Homework 3

**1, 7.1.1**

1. The producer of a movie must be someone mentioned in MovieExec. Modifications to MovieExec that violate this constraint are rejected.

CREATE TABLE Movies (

title VARCHAR(50),

year INTEGER,

length INTEGER,

genre VARCHAR(20),

studioName VARCHAR(20),

producerC# INTEGER REFERENCE MovieExec(cert#),

PRIMARY KEY ( title, year));

1. Repeat (a), but violations result in the producerC# in Movie being set to NULL.

CREATE TABLE Movies (

title VARCHAR(50),

year INTEGER,

length INTEGER,

genre VARCHAR(20),

studioName VARCHAR(20),

producerC# INTEGER REFERENCE MovieExec(cert#)

ON DELETE SET NULL

ON UPDATE SET NULL,

PRIMARY KEY ( title, year));

1. Repeat (a), but violations result in the deletion or update of the offending Movie tuple.

CREATE TABLE Movies (

title VARCHAR(50),

year INTEGER,

length INTEGER,

genre VARCHAR(20),

studioName VARCHAR(20),

producerC# INTEGER REFERENCE MovieExec(cert#)

ON DELETE CASCADE

ON UPDATE CASCADE,

PRIMARY KEY ( title, year));

1. A movie that appears in Starsln must also appear in Movie. Handle violations by rejecting the modification.

CREATE TABLE StarsIn (

movieTitle VARCHAR(50) REFERENCE Movie (title),

MovieYear INTEGER,

StarName VARCHAR(25),

PRIMARY KEY ( movieTitle, movieYear, starName));

1. A star appearing in Starsln must also appear in MovieStar. Handle violations by deleting violating tuples.

CREATE TABLE StarsIn (

movieTitle VARCHAR(50) REFERENCE Movie (title)

ON DELETE CASCADE,

MovieYear INTEGER,

StarName VARCHAR(25),

PRIMARY KEY ( movieTitle, movieYear, starName));

**2. 7.4.1**

1. No manufacturer of PC’s may also make laptops.

CREATE ASSERTION CHECK

(NOT EXISTS

(

(SELECT maker FROM Product NATURAL JOIN PC)

INTERSECT

(SELECT maker FROM Product NATURAL JOIN Laptop)

)

);

1. A manufacturer of a PC must also make a laptop with at least as great a processor speed.

CREATE ASSERTION CHECK

(NOT EXISTS

(SELECT maker

FROM Product NATURAL JOIN PC)

WHERE speed > ALL

(SELECT b.speed

FROM Product a, Laptop b

WHERE a.maker = maker AND

a.model = b.model

)

) );

1. If a laptop has a larger main memory than a PC, then the laptop must also have a higher price than the PC.

CREATE ASSERTION CHECK

(NOT EXISTS

(SELECT model FROM LaptopWHERE price <=ALL

(SELECT price FROM PC WHERE PC.ram < Laptop.ram)

)

);

1. If the relation Product mentions a model and its type, then this model must appear in the relation appropriate to that type.

CREATE ASSERTION CHECK

(EXISTS

(

(SELECT b.model

FROM Product a, PC b

WHERE a.type = “pc” AND

a.model = b.model)

UNION ALL

(SELECT L.model

FROM Product c, Laptop d

WHERE c.type = “laptop” AND

c.model = d.model )

UNION ALL

(SELECT e.model

FROM Product e, Printer f

WHERE e.type = “printer ” AND

e.model = f.model )

)

);

**3. 7.5.2**

1. When updating the price of a PC, check that there is no lower priced PC with the same speed.

CREATE TRIGGER LowPricePCTrigger

AFTER UPDATE OF price ON PC

REFERENCING

OLD ROW AS Row,

OLD TABLE AS Table,

NEW ROW AS Row2,

NEW TABLE AS Table2

FOR EACH ROW

WHEN (Row2.price < ALL

(SELECT PC.price

FROM PC

WHERE PC.speed = Row2.speed))

BEGIN

DELETE FROM PC

WHERE (model, speed, ram, hd, price) IN Table2;

INSERT INTO PC

(SELECT \*

FROM Table);

END;

1. When inserting a new printer, check that the model number exists in Product.

CREATE TRIGGER NewPrinterTrigger

AFTER INSERT ON Printer

REFERENCING

NEW ROW AS Row,

NEW TABLE AS Table

FOR EACH ROW

WHEN (NOT EXISTS (SELECT \* FROM Product

WHERE Product.model = Row.model)) DELETE FROM Printer

WHERE (model, color, type, price) IN Table;

1. When making any modification to the Laptop relation, check that the average price of laptops for each manufacturer is at least $1500.

