Tenzin Tashi

Prof. Hesham A Auda

CSc - 33600 Introduction to Database Systems

April 19, 2021

Assignment 3

**The Products Database**

1. Consider the **Products** database schema and sample data in the course:

**Products**(*maker*, *model*, *type*)

**PCs**(*model, speed, ram, hd, price*)

**Laptops**(*model, speed, ram, hd, screen, price*)

**Printers**(*model, color, type, price*)

where *type* ∈{‘PC’, ‘Laptop’, ‘Printer’}, and *model* is the key of the database relations.

1. Implement the database using a RDBS of your choice.

/\*Table prodcuts\*/

CREATE TABLE Products(

maker CHAR,

model INT PRIMARY KEY,

TYPE CHAR (7) CHECK(type= 'PC' or type = 'Laptop' or type ='Printer')

);

/\*

Insert into Products table

\*/

INSERT INTO Products Values ('A',1001,'PC');

INSERT INTO Products Values ('A',1002,'PC');

INSERT INTO Products Values ('A',1003,'PC');

INSERT INTO Products Values ('A',2004,'Laptop');

INSERT INTO Products Values ('A',2005,'Laptop');

INSERT INTO Products Values ('A',2006,'Laptop');

INSERT INTO Products Values ('B',1004,'PC');

INSERT INTO Products Values ('B',1005,'PC');

INSERT INTO Products Values ('B',1006,'PC');

INSERT INTO Products Values ('B',2007,'Laptop');

INSERT INTO Products Values ('C',1007,'PC');

INSERT INTO Products Values ('D',1008,'PC');

INSERT INTO Products Values ('D',1009,'PC');

INSERT INTO Products Values ('D',1010,'PC');

INSERT INTO Products Values ('D',3004,'Printer');

INSERT INTO Products Values ('D',3005,'Printer');

INSERT INTO Products Values ('E',1011,'PC');

INSERT INTO Products Values ('E',1012,'PC');

INSERT INTO Products Values ('E',1013,'PC');

INSERT INTO Products Values ('E',2001,'Laptop');

INSERT INTO Products Values ('E',2002,'Laptop');

INSERT INTO Products Values ('E',2003,'Laptop');

INSERT INTO Products Values ('E',3001,'Printer');

INSERT INTO Products Values ('E',3002,'Printer');

INSERT INTO Products Values ('E',3003,'Printer');

INSERT INTO Products Values ('F',2008,'Laptop');

INSERT INTO Products Values ('F',2009,'Laptop');

INSERT INTO Products Values ('G',2010,'Laptop');

INSERT INTO Products Values ('H',3006,'Printer');

INSERT INTO Products Values ('H',3007,'Printer');

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Table PCs \*/

CREATE TABLE PCs(

model INT PRIMARY KEY REFERENCES Products(model),

speed FLOAT,

ram INT,

hd INT,

price FLOAT

);

/\*

Insert into PCs table

\*/

INSERT INTO PCs VALUES (1001,2.66,1024,250,2114);

INSERT INTO PCs VALUES (1002,2.10,512,250,995);

INSERT INTO PCs VALUES (1003,1.42,512,80,478);

INSERT INTO PCs VALUES (1004,2.80,1024,250,649);

INSERT INTO PCs VALUES (1005,3.20,512,250,630);

INSERT INTO PCs VALUES (1006,3.20,1024,320,1049);

INSERT INTO PCs VALUES (1007,2.20,1024,200,510);

INSERT INTO PCs VALUES (1008,2.20,2048,250,770);

INSERT INTO PCs VALUES (1009,2.00,1024,250,650);

INSERT INTO PCs VALUES (1010,2.80,2048,300,770);

INSERT INTO PCs VALUES (1011,1.86,2048,160,959);

INSERT INTO PCs VALUES (1012,2.80,1024,160,649);

INSERT INTO PCs VALUES (1013,3.06,512,80,529);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Table Laptop \*/

