CSC343 Worksheet 6

June 20, 2020

1. **Exercise 6.6.1:** This and the next exercises involve certain programs that operate on the two relations

```
1    Product(maker, model, type)
2    PC(model, speed, ram, hd, price)
3
```

   from our running PC exercise. Sketch the following programs, including SQL statements and work done in a conventional language. Do not forget to issue BEGIN TRANSACTION, COMMIT, and ROLLBACK statements at the proper times and to tell the system your transactions are read-only if they are.

   - Given a speed and amount of RAM (as arguments of the function), look up the PC 's with that speed and RAM, printing the model number and price of each.
   - Given a model number, delete the tuple for that model from both PC and Product.
   - Given a model number, decrease the price of that model PC by $100.
   - Given a maker, model number, processor speed, RAM size, hard-disk size, and price, check that there is no product with that model. If there is such a model, print an error message for the user. If no such model existed in the database, enter the information about that model into the PC and Product tables.

2. **Exercise 6.6.2:** For each of the programs of Exercise 6.6.1, discuss the atomicity problems, if any, that could occur should the system crash in the middle of an execution of the program.

3. **Exercise 6.6.3:** Suppose we execute as a transaction T one of the four programs of Exercise 6.6.1, while other transactions that are executions of the same or a different one of the four programs may also be executing at about the same time. What behaviors of transaction T may be observed if all the transactions run with isolation level READ UNCOMMITTED that would not be possible if they all ran with isolation level SERIALIZABLE? Consider separately the case that T is any of the programs (a) through (d) of Exercise 6.6.1.

4. **Exercise 8.1.1:** From the following base tables of our running example

```
MovieStar(name, address, gender, birthdate)
MovieExec(name, address, cert# , netWorth)
Studio(name, address, presC#)
```

Construct the following views

- A view RichExec giving the name, address, certificate number and networth of all executives with a net worth of at least $10,000,000.
- A view StudioPres giving the name, address, and certificate number of all executives who are studio presidents.
- A view ExecutiveStar giving the name, address, gender, birth date, certificat number, and net worth of all individuals who are both executives and stars.

5. **Exercise 8.1.2:** Write each of the queries below, using one or more of the views from Exercise 8.1.1 and no base tables.

- Find the names of females who are both stars and executives.
- Find the names of those executives who are both studio presidents and worth at least $10,000,000.
- Find the names of studio presidents who are also stars and are worth at least $50,000,000.

6. **Exercise 8.2.1:** Which of the views of Exercise 8.1.1 are updatable?

7. **Exercise 8.2.2:** Suppose we create the view:

```
CREATE VIEW DisneyComedies AS
SELECT title ,year, length FROM Movies
WHERE studioName = 'Disney' AND genre = 'comedy';
```

- Is this view updatable?
- Write an instead-of trigger to handle an insertion into this view.
- Write an instead-of trigger to handle an update of the length for a movie (given by title and year) in this view.

8. **Exercise 8.2.3:** Using the base tables

```
Product(maker, model, type)
PC(model, speed, ram, hd, price )
```

suppose we create the view

```
CREATE VIEW NewPC AS
SELECT maker, model, speed, ram, hd, price
FROM Product, PC
WHERE Product.model = PC.model AND type = 'pc';

```

Notice that we have made a check for consistency: that the model number not only appears in the PC relation, but the ty p e attribute of Product indicates that the product is a PC.

- Is this view updatable?
- Write an instead-of trigger to handle an insertion into this view.
- Write an instead-of trigger to handle an update of the price.
- Write an instead-of trigger to handle a deletion of a specified tuple from this view.

9. **Exercise 8.3.1:** For our running movies example:

```
Movies(title, year, length, genre, studioName, producerC\#)
StarsIn(movieTitle, movieYear, starName)
MovieExec(name, address, cert\#, netWorth)
Studio(name, address, presC\#)

```

Declare indexes on the following attributes or combination of attributes:

- studioName.
- address of MovieExec.
- genre and length.

10. **Exercise 8.4.1:** Suppose that the relation StarsIn discussed in Example 8.14 required 100 pages rather than 10, but all other assumptions of that example continued to hold. Give formulas in terms of pi and p? to measure the cost of queries Q and Q2 and insertion I, under the four combinations of index/no index discussed there.

11. **Exercise 8.4.2:** In this problem, we consider indexes for the relation

```
Ships(name, class, launched)

```

from our running battleships exercise. Assume:

i. name is the key.

ii. The relation Ships is stored over 50 pages.

iii. The relation is clustered on class so we expect that only one disk access is needed to find the ships of a given class.

iv. On average, there are 5 ships of a class, and 25 ships launched in any given year.

v. With probability p the operation on this relation is a query of the form SELECT * FROM Ships WHERE name = n.

vi. With probability p2 the operation on this relation is a query of the form SELECT * FROM Ships WHERE class = c.

vii. With probability pz the operation on this relation is a query of the form SELECT * FROM Ships WHERE launched = y.

viii. With probability 1 — pi — p2 — P3 the operation on this relation is an insertion of a new tuple into Ships.

You can also make the assumptions about accessing indexes and finding empty space for insertions that were made in Example 8.14.

Consider the creation of indexes on name, class , and launched. For each combination of indexes, estimate the average cost of an operation. As a function of p_1, p_2, and p_3, what is the best choice of indexes?

12. **Exercise 8.5.1:** Complete Example 8.15 by considering updates to either of the base tables.

13. **Exercise 8.5.2:** Suppose the view NewPC of Exercise 8.2.3 were a materialized view. What modifications to the base tables Product and PC would require a modification of the materialized view? How would you implement those modifications incrementally?

14. **Exercise 8.5.3:** This exercise explores materialized views that are based on aggregation of data. Suppose we build a materialized view on the base tables

```
Classes(class, type, country, numGuns, bore, displacement)
Ships(name, class, launched)

```

from our running battleships exercise, as follows:

```
CREATE MATERIALIZED VIEW ShipStats AS
SELECT country, AVG(displacement), COUNT(*)
FROM Classes, Ships
WHERE Classes.class = Ships.class
GROUP BY country;

```

What modifications to the base tables Classes and Ships would require a modification of the materialized view? How would you implement those modifications incrementally?

15. **Exercise 8.5.4:** In Section 8.5.3 we gave conditions under which a materialized view of simple form could be used in the execution of a query of similar form. For the view of Example 8.15, describe all the queries of that form, for which this view could be used.