CREATE TRIGGER AvgPriceTrigger

AFTER UPDATE OF price ON Laptop

REFERENCING

OLD TABLE AS Tableold,

NEW TABLE AS Tablenew

FOR EACH STATEMENT

WHEN (1500 > (SELECT AVG(price) FROM Laptop))

BEGIN

DELETE FROM Laptop

WHERE (model, speed, ram, hd, screen, price) IN Tablenew;

INSERT INTO Laptop

(SELECT \* FROM Tableold);

END;

1. When updating the RAM or hard disk of any PC, check that the updated PC has at least 100 times as much hard disk as RAM.

CREATE TRIGGER HardDiskTrigger

AFTER UPDATE OF hd, ram ON PC

REFERENCING

OLD ROW AS RowOld,

OLD TABLE AS TableOld,

NEW ROW AS RowNew,

NEW TABLE AS TableNew

FOR EACH ROW

WHEN (RowNew.hd < RowNew.ram \* 100)

BEGIN

DELETE FROM PC

WHERE (model, speed, ram, hd, price) IN TableNew ;

INSERT INTO PC

(SELECT \* FROM TableOld);

END;

1. When inserting a new PC, laptop, or printer, make sure that the model number did not previously appear in any of PC, Laptop, or Printer

CREATE TRIGGER DupModelTrigger

BEFORE INSERT ON PC, Laptop, Printer

REFERENCING

NEW ROW AS RowNew,

NEW TABLE AS TableNew

FOR EACH ROW

WHEN (EXISTS (SELECT \* FROM TableNew NATUAL JOIN PC)

UNION ALL

(SELECT \* FROM TableNew NATUAL JOIN Laptop)

UNION ALL

(SELECT \* FROM TableNew NATUAL JOIN Printer))

BEGIN SIGNAL SQLSTATE “10001”

(“Duplicate Model – Insert Failed‟);

END;

**4. 8.1.1**

1. A view RichExec giving the name, address, certificate number and net

worth of all executives with a net worth of at least $10,000,000.

CREATE VIEW RichExec AS

SELECT name, address, cert# netWorth

FROM MoviewExec= WHERE netWorth >=10000000;

1. A view StudioPres giving the name, address, and certificate number of all executives who are studio presidents.

CREATE VIEW StudioPres (name, address, cert#) AS

SELECT MovieExec.name, MovieExec.address, MovieExec.cert#

FROM MovieExec, Studio

WHEREMovieExec.cert# = Studio.presC#;

1. A view ExecutiveStar giving the name, address, gender, birth date, cer­ tificate number, and net worth of all individuals who are both executives and stars.

CREATE VIEW ExecutiveStar (name, address, gender, birthdate, cert#, netWorth) AS

SELECT star.name, star.address, star.gender, star.birthdate, exec.cert#, exec.netWorth

FROM MovieStar star, MovieExec exec

WHERE star.name = exec.name AND star.address = exec.address;

**5. 8.2.3**

1. Is this view updatable?

NO, the view is constructed from two different relations therefore it is not updatable

1. Write an instead-of trigger to handle an insertion into this view.

CREATE TRIGGER NewPCInsert

INSTEAD OF INSERT ON PCNew

REFERENCING

NEW ROW AS RowNew

FOR EACH ROW

(INSERT INTO Product VALUES(RowNew.maker, RowNew.model, ‘pc’)) (INSERT INTO PC VALUES(RowNew.model, RowNew.speed, RowNew.ram, RowNew.hd, RowNew.price));

1. Write an instead-of trigger to handle an update of the price.

CREATE TRIGGER NewPCUpdate

INSTEAD OF UPDATE ON PCNew

REFERENCING

NEW ROW AS RowNew

FOR EACH ROW

UPDATE PC SET price = PCNew.price where model = PCNew.model;

1. Write an instead-of trigger to handle a deletion of a specified tuple from this view.

CREATE TRIGGER NewPCDelete

INSTEAD OF DELETE ON PCNew

REFERENCING

OLD ROW AS RowOld

FOR EACH ROW

(DELETE FROM Product WHERE model = RowOld.model)

(DELETE FROM PC where model = RowOld.model);