Kevin Hance
DBMS (CPSC 321)
10/6/2019
HW5

Reading Assignment:

1. Procedural vs. non-procedural query languages:
    a. Procedural query languages involve the user instructing the system goes through a list of commands and operations, to come up with the final result. These languages often involve writing code (or a “procedure”) that the system executes on the database. This can be rather time-consuming compared to the process required to do the same operations in a non-procedural query language, but it’s much far more efficient. Examples of these languages include COBOL, BASIC, Assembler, and ALGOL.
    b. Non-procedural query languages involve more of the “what” rather than the “how”. This makes writing code much easier, as the user deals with fewer low-level details of the system. However, this can lead to code being less efficient, as the system follows a set of predefined rules for reading and executing code, giving the user less control over memory management and execution time. Examples of non-procedural languages include PROLOG, LISP, and SQL.
    c. The description of the non-procedural language best fits SQL in my opinion, as the code we’ve been working with has largely focused on the “what” rather than the “how”. While we do some of the “how” by defining attribute types and size (e.g. VARCHAR(20)), we leave the processes of iterating through the data, searching for and comparing values up to the system. In addition, it was mentioned in the textbook that non-procedural languages deal more with higher-level operations, and that’s what we’ve been working with in SQL thus far.
2. The cartesian product of two tuples, or rows of values, returns the product of the two sets. In other words, if we did a cartesian join with two tuples (a, b, c) and (d, e, f), we would get (a, b, c, d, e, f) as the result. In the context of SQL, this is used when using the SELECT command. For example, if we were to execute a command such as:

    ```
    SELECT name, account_type, balance FROM accounts, orders;
    ```

    Then we would be performing a cartesian join on name, account_type, and balance.

3.

```sql
DROP TABLE IF EXISTS table1;
DROP TABLE IF EXISTS table2;

CREATE TABLE table2(
    company_id INT UNSIGNED,
    company_name VARCHAR(40),
    company_city VARCHAR(40),
    PRIMARY KEY(company_id)
);

CREATE TABLE table1(
    item_id INT UNSIGNED,
    item_name VARCHAR(30),
    item_price FLOAT,
    company_id INT UNSIGNED,
    FOREIGN KEY (company_id) REFERENCES table2 (company_id),
    PRIMARY KEY (item_id)
);

INSERT INTO table2 VALUES (1001, 'Apple', 'San Francisco');
INSERT INTO table2 VALUES (1002, 'Microsoft', 'Redmond');
INSERT INTO table2 VALUES (1003, 'Uber', 'Los Angeles');
INSERT INTO table2 VALUES (1004, 'Lyft', 'Chicago');

INSERT INTO table1 VALUES (10001, 'Black Airpods', 129.99, 1001);
INSERT INTO table1 VALUES (10024, 'Windows 10 Professional', 149.99, 1002);
INSERT INTO table1 VALUES (10157, 'Uber T-shirt', 19.95, 1003);
INSERT INTO table1 VALUES (10402, 'Lyft Baseball Cap', 21.99, 1004);

SELECT * FROM table1;
SELECT * FROM table2;

SELECT * FROM table1 NATURAL JOIN table2;
```

```
mysql>
mysql> SELECT * FROM table1;
S+---------+-------------------------+------------+------------+
| item_id | item_name               | item_price | company_id |
+---------+-------------------------+------------+------------+
|   10001 | Black Airpods           |     129.99 |       1001 |
|   10024 | Windows 10 Professional |     149.99 |       1002 |
|   10157 | Uber T-shirt            |      19.95 |       1003 |
|   10402 | Lyft Baseball Cap       |      21.99 |       1004 |
+---------+-------------------------+------------+------------+
4 rows in set (0.01 sec)

mysql> SELECT * FROM table2;
+------------+--------------+---------------+
| company_id | company_name | company_city  |
+------------+--------------+---------------+
|       1001 | Apple        | San Francisco |
|       1002 | Microsoft    | Redmond       |
|       1003 | Uber         | Los Angeles   |
|       1004 | Lyft         | Chicago       |
+------------+--------------+---------------+
4 rows in set (0.00 sec)

mysql>
mysql> SELECT * FROM table1 NATURAL JOIN table2;
+------------+---------+-------------------------+------------+--------------+---------------+
| company_id | item_id | item_name               | item_price | company_name | company_city  |
+------------+---------+-------------------------+------------+--------------+---------------+
|       1001 |   10001 | Black Airpods           |     129.99 | Apple        | San Francisco |
|       1002 |   10024 | Windows 10 Professional |     149.99 | Microsoft    | Redmond       |
|       1003 |   10157 | Uber T-shirt            |      19.95 | Uber         | Los Angeles   |
|       1004 |   10402 | Lyft Baseball Cap       |      21.99 | Lyft         | Chicago       |
+------------+---------+-------------------------+------------+--------------+---------------+
4 rows in set (0.00 sec)

mysql>
```

The command returns all attributes contained in either table, but only includes company_id once (and displays it in the first column) because it's foreign key in table1 and the primary key in table2.

4. command: **`SELECT * FROM table1 WHERE item_name LIKE '%r%';`**

```
mysql> SELECT * FROM table1 WHERE item_name LIKE '%r%';
+---------+-------------------------+------------+------------+
| item_id | item_name               | item_price | company_id |
+---------+-------------------------+------------+------------+
|   10001 | Black Airpods           |     129.99 |       1001 |
|   10024 | Windows 10 Professional |     149.99 |       1002 |
|   10157 | Uber T-shirt            |      19.95 |       1003 |
+---------+-------------------------+------------+------------+
3 rows in set (0.00 sec)
```

The command returns this because the item_name attribute values 'Airpods', 'Professional', and 'Uber T-shirt' all contain r's. 'Lyft Baseball Cap' does not contain any r's, so that attribute value was not included in the list.

