SQL Programming 4

- Understanding the SQL Syntax
- Problem solving using SQL
    - –Formulating queries step by step
- Debugging using Test data

SQL Query Components

```
SELECT attributes to output
FROM  tables to scan
WHERE logical expression targeting tuples
GROUP BY group attributes
HAVING logical expression targeting groups
```

- The FROM clause can be multiple tables (join)
- The FROM and WHERE clause can involve subqueries.

Understanding SQL Syntax

- Strings and identifiers (variables)
  - Single and double quotes.
- Join multiple relations
  - Distinguishing attributes
- Subqueries
  - In brackets (….)
  - outer relation attributes to output
  - Variable scope in the subquery

Understanding SQL Syntax

So, you have learnt SQL programming. Given a relation R(A, B), are the following queries equivalent?

```
select *
from R
where a =‘b’;
```

```
select *
from R
where a =‘B’;
```

```
select *
from R
where a =b;
```

```
select *
from R
where A =B;
```

```
select *
from R
where a =“b”;
```

```
select *
from R
where A =“B”;
```

R

A	B
a	b
B	a
B	B
b	a
A	B

SQL Syntax: Join

- Is there anything wrong with the following queries?

```
select  mvID, genre
from Movie, Classification
Where Movie.mvID=Classfication.mvID;
```

```
select movie.mvID, director
from Movie NATURAL JOIN Direct;
```

```
select movie.mvID, director
from Movie JOIN Direct ON movie.mvID=Direct.mvID;
```

Subqueries: In Brackets (…)

- Subqueries must be enclosed in brackets.
- A subquery generally returns a set of tuples.
- Is there anything wrong with the following queries?

```
SELECT *
FROM Movie
WHERE mvID in (select *
   from Classification)
```

```
SELECT *
FROM Movie
WHERE length > (select length
   from Classification)
```

Subqueries: outer relation attributes to output

- What’s wrong with the following query?

```
select mvID, title, director
from movie
where mvID in (select mvID
        from direct)
```

Subqueries: Variable Scope

What are the output of the following queries?

```
select mvID, title
from movie
where rating in (select rating
        from movie
        where movie.mvID != mvID)
```

```
select mvID
from movie M
where  rating in (select  rating
        from movie
        where mvID != M.mvID)
```

```
select mvID
from movie M
where  rating in (select  rating
        from movie
        where movie.mvID != M.mvID)
```

Problem Solving Using SQL

- Formulating a complex query step by step.
  - Which tables have the required information?
  - How many scans of a table (loops over tuples in the table)?
  - Putting things together
    - Join or Subquery
  - Test initial solution with sample data and debug

Problem 1

Which movies are produced in the same studio?

- The Movie table has the production studio information.
- Two scans of Movie are needed
  - Each scan gives the studio information for one movie.
  - Compare the studio information for each movie.
- Try on sample data and debug.

Problem 1…

Movie.

```
 MVID TITLE                      RA  Rel_Date   LENGTH STUDIO
---------- -------------------- ---- ------ ------------ --------------------------
         1 Angels and Demons    M   14-05-2009       138 Sony Pictures
         2 Coco Avant Chanel    PG  25-06-2009       108 Roadshow
         3 Harry Potter 6       M   15-07-2009       153 Roadshow
         5 Ice Age 3            PG  01-07-2009        94 20th Century Fox
         6 The Da Vinci Code    M   18-05-2006
```

```
select m1.mvid, m2.mvid
from movie m1, movie m2
where m1.studio=m2.studio
```

```
  MVID       MVID
---------- ----------
         1          1
         3          2
         2          2
         3          3
         2          3
         5          5
```

So each movie is made in the same studio with itself?!
This is not desirable!

Problem 1 Solution

```
select m1.mvid, m2.mvid
from movie m1, movie m2
where m1.studio = m2.studio
  and m1.mvid != m2.mvid
```

```
 MVID       MVID
---------- ----------
         3          2
         2          3
```

Each pair of movies with the same studio are repeated?! This is not desirable!