CREATE TABLE Laptops(

model INT PRIMARY KEY REFERENCES Products(model),

speed FLOAT,

ram INT,

hd INT,

screen FLOAT,

Price FLOAT);

/\*

Insert into Laptops table

\*/

INSERT INTO Laptops VALUES (2001,2.00,2048,240,20.1,3673);

INSERT INTO Laptops VALUES (2002,1.73,1024,80,17.0,949);

INSERT INTO Laptops VALUES (2003,1.80,512,60,15.4,549);

INSERT INTO Laptops VALUES (2004,2.00,512,60,13.3,1150);

INSERT INTO Laptops VALUES (2005,2.16,1024,120,17.0,2500);

INSERT INTO Laptops VALUES (2006,2.00,2048,80,15.4,1700);

INSERT INTO Laptops VALUES (2007,1.83,1024,120,13.3,1429);

INSERT INTO Laptops VALUES (2008,1.60,1024,100,15.4,900);

INSERT INTO Laptops VALUES (2009,1.60,512,80,14.1,680);

INSERT INTO Laptops VALUES (2010,2.00,2048,160,15.4,2300);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE Printers(

model INT PRIMARY KEY REFERENCES Products(model),

color BIT NOT NULL,

type CHAR (7) CHECK(type ='ink-jet' or type='laser'),

Price FLOAT);

/\*

Insert into Printer table

\*/

INSERT INTO Printers VALUES (3001,1,'ink-jet',99);

INSERT INTO Printers VALUES (3002,0,'laser',239);

INSERT INTO Printers VALUES (3003,1,'laser',899);

INSERT INTO Printers VALUES (3004,1,'ink-jet',120);

INSERT INTO Printers VALUES (3005,0,'laser',120);

INSERT INTO Printers VALUES (3006,1,'ink-jet',100);

INSERT INTO Printers VALUES (3007,1,'laser',200);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

B. Provide SQL queries that produce the **violations** of the following constraints:

Show the outputs of your queries using the sample data provided in the textbook.

*a.* No manufacturer of PCs may also make laptops.

/\*

Query: No manufacturer of PCs may also make laptops

\*/

SELECT DISTINCT Products.maker FROM Products where type IN ('PC')

AND Products.maker IN (

SELECT maker FROM Products where type IN ('Laptop'));

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

/\*

Query: A manufacturer of a PC must also make a laptop with at least as great a processor

\*/

SELECT DISTINCT Product.maker

FROM (SELECT DISTINCT maker from Products where model in (SELECT model from PCs)) Product

WHERE Product.maker not in

(SELECT DISTINCT R1.maker FROM

(SELECT P1.maker, P1.model, PC.speed FROM Products P1, PCs PC WHERE P1.model = PC.model) R1,

(SELECT P2.maker, Laptop.speed FROM Products P2, Laptops Laptop WHERE P2.model = Laptop.model) R2

WHERE R1.maker = R2.maker AND R1.speed <= R2.speed);

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

/\*

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

\*/

SELECT Laptops.model AS 'Models'

FROM PCs, Laptops

WHERE Laptops.ram>PCs.ram AND Laptops.price<= PCs.price;

*d.* If the relation Products mentions a model and its type, then the model must also appear in the relation appropriate to that type.

/\*

Query: D) If the relation Products mentions a model and its type, then the model must also appear in the relation appropriate to that type.

\*/

SELECT DISTINCT maker, model, type

FROM Products P

WHERE P.model NOT IN (

SELECT PCs.model FROM PCs UNION SELECT Laptops.model FROM Laptops

UNION

SELECT Printers.model from Printers) ;

**NOTE:** No violation found

C. Use CREATE FUNCTION and ALTER TABLE to implement the constraints into the database schema. Discuss the association of the functions created with the database relations and indicate the outcome for the sample data provided.

