RMIT Classification: Trusted

SQL Programming 4

Sections to read:

-”The Database language SQL” chapter
- SQL*Plus tutorial by R. Holowczak in the ”Oracle” page on Canvas.

RMIT Classification: Trusted

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.

RMIT Classification: Trusted

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
- Group-by and aggregation

RMIT Classification: Trusted

Understanding SQL Syntax

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

```
Q1: select *
from R
where a ='b';
```

```
Q2: select *
from R
where a ='B';
```

```
Q3: select *
from R
where a =b;
```

```
Q4: select *
from R
where A =B;
```

```
Q5: select *
from R
where a ="b";
```

```
Q6: select *
from R
where A ="B";
```

R

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

RMIT Classification: Trusted

SQL Syntax: Join

- Is there anything wrong with the following queries?

```
select  mvID, director
from Movie, Direct
Where Movie.mvID=Direct.mvID;
```

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

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

RMIT Classification: Trusted

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 Movie)
```

RMIT Classification: Trusted

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)
```

RMIT Classification: Trusted

Group-by and aggregation output

- With the GROUP BY operator, logically only attributes on the group-by list and their dependent attributes, as well as aggregates should be output.

```
select title, count(director)
from  movie, direct
where movie.mvid=direct.mvid
group by movie.mvid;
```

RMIT Classification: Trusted

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 rows in the table)?
    - Join or Subquery
  - Test initial solution with sample data and debug

RMIT Classification: Trusted

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.

RMIT Classification: Trusted

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!

RMIT Classification: Trusted

Problem 1 Solution

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

$\Rightarrow$

```
 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.mvid < m2.mvid
  and m1.studio = m2.studio
```

$\Rightarrow$

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

RMIT Classification: Trusted

Problem 2

- Which movies 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.

RMIT Classification: Trusted

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
```

RMIT Classification: Trusted

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)
```

RMIT Classification: Trusted

Solution 2: Join

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

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

RMIT Classification: Trusted

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 produces the correct output on the current database instance may not guarantee the query is logically correct.
  - Create marginal data to test SQL queries.

RMIT Classification: Trusted

Problem 3

- List actors having not starred in any movies with Tom Hanks?
- A draft query is given below. Is it correct?

```
select C2.actor
from Cast C1, Cast C2
where C1.actor='Tom Hanks'
   and C1.mvID!=C2.mvID
   and C2.actor !='Tom Hanks';
```

RMIT Classification: Trusted

Problem 3 …

- Running the query on the current Cast table seems to return the correct result.
- But the content in the Cast table fails to represent that an actor can appear in several movies and a movie can have several actors:
  - Every actor, including Tom Hanks only appears, in one movie;
  - No one appears in the same movie with Tom Hanks.

RMIT Classification: Trusted

Problem 3 …

- So update the Cast table as follows:

```
insert into Cast values (5, ‘Tom Hanks’);
insert into Cast values(1, ‘Audrey Tautou’);
```

- The query is test on the Cast table again and is shown to fail.

RMIT Classification: Trusted

Problem 3 ...

- An actor can appear in a set of movies. If any movie from this set is in the set of movies for Tom Hanks, then the actor should NOT be in the query result.

```
select actor
from Cast
EXCEPT
select actor
from Cast where mvID in
    (select mvID
     from Cast
     where actor = 'Tom Hanks')
```

RMIT Classification: Trusted

Efficiency of SQL queries

- Join queries and subqueries are time-consuming operations.
  - Avoid unnecessary joins or subqueries
- Select from a big table is time-consuming.
  - Create an index on big tables to speed up search of tables.

RMIT Classification: Trusted

Example: “List genres for movies”.

Inefficient approaches:

```
select genre
from classification
group by genre;
```

```
select distinct genre
from movie, classification
where movie.mvid=classification.mvid;
```

The correct approach:

```
select distinct genre
from classification;
```

RMIT Classification: Trusted

Example: Find movies that have number of actors greater than the average. Output the title of these movies.

