.ID=student.ID
            group by takes.ID)>2;
```

例题

- 查询没有选数据库课（CS-347）的学生名字。
  （使用exists/not exists）

classroom(<u>*building*</u>, <u>*room_number*</u>, *capacity*)
department(<u>*dept_name*</u>, *building*, *budget*)
course(<u>*course_id*</u>, *title*, *dept_name*, *credits*)
instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)
section(<u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>, *building*, *room_number*, *time_slot_id*)
teaches(<u>*ID*</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)
student(<u>*ID*</u>, *name*, *dept_name*, *tot_cred*)
takes(<u>*ID*</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>, *grade*)
advisor(<u>*s_ID*</u>, *i_ID*)
time_slot(<u>*time_slot_id*</u>, <u>*day*</u>, <u>*start_time*</u>, *end_time*)
prereq(<u>*course_id*</u>, <u>*prereq_id*</u>)

Not Exists

- Find all students who have taken all courses offered in the Biology department.

- Note that $X - Y = \emptyset \Leftrightarrow X \subseteq Y$
- *Note:* Cannot write this query using = **all** and its variants

Not Exists

- Find all students who have taken all courses offered in the Biology department.

```
select distinct S.ID, S.name
from student as S
where not exists ( (select course_id
                    from course
                    where dept_name = 'Biology')
                   except
                   (select T.course_id
                    from takes as T
                    where S.ID = T.ID) );
```

Section Review

- Nested Subqueries
  - set membership
    - in , not in
  - set comparisons
    - some, any, all
  - Test for Empty Relations
    - exists, not exists
    - Correlated subquery