TECHNICAL WORK

1.
```
mysql> SELECT title FROM album WHERE year_of_recording = 1999;
+-------------------+
| title             |
+-------------------+
| Another Rap Album |
| Rap Album         |
+-------------------+
2 rows in set (0.00 sec)
```

2.
```
mysql> SELECT label_name FROM record_label WHERE year_of_founding = 1987 AND label_type_id = 2;
+--------------------------+
| label_name               |
+--------------------------+
| Real Ones Entertainment |
| Ref Jam Productions     |
+--------------------------+
2 rows in set (0.00 sec)
```

3.
```
mysql> SELECT label_name FROM record_label WHERE year_of_founding >= 1980 AND year_of_founding <= 1989 AND label_type_id = 2;
+--------------------------+
| label_name               |
+--------------------------+
| Real Ones Entertainment |
| Ref Jam Productions     |
+--------------------------+
2 rows in set (0.00 sec)
```

4.
```
mysql> SELECT first_name, last_name FROM music_artist WHERE music_group1 = "The Rappers" OR music_group2 = "The Rappers" OR music_group3 = "The Rappers";
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Jack       | Peterson  |
| Joanna     | Jacobson  |
| Josh       | Johnson   |
+------------+-----------+
3 rows in set (0.00 sec)
```

5.
```
mysql> SELECT first_name, last_name FROM music_artist WHERE music_group2 != " ";
+-------------+------------+
| first_name | last_name |
+-------------+------------+
| Josh       | Johnson    |
| Joanna     | Jacobson   |
+-------------+------------+
2 rows in set (0.00 sec)
```

6.
```
mysql> SELECT group_name, year_of_founding FROM music_group WHERE genre1 != " " AND genre2 != " " AND genre3 != " ";
+-------------------------+------------------+
| group_name              | year_of_founding |
+-------------------------+------------------+
| A Tribe Called Question |             1981 |
| Beat Creators           |             2008 |
| Jim Bean                |             1981 |
+-------------------------+------------------+
3 rows in set (0.01 sec)
```

7.
```
mysql> SELECT music_group FROM music_influences WHERE is_influenced_by = "A Tribe Called Question";
+-------------+
| music_group |
+-------------+
| Jim Bean    |
| The Rappers |
+-------------+
2 rows in set (0.00 sec)
```

8.

```
mysql> SELECT DISTINCT title FROM album NATURAL JOIN music_artist WHERE music_group = C
+--------------------+
| title              |
+--------------------+
| Jim's Album        |
| Jim's Second Album |
+--------------------+
2 rows in set (0.00 sec)
```

```
= CONCAT(first_name, " ", last_name) AND year_of_recording >= 1990 AND year_of_recording <= 1999;
```

9.

```
mysql> SELECT DISTINCT first_name, last_name, group_name FROM music_group, music_artist WHERE genre2 != " "
    -> ;
+------------+-----------+--------------------------+
| first_name | last_name | group_name               |
+------------+-----------+--------------------------+
| Joanna     | Jacobson  | A Tribe Called Question  |
| Joanna     | Jacobson  | Beat Creators            |
| Joanna     | Jacobson  | Jim Bean                 |
| Jim        | Bean      | A Tribe Called Question  |
| Jim        | Bean      | Beat Creators            |
| Jim        | Bean      | Jim Bean                 |
| John       | Doe       | A Tribe Called Question  |
| John       | Doe       | Beat Creators            |
| John       | Doe       | Jim Bean                 |
| Jack       | Peterson  | A Tribe Called Question  |
| Jack       | Peterson  | Beat Creators            |
| Jack       | Peterson  | Jim Bean                 |
| Josh       | Johnson   | A Tribe Called Question  |
| Josh       | Johnson   | Beat Creators            |
| Josh       | Johnson   | Jim Bean                 |
+------------+-----------+--------------------------+
15 rows in set (0.00 sec)
```

10.

```
mysql> SELECT a.music_group, b.music_group, a.songs FROM album a, album b WHERE a.songs = b.songs AND a.music_group != b.music_group;
+-------------------------+-------------------------+---------------------------------+
| music_group             | music_group             | songs                           |
+-------------------------+-------------------------+---------------------------------+
| The Rappers             | A Tribe Called Question | beat1, beat2, beat3, beat4, beat5 |
| A Tribe Called Question | The Rappers             | beat1, beat2, beat3, beat4, beat5 |
+-------------------------+-------------------------+---------------------------------+
2 rows in set (0.00 sec)
```

11. Show the names and years of founding of all groups influenced by A Tribe Called Question who were formed between 1900 and 2000.

SELECT group_name, year_of_founding FROM music_group NATURAL JOIN music_influences WHERE is_influenced_by = "A Tribe Called Question" AND year_of_founding < 2000 AND year_of_founding > 1900;

```
mysql> SELECT group_name, year_of_founding FROM music_group NATURAL JOIN music_influences WHERE
+-------------------------+------------------+
| group_name              | year_of_founding |
+-------------------------+------------------+
| A Tribe Called Question |             1981 |
| A Tribe Called Question |             1981 |
| Jim Bean                |             1981 |
| Jim Bean                |             1981 |
| The Rappers             |             1998 |
| The Rappers             |             1998 |
+-------------------------+------------------+
6 rows in set (0.00 sec)
```