

CSC343 Worksheet 6 Solution

June 21, 2020

1. Exercise 6.6.1:

a) Notes:

- Transactions
 - is a collection of one or more operations that must be executed atomically
 - COMMIT causes the transaction to end successfully
 - ROLLBACK causes the transaction to abort. Any changes are undone

Example:

```
1  BEGIN TRANSACTION;  
2  
3  UPDATE accounts  
4  SET balance = balance - 1000  
5  WHERE account_no = 100;  
6  
7  UPDATE accounts  
8      SET balance = balance + 1000  
9  WHERE account_no = 200;  
10  
11 INSERT INTO account_changes(account_no,flag,amount,  
12 changed_at)  
13 VALUES(100,'-',1000,datetime('now'));  
14 COMMIT;  
15
```

2. Exercise 8.1.1:

```
a) CREATE VIEW RichExec AS  
2  SELECT * FROM MovieExec  
3  WHERE netWorth >= 100000000;  
4
```

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:

```

1      CREATE VIEW ParamountMovies AS
2          SELECT title, year
3          FROM Movies
4          WHERE studioName = 'Paramount';
5

```

```

b)      CREATE VIEW StudioPres AS
2          SELECT * FROM Movies
3          INNER JOIN Studio ON cert# = presC#;
4

```

```

c)      CREATE VIEW ExecutiveStar AS
2          SELECT * FROM MovieExec
3          NATURAL JOIN MovieStar;
4

```

3. Exercise 8.1.2:

```

a)      SELECT name, gender FROM ExecutiveStar;
2

```

```

b)      SELECT name FROM RichExec WHERE netWorth > 10000000;
2

```

```

c)      SELECT name FROM StudioPres
2          NATURAL JOIN ExecutiveStar
3          WHERE netWorth > 50000000
4

```

4. Exercise 8.2.1:

RichExec is updatable.

Notes:

- Updatable View Conditions

- The WHERE clause in CREATE VIEW must not be a subquery
- The FROM clause has only one occurrence 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:

```

1  Movies(title, year, length, genre, studioName, producerC#)
2  Suppose studioName is NOT NULL but has no default value.
   Then, a fix is:
3
4  CREATE VIEW Paramount AS
5      SELECT studioName, title, year
6      FROM Movies
7      WHERE studioName = 'Paramount';
8

```

5. Exercise 8.2.2:

a) No. It is not updatable. Since,

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

b)

```

1  CREATE TRIGGER DisneyComediesInsert
2  INSTEAD OF INSERT ON DisneyComedies
3  REFERENCING
4      NEW ROW AS NewTuple
5  FOR EACH ROW
6  INSERT INTO Movies(title, year, length, genre, studioName)
7  VALUES(NewTuple.title, NewTuple.year, NewTuple.length, 'comedy',
8  '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:

```

1      CREATE VIEW ParamountMovies AS
2          SELECT title, year
3          FROM Movies
4          WHERE studioName = 'paramount';
5
6      CREATE TRIGGER ParamountInsert
7      INSTEAD OF INSERT ON ParamountMovies
8      REFERENCING NEW ROW AS NewRow
9      FOR EACH ROW
10     INSERT INTO Movies(title, year, studioName)
11     VALUES(NewRow.title, NewRow.year, 'Paramount');
12

```

```

c)    CREATE TRIGGER DisneyComediesInsert
2      INSTEAD OF INSERT ON DisneyComedies
3      REFERENCING
4          NEW ROW AS NewTuple
5          OLD ROW AS OldTuple
6      FOR EACH ROW
7      UPDATE Movies
8      SET length=NewTuple.length
9      WHERE title=OldTuple.title AND year=OldTuple.year;
10

```

6. 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
2      INSTEAD OF INSERT ON NewPC
3      REFERENCING
4          NEW ROW AS NewTuple
5          OLD ROW AS OldTuple
6      FOR EACH ROW
7      INSERT INTO PC(model speed, ram, hd ,price)
8      VALUES (NewTuple.model, NewTuple.speed, NewTuple.ram, NewTuple.hd
9      , NewTuple.price);
10
11     INSERT INTO Product(maker, model, type)
12     VALUES (NewTuple.maker, NewTuple.model, 'pc');

```

```

c)    CREATE TRIGGER NewPCUpdate
2      INSTEAD OF INSERT ON NewPC
3      REFERENCING
4          NEW ROW AS NewTuple
5      FOR EACH ROW
6      UPDATE PC
7      SET model=NewTuple.model
8          speed=NewTuple.speed,
9          ram=NewTuple.ram,

```

```

10         hd=NewTuple.hd,
11         price=NewTuple.price;
12
13     UPDATE Product
14     SET maker=NewTuple.maker,
15         model=NewTuple.model,
16         type='pc';
17

```

Correct Solution:

```

1     CREATE TRIGGER NewPCUpdate
2     INSTEAD OF UPDATE ON NewPC
3     REFERENCING
4         NEW ROW AS NewTuple
5     FOR EACH ROW
6     UPDATE PC
7     SET model=NewTuple.model
8         speed=NewTuple.speed,
9         ram=NewTuple.ram,
10        hd=NewTuple.hd,
11        price=NewTuple.price;
12
13     UPDATE Product
14     SET maker=NewTuple.maker,
15         model=NewTuple.model,
16         type='pc';
17

```

```

d)    CREATE TRIGGER NewPCDelete
2     INSTEAD OF DELETE ON NewPC
3     REFERENCING
4         NEW ROW AS NewTuple
5     FOR EACH ROW
6     DELETE FROM PC
7     WHERE model=NewTuple.model;
8
9     DELETE FROM Product
10    WHERE model=NewTuple.model;
11

```

```

7. a)    CREATE INDEX studioNameIndex Studio(name)
2

```

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

```
1 CREATE INDEX KeyIndex ON Movies(title, year);
2
```

– e.g. Dropping index

```
1 CREATE INDEX KeyIndex ON Movies(title, year);
2
```

b)

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

c)

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

8. **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 speeds up queries that can use it
- Index should NOT be created when modifications are the frequent choice of action

9. **Exercise 8.4.2:**

Omitted for the time being

10. **Exercise 8.5.1:**

```
1 UPDATE MovieProd
2 SET name='New Name'
3 WHERE (title, year) IN
4 (
5     SELECT title, year FROM Movies
6     INNER JOIN MovieExecs
```

```
7      ON Movies.productC# = MovieExec.cert#  
8      WHERE cert# = '4567'  
9  );  
10
```

Notes:

- Materialized Views
 - Is also known as a summary
 - Is also known as black-box abstraction
 - Stores view in physical storage
 - Useful when storing expensive operation like AVG or COUNT