CSC343 Worksheet 6 Solution

June 21, 2020

1. **Exercise 8.1.1:**

a)
```sql
CREATE VIEW RichExec AS
    SELECT * FROM MovieExec
    WHERE netWorth >= 10000000;

```

Notes:

- Virtual Views
  - **Syntax:** CREATE VIEW < view-name > AS < view-definition >
  - Contrasts to database that exists in physical storage
  - Exists in RAM
  - Is created using query
  - can be used like a relation

Notes:

```sql
CREATE VIEW ParamountMovies AS
    SELECT title, year
    FROM Movies
    WHERE studioName = 'Paramount';

```

b)
```sql
CREATE VIEW StudioPres AS
    SELECT * FROM Movies
    INNER JOIN Studio ON cert# = presC#;

```

c)
```sql
CREATE VIEW ExecutiveStar AS
    SELECT * FROM MovieExec
    NATURAL JOIN MovieStar;

```

2. **Exericse 8.1.2:**

a)
```
SELECT name, gender FROM ExecutiveStar;

```

b)
```
SELECT name FROM RichExec WHERE netWorth > 10000000;

```

c)
```
SELECT name FROM StudioPres
NATURAL JOIN ExecutiveStar
WHERE netWorth > 50000000

```

3. **Exericse 8.2.1:**

RichExec is updatable.

Notes:

- Updatable View Conditions
  - The WHERE cluase in CREATE VIEW must not be a subquery
  - The FROM clause has only one occurence of R
  - The SELECT clause must include enough attributes
  - NOT NULL attributes must have default values
    - A solution to this is by including the attribute without default value in CREATE VIEW

Example:

```
 Movies(title, year, length, genre, studioName, producerC#)
 Suppose studioName is NOT NULL but has no default value.
Then, a fix is:

 CREATE VIEW Paramount AS
     SELECT studioName, title, year
     FROM Movies
     WHERE studioName = 'Paramount';

```

4. **Exericse 8.2.2:**

a) No. It is not updatable. Since,

1. studioName attribute in Movies is NOT NULL without default value

b)
```
CREATE TRIGGER DisneyComediesInsert
INSTEAD OF INSERT ON DisneyComedies
REFERENCING
    NEW ROW AS NewTuple
FOR EACH ROW
INSERT INTO Movies(title, year, length, genre, studioName)
VALUES(NewTuple.title, NewTuple.year, NewTuple.length, 'comedy',
'Disney');

```

Notes:

- Using Trigger in VIEW
  - Uses INSTEAD OF in place of BEFORE or AFTER
  - When event causes the trigger, the trigger is done instead of the event

Example:

```
CREATE VIEW ParamountMovies AS
    SELECT title, year
    FROM Movies
    WHERE studioName = 'paramount';

CREATE TRIGGER ParamountInsert
INSTEAD OF INSERT ON ParamountMovies
REFERENCING NEW ROW AS NewRow
FOR EACH ROW
INSERT INTO Movies(title, year, studioName)
VALUES(NewRow.title, NewRow.year, 'Paramount');

```

c)
```
CREATE TRIGGER DisneyComediesInsert
INSTEAD OF INSERT ON DisneyComedies
REFERENCING
    NEW ROW AS NewTuple
    OLD ROW AS OldTuple
FOR EACH ROW
UPDATE Movies
SET length=NewTuple.length
WHERE title=OldTuple.title AND year=OldTuple.year;

```

5. **Exercise 8.2.3**

a) No. the view is not updatable. Because for it to be updatable, only one relation must exist in FROM

b)
```
CREATE TRIGGER NewPCInsert
INSTEAD OF INSERT ON NewPC
REFERENCING
```

```
        NEW ROW AS NewTuple
        OLD ROW AS OldTuple
    FOR EACH ROW
    INSERT INTO PC(model speed, ram, hd ,price)
    VALUES (NewTuple.model, NewTuple.speed, NewTuple.ram, NewTuple.hd
    , NewTuple.price);

    INSERT INTO Product(maker, model, type)
    VALUES (NewTuple.maker, NewTuple.model, 'pc');

```

c)

```
    CREATE TRIGGER NewPCUpdate
    INSTEAD OF INSERT ON NewPC
    REFERENCING
        NEW ROW AS NewTuple
    FOR EACH ROW
    UPDATE PC
    SET model=NewTuple.model
        speed=NewTuple.speed,
        ram=NewTuple.ram,
        hd=NewTuple.hd,
        price=NewTuple.price;

    UPDATE Product
    SET maker=NewTuple.maker,
        model=NewTuple.model,
        type='pc';

```

Correct Solution:

```
    CREATE TRIGGER NewPCUpdate
    INSTEAD OF UPDATE ON NewPC
    REFERENCING
        NEW ROW AS NewTuple
    FOR EACH ROW
    UPDATE PC
    SET model=NewTuple.model
        speed=NewTuple.speed,
        ram=NewTuple.ram,
        hd=NewTuple.hd,
        price=NewTuple.price;

    UPDATE Product
    SET maker=NewTuple.maker,
        model=NewTuple.model,
        type='pc';

```

d)

```
    CREATE TRIGGER NewPCDelete
    INSTEAD OF DELETE ON NewPC
```

```
    REFERENCING
        NEW ROW AS NewTuple
    FOR EACH ROW
    DELETE FROM PC
    WHERE model=NewTuple.model;

    DELETE FROM Product
    WHERE model=NewTuple.model;

```

6. a)
```
    CREATE INDEX studioNameIndex Studio(name)

```

Notes:

- Indexes
  - **Syntax (Create Index):**
    CREATE INDEX < index-name > R(< attributes >)
  - **Syntax (Drop Index):**
    DROP INDEX < index-name >
  - Used to find tuples in a very large database
    - Is efficient
  - Can be thought as (key, value) pair in a binary search tree
  - e.g. Declaring Index
```
    CREATE INDEX KeyIndex ON Movies(title, year);

```
  - e.g. Dropping index
```
    CREATE INDEX KeyIndex ON Movies(title, year);

```

b)
```
    CREATE INDEX movieExecAddressIndex MovieExec(address)

```

c)
```
    CREATE INDEX movieKeyIndex Movies(genre, length)

```

7. **Exercise 8.4.1:**

Action	No Index	Star Index	Movie Index	Both Indexes
Q_1	100	4	100	4
Q_2	100	100	4	4
I	2	4	4	6
Average	$2 + 100p_1 + 100p_2$	$4 + 96p_2$	$4 + 96p_1$	$6 - 2p_1 - 2p_2$

Notes:

- Database Tuning
  - Index sppeds up queries that can use it
  - Index should NOT be created when modifications are the frequent choice of action

8. **Exercise 8.4.2:**

   Omitted for the time being

9. **Exercise 8.5.1:**

```sql
UPDATE MovieProd
SET name='New Name'
WHERE (title, year) IN
(
    SELECT title, year FROM Movies
    INNER JOIN MovieExecs
    ON Movies.productC# = MovieExec.cert#
    WHERE cert# = '4567'
);

```

Notes:

- Materialized Views
  - Stores view in physical storage