CS 317 – Data Management and Information Systems

Professor Yan Chen
Homework 5 Solutions
Grader: Jin Hu

Problem 6.4.1 (15 points)

Part b) (5 points)

```
SELECT DISTINCT   maker
FROM Product, Laptop
WHERE Product.model = Laptop.model AND hd >= 1;
```

Part c) (5 points)

```
(SELECT DISTINCT  PC.model as model,
                  PC.price as price
 FROM Product, PC
 WHERE Product.model = PC.model AND maker = 'B')

UNION

(SELECT DISTINCT  Laptop.model as model,
                  Laptop.price as price
 FROM Product, Laptop
 WEHRE Product.model = Laptop.model AND maker = 'B')

UNION

(SELECT DISTINCT  Printer.model as model,
                  Printer.price as price
 FROM Product, Printer
 WEHRE Product.model = Printer.model AND maker = 'B');
```

Part d) (5 points)

```
SELECT DISTINCT   model
FROM Printer
WHERE color = 'true' AND type = 'laser';
```

Problem 6.4.6 (30 points)

Part b) (5 points)

SQL:
```
SELECT      AVG(speed)
FROM Laptop
WHERE price > 2000;
```

Result:
```
AVG(speed)
----------
   775
```

Part c) (5 points)

SQL:
```
SELECT      AVG(price)
FROM Product, PC
WHERE Product.model = PC.model AND maker = 'A';
```

Result:
```
AVG(price)
----------
   1776
```

Part e) (5 points)

SQL:
```
SELECT      speed, AVG(price)
FROM PC
GROUP BY speed;
```

Result:

speed	AVG(price)
700	899
1500	2499
866	1499
1000	1499
1300	2199
1400	2299
1200	1699
750	699
1100	1299
350	799
733	2499

Part g) (5 points)

SQL:
```
SELECT      maker
FROM Product
WHERE type = 'PC'
GROUP BY maker
HAVING COUNT(model) >= 3;
```

Result:
```
maker
-----
  A
  B
  D
```

Part h) (5 points)

SQL:
```
SELECT      maker, MAX(price)
FROM Product, PC
WHERE Product.model = PC.model
GROUP BY maker;
```

Result:
```
maker    MAX(price)
-------------------
  A         2499
  B         2119
  C         2299
  D         1699
  E         2499
```

Part i) (5 points)

SQL:
```
SELECT      speed, AVG(price)
FROM PC
WHERE speed > 800
GROUP BY speed;
```

Result:
```
speed    AVG(price)
-------------------
 1500       2499
  866       1499
 1000       1499
 1300       2199
 1400       2299
 1200       1699
 1100       1299
```

Problem 6.5.1 (20 points)

Part a) (5 points)

SQL 1:
```
INSERT INTO Product
VALUES('C', 1100, 'pc');
```

SQL 2:
```
INSERT INTO PC
VALUES(1100, 1800, 256, 80, '20x DVD', 2499);
```

Effect 1: `Product` relation now has a new tuple with the above attributes.

Effect 2: `PC` table now has a corresponding tuple describing in further detail about model 1100, such as the speed, hd, rd, etc.

Part d) (5 points)

SQL:
```
DELETE FROM Laptop
WHERE Laptop.model IN
(SELECT     model
 FROM Product
 WHERE maker NOT IN
      (SELECT     maker
       FROM Product
       WHERE type = 'printer'));
```

Effect: From the `Laptop` relation, all tuples that have models corresponding to makers that make printers are deleted.

Part e) (5 points)

SQL:
```
UPDATE      Product
SET maker = 'A'
WHERE maker = 'B';
```

Effect: For the `Product` table, all makers that were 'B' are now 'A'.

Part f) (5 points)

SQL:
```
UPDATE      PC
SET   ram = ram * 2,
      hd = hd + 20;
```

Effect: For every entry in PC, all RAM values are now doubled (x2) and all hard-drive values have gained 20 gigabytes.

Problem 6.5.2 (5 points)

Part e) (5 points)

SQL:

```
DELETE FROM Classes
WHERE class IN
(SELECT     class
 FROM Ships
 GROUP BY class
 HAVING COUNT(name) < 3);
```

Effect: From the `Classes` table, all tuples whose `class` attribute corresponded to having fewer than 3 ships were deleted.

Problem 6.6.2 (25 points)

Part a) (5 points)

```
CREATE TABLE Product
(
      maker       CHAR(1),
      model       INTEGER,
      type        VARCHAR(7)
);
```

Part b) (5 points)

```
CREATE TABLE PC
(
      model       INTEGER,
      speed       INTEGER,
      ram         INTEGER,
      hd          INTEGER,
      rd          VARCHAR(10),
      price       INTEGER
);
```

Part d) (5 points)

```
CREATE TABLE Printer
(
      model       INTEGER,
      color       BOOLEAN,
      type        VARCHAR(7),
      price       INTEGER
);
```

Part e) (5 points)

```
ALTER TABLE Printer DROP color;
```

Part f) (5 points)

```
ALTER TABLE Laptop ADD cd VARCHAR(10) DEFAULT ‘none’;
```

Problem 6.7.1 (10 points)

Part b) (5 points)

```
CREATE VIEW StudioPres AS
      SELECT      MovieExec.name as name,
                  MovieExec.address as address,
                  cert#
      From Studio, MovieExec
      WHERE cert# = presC#;
```

Part c) (5 points)

```
CREATE VIEW ExecutiveStar AS
      SELECT      MovieStar.name as name,
                  MovieStar.address as address,
                  gender,
                  birthdate,
                  cert#,
                  networth
      FROM MovieStar, MovieExec
      WHERE MovieStar.name = MovieExec.name
            AND MovieStar.address = MovieExec.address;
```

Problem 6.7.2 (10 points)

Part a) (5 points)

Updatable
We are only grabbing attributes from one table and so long as we insert the necessary or required information (all attributes displayed), then we can update the view. Also note that `RichExec` is not part of any subquery of the original view creation:

```
CREATE VIEW RichExec AS
      SELECT *
      FROM MovieExec
      WHERE networth >= 10000000;
```

Part c) (5 points)

Not Updatable
Even if we insert all the necessary information (all the displayed attributes), either:

`MovieStar.name` and `MovieStar.address`

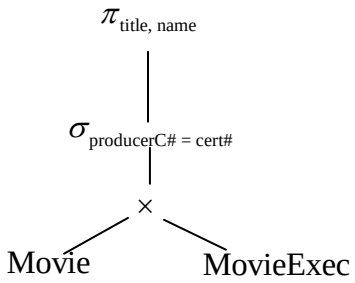
or

`MovieExec.name` and `MovieExec.address`

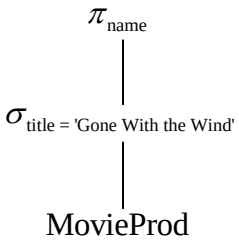
will be NULL (in my case, the latter). Thus, there will be no way to compare the names and addresses in the WHERE clause, causing updating problems.

Problem 6.7.4 (6 points)

Part a) (2 points)



Part b) (2 points)



Part c) (2 points)