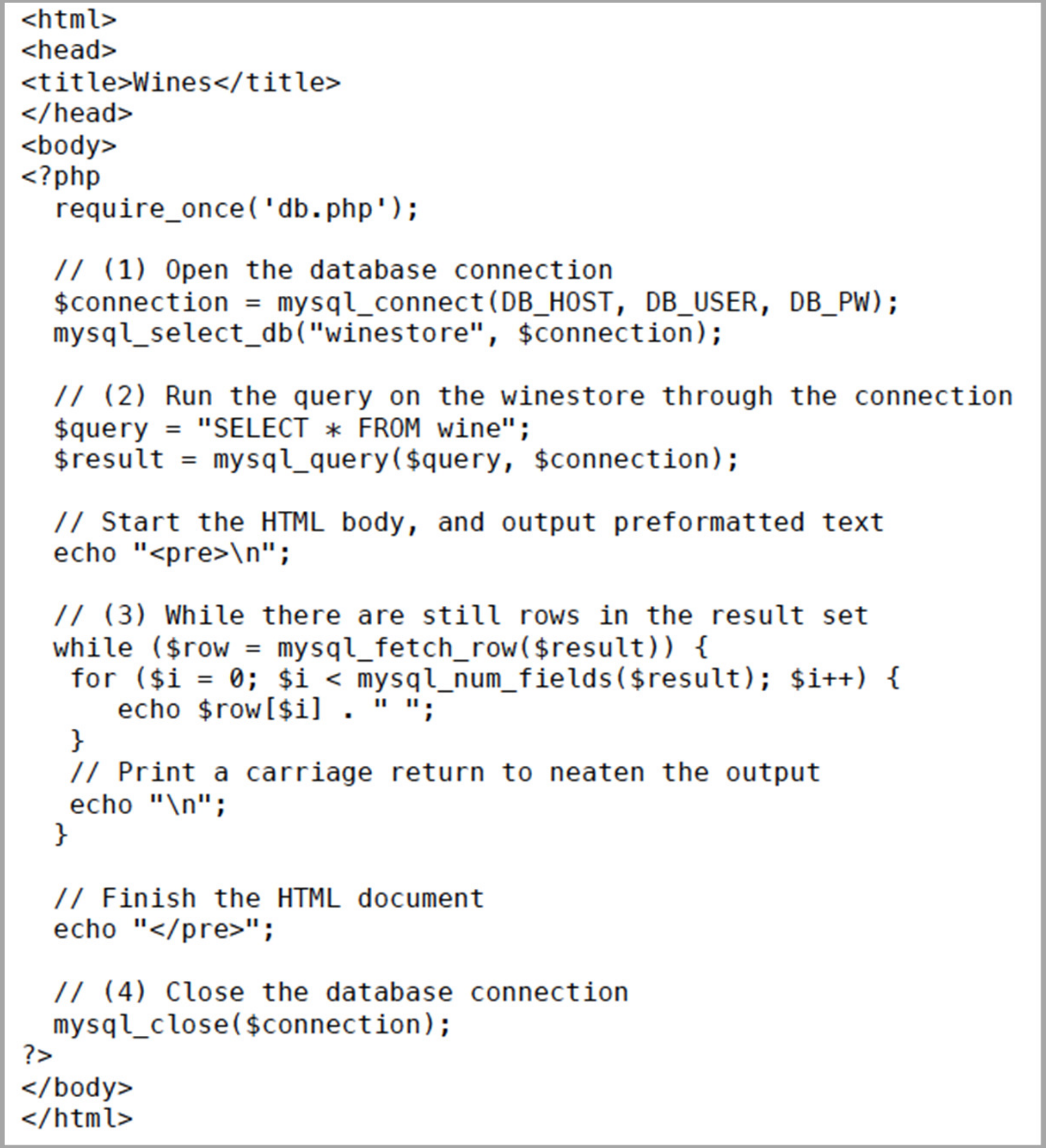
RMIT Classification: Trusted

Real SQL in applications

- We have seen only how SQL is used at the generic query interface --- an environment where we sit at a terminal and ask queries of a database.
- Reality is almost always different in real applications:
  - Conventional programs in a host language interacting via a DB library in SQL. Examples include Java + JDBC and PHP + PEAR/DB

RMIT Classification: Trusted

SQL embedded in PHP: example



```
<html>
<head>
<title>Wines</title>
</head>
<body>
<?php
  require_once('db.php');

  // (1) Open the database connection
  $connection = mysql_connect(DB_HOST, DB_USER, DB_PW);
  mysql_select_db("winestore", $connection);

  // (2) Run the query on the winestore through the connection
  $query = "SELECT * FROM wine";
  $result = mysql_query($query, $connection);

  // Start the HTML body, and output preformatted text
  echo "<pre>\n";

  // (3) While there are still rows in the result set
  while ($row = mysql_fetch_row($result)) {
   for ($i = 0; $i < mysql_num_fields($result); $i++) {
      echo $row[$i] . " ";
   }
   // Print a carriage return to neaten the output
   echo "\n";
  }

  // Finish the HTML document
  echo "</pre>";

  // (4) Close the database connection
  mysql_close($connection);
?>
</body>
</html>
```