COMP 353 Nooraldin Abdelgaiyd, 40102069

Assignment 4

1.

Exercise 5.4.1 (D, F)

D)

(R ∪ S) - T

(R ∪ S) <= R(a,b,c) OR S(a,b,c)

-T <= NOT S(a,b,c)

Thus

D(a,b,c) <= R(a,b,c) OR S(a,b,c) AND NOT T(a,b,c)

F)

πa,b(R)

F(a,b) <= R(a,b,c)

2.

Exercise 5.4.2 (a,b,f)

A)

Let A(x,y,z) be the resulting relation. X = Y means an AND operation must be used. Therefore

A(x,y,z) <= R(x,y,z) AND X = Y

B)

Let B(x,y,z) be the resulting relation. X< y AND y <z means an AND operation must be used. Therefore

B(x,y,z) <= R(x,y,z) AND x < y AND y < z

F)

Let F(x,y,z) be the resulting relation. NOT((x<y OR x<z) AND y<z) means Demorgan's law must be used. Therefore

It can be rewritten as = NOT(x<y) AND NOT(x<z) OR NOT (y<z)

= x >= y AND x >=z OR y>= z

Hence

F(x,y,z) <= R(x,y,z) AND x >= y AND x>= z

3.

Exercise 5.4.4 (c,d)

C)

R(rx,ry,rz) AND S(sx,sy,sz) AND rx < sy therefore,

Result(rx,ry,rz,sx,sy,sz) <= R(rx,ry,rz) AND S(sx,sy,sz) AND rx<sy

D)

R(rx,ry,rz) AND S(sx,sy,sz) AND rx = sy therefore,

Result(rx,ry,rz,sx,sy,sz) <= R(rx,ry,rz) AND S(sx,sy,sz) AND rx=sy

4.

Exercise 6.3.1 (a, b, e, f)

A)

1.

```
SELECT DISTINCT PR.maker FROM Product PR

WHERE EXISTS (SELECT COM.model FROM PC COM WHERE PR.speed >= 3.0 AND PR.model = COM.model);
```

2.

```
SELECT DISTINCT maker FROM Product

WHERE model MO (SELECT model from PC WHERE speed >- 3.0);
```

B)

1.

```
SELECT PRINT.model FROM Printer PRINT WHERE PRINT.price IN

(SELECT MAX (PRINT2.price) FROM Printer PRINT2 );
```

2.

```
SELECT PRINT.model FROM Printer PRINT WHERE PRINT.price >= ALL (SELECT PRINT2.price FROM Printer PRINT2);
```

E)

1.

```
SELECT DISTINCT maker from product, printer

WHERE color = ‘TRUE’ AND printer.model = product.model AND price <= ALL (SELECT MIN(price) from
printer where color = ‘TRUE’);
```

2.

```
SELECT DISTINCT maker from product, printer

WHERE color = ‘TRUE’ AND printer.model = product.model AND price <= INT (SELECT MIN(price) from
printer where color = ‘TRUE’);
```

F)

1.

```
Select maker from product, PC where product.model = pc.model AND RAM in (select min(RAM) from PC)

AND speed >= ALL (select speed from PC where RAM = (select min(RAM) from PC));
```

2.

```
SELECT distinct maker from product, pc WHERE product product.model AND RAM<= ALL (select RAM
from PC) AND speed >= ALL (select speed from PC where RAM =(select min(RAM) from PC));
```

5.

Exercise 6.3.8

```
SELECT Product.maker, Product.type, PC.model, PC.speed, PC.ram, PC.hd, NULL as 'screen', PC.rd, NULL
as 'color', NULL as 'printer_type', PC.price FROM PC, Product

WHERE PC.model = Product.model

UNION

SELECT Product.maker, Product.type, Laptop.model, Laptop.speed, Laptop.ram, Laptop.hd,
Laptop.screen, NULL as 'rd', NULL as 'color', NULL as 'printer_type', Laptop.price FROM Laptop, Product

WHERE Laptop.model = Product.model
```

```
UNION

SELECT Product.maker, Product.type, Printer.model, NULL as 'speed', NULL as 'ram', NULL as 'hd', NULL as 'screen', NULL as 'rd', Printer.color, Printer.type as 'printer_type', Printer.price FROM Printer, Product

WHERE Printer.model = Product.model

ORDER BY model;
```

6.

Exercise 6.5.1 (a, b, c, f)

A)

```
Insert into Product(maker, model, type) values ('C','1100','PC')

Insert into PC(model, speed, ram, hd, price) values ('1100',3.2,1024,180,2499)
```

B)

```
insert into Product(maker, model, type)

(select maker, model + 1100, 'Laptop' from Product where type = 'PC');


insert into Laptop(model, speed, ram, hd, screen, price)

(select model + 1100, speed, ram, hd, 17, price + 500 from PC);
```

C)

```
DELETE

FROM pc

WHERE hd < 100;
```

F)

```
UPDATE pc

SET ram = ram * 2, hd = hd + 60;
```

7.

Exercise 7.2.5 (a, b)

A)

```
Alter table Classes ADD CONSTRAINT borechecksize CHECK (bore < 16)
```

B)

```
Alter table Classes ADD CONSTRAINT borecheck CHECK ( numGUNS > 9 AND bore < 14)
```

8.

Exercise 8.2.3 (c, d)

C)

```
CREATE OR REPLACE FUNCTION updateProcedure()
RETURNS TRIGGER
AS $updateProcedure$
BEGIN
UPDATE PC set model = NEW.model, speed = NEW.speed, ram = NEW.ram, hd = NEW.hd, price = NEW.price;
update Product set maker=NEW.maker, model = NEW.model;
RETURN NULL;
END;
$updateProcedure$ LANGUAGE plpgsql;
CREATE TRIGGER updateTrigger INSTEAD OF UPDATE ON NewPC
FOR EACH ROW EXECUTE PROCEDURE updateProcedure();
```

D)

```
CREATE OR REPLACE FUNCTION deleteProcedure()
RETURNS TRIGGER
AS $deleteProcedure$
BEGIN
delete from PC where model= OLD.model;
delete Product where model = OLD.model;
RETURN NULL;
END;
$deleteProcedure$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER deleteTrigger INSTEAD OF DELETE ON NewPC

FOR EACH ROW EXECUTE PROCEDURE deleteProcedure();
```