Due to the short coming of the mysql. I was not able to implement the create function and alter table on my MySQLWorkbeanch. So, that is why as professor said I am just giving the code for the create function and alter table without the sample output.

a)

/\* Constraint 1. C. a \*/

CREATE FUNCTION Constraint1Ca(@maker AS char, @type AS char(7)) RETURNS BIT

AS

BEGIN

DECLARE @Flag BIT=1

DECLARE @check INT = (select COUNT(\*) from Products)

IF @check = 0

BEGIN

SET @Flag = 1

RETURN @Flag

END

IF (@type = 'Laptop')

BEGIN

if(@maker IN (select maker from Products WHERE type = 'PC' ))

BEGIN

SET @Flag = 0

RETURN @Flag

END

END

RETURN @Flag

END

ALTER TABLE Products

ADD CONSTRAINT CheckConstraint1Ca CHECK(dbo.Constraint1Ca(maker, type)=1);

For this Constraint we want to check if the manufacture makes a PC, then they can’t make Laptops. To do so, first we check if the products table is empty or not, if it is then we can just add the data in the table but if its not empty then we scan through the table and check if any entries have a maker that is same as the maker of entry we want to insert and the “type” of the entry is Laptop or not.

b)

/\* Constraint 1. C. b \*/

CREATE FUNCTION Constraint1Cb(@maker AS char,@speed AS FLOAT, @type AS char(7)) RETURNS BIT

AS

BEGIN

DECLARE @Flag BIT= 1

IF (@type = 'PCs')

BEGIN

if(@maker IN (select maker from PCs))

BEGIN

IF (@speed < (select speed from PCs AS speed))

BEGIN

SET @Flag = 0

RETURN @Flag

END

END

END

RETURN @Flag

END

ALTER TABLE PCs

ADD CONSTRAINT CheckConstraint1Cb CHECK(dbo.Constraint1Cb(maker, speed, type)=1);

For the problem, we want a constraint that the manufactural of PCs should also manufacture Laptops that has at least as great a processor. We will start with the input of maker , speed and type as we want the maker of PC to also make Laptops we check both the Products table if there exist another entry with same maker but different type (i.e. type of Laptops). If it exists, then we compare the speed of the Laptops with the speed from the parameter and if its less then it means there is a violation of the constraint so we block that entry.

c)

/\* Constraint 1. C. c \*/

CREATE FUNCTION Constraint1Cc(@ram AS INT,@price AS FLOAT )

RETURNS BIT

AS

BEGIN

DECLARE @Flag BIT=1

If (@price > All(SELECT price FROM PCs WHERE ram < @ram ))

SET @Flag =0

RETURN @Flag

END

ALTER TABLE Laptops

ADD CONSTRAINT CheckConstraint1Cc CHECK(dbo.Constraint1Cc(ram,price)=1);

This constraint states that if a laptop has higher memory than a PC, then the laptop should also have higher price. To solve this constraint we first need to get all the entries from the PC table where the price of the laptop is greater than that of the price of PC. After getting all this PC entries we compare if any entries of the PC has higher price than that of the Laptop then we have a conflict and we stop user from inserting that data in the Laptops table.

d)

CREATE FUNCTION Constraint741d\_PC(@model AS INT)

RETURNS BIT

AS

BEGIN

DECLARE @Flag BIT=0

If (@model IN (select model from products WHERE type IN(‘PCs’)))

SET @Flag =1

RETURN @Flag

END

ALTER TABLE PCs

ADD CONSTRAINT CheckConstraint741d\_PC CHECK(dbo.Constraint741d\_PC(model)=1);

CREATE FUNCTION Constraint741d\_Laptops(@model AS INT)

RETURNS BIT

AS

BEGIN

DECLARE @Flag BIT=0

If (@model IN (select model from products WHERE type IN('Laptops')))

SET @Flag =1

RETURN @Flag

END

ALTER TABLE Laptops

ADD CONSTRAINT CheckConstraint741d\_Laptop CHECK(dbo.Constraint741d\_Laptops(model)=1);

CREATE FUNCTION Constraint741d\_Printer(@model AS INT)

RETURNS BIT

AS

BEGIN

DECLARE @Flag