The final solution:

```
select m1.mvid, m2.mvid
from movie m1, movie m2
where m1.studio = m2.studio
  and m1.mvid < m2.mvid
```

```
 MVID       MVID
---------- ----------
         2          3
```

Problem 2

- What are the movies that have at least two directors? Return the mvID and title of these movies.
  - Find these movies (mvID) first from the Direct table.
  - Output the mvID and title – Join with the Movie table.

Step by step …

- Find the movies (mvID) that have at least two directors from the Direct table.

```
select mvID
from direct
group by mvID
having count(director) >=2
```

```
   MVID
----------
     5
```

Solution 1: Subquery

- Find the title of these movies from the Movie table … using a subquery.

```
select mvID,title
from Movie
where mvID in (
        select mvID
        from direct
        group by mvID
        having count(director) >=2)
```

Solution 2: Join

- Find the title of these movies from the Movie table … using Join – more difficult.

```
select Movie.mvID, title
from Movie, (select mvID
        from direct
        group by mvID
        having count(director) >=2) M1
where Movie.mvID = M1.mvID
```

Debugging: Using Test Data

- Debugging queries using a given database instance.
  - Focusing on logic in the WHERE clause.
  - Test a query from different angles.
  - Make use of “SELECT *”.
- A database instance is only one collection of test data. A query that produce the correct output on the current database instance may not guarantee the query is logically correct.
  - Create marginal data to test SQL queries.

Problem 3

- Which movies (find the mvID) do not have any genre classification that is the same as the movie with mvID=1?
- A draft query is given below. Is it correct?

```
select C2.mvID
from classification C1, classification C2
where C1.genre != C2.genre  and C1.mvID=1
```

Problem 3 …

- Run the query on the test database instance.

Classification

```
MVID GENRE
--------------- ----------
      1 Drama
      2 Drama
      3 Action
      3 Adventure
      3 Drama
      4 Comedy
      5 Animated
      5 Comedy
```

```
select C2.mvID
from classification C1, classification C2
where C1.genre != C2.genre and C1.mvID=1
```

```
   MVID
-------------
      3
      3
      4
      5
      5
```

Movie #1 is classified as Drama, but Movie #3 is classified as Action, Adventure and Drama. So the query is wrong!

Problem 3 …

- Using SELECT *:
  - Check the output from “select *” with the same Where clause can reveal the underlying logic of the SQL for selecting rows.

```
select *
from classification C1, classification C2
where C1.genre != C2.genre and C1.mvID=1
```

```
 MVID GENRE             MVID GENRE
---------- ---------- ---------- ----------
         1 Drama               3 Action
         1 Drama               3 Adventure
         1 Drama               4 Comedy
         1 Drama               5 Animated
         1 Drama               5 Comedy
```

Problem 3 ...

- Where does the logic go wrong?

```
select *
from classification C1, classification C2
where C1.genre != C2.genre
    and C1.mvID=1
```

```
 MVID GENRE      MVID GENRE
---------- ---------- ---------- ----------
         1 Drama          3 Action
         1 Drama          3 Adventure
         1 Drama          4 Comedy
         1 Drama          5 Animated
         1 Drama          5 Comedy
```

This condition only checks that there exists a genre (of a movie) that is different from Drama, but not all genres of the movie --- so a movie that has a genre that is not Drama and a genre that is Drama will be in the output.

Problem 3 Solution

- If the set of all genres for a movie classification is different from the set of all genres for the classification of Movie#1, the movie should be output.
  - Set operation!

The set of mvIDs that have at least one classification that is the same as that of Movie#1.

```
select mvID
from classification
minus
select C2.mvID
from classification C1, classification C2
where C1.genre = C2.genre
          and C1.mvID=1
```

```
     MVID
-------------
        4
        5
```

Problem 4

- Find movies with mvID>3 with extremely long (>180 minutes) or short (<100 minutes) length.
- The following query seem produce correct result on the current Movie table instance. But is it logically correct?

```
select *
from movie
where mvID >3 and length <100 or length >180
```

Exercise

- Find the genre that has the largest number of movies. Return the genre and the total number of movies in the genre.
- The following query does not work – the genre with the largest number of movies can not be output.

```
select max(count(mvID))
from classification
group by genre
```

Exercise

- Solution hint: use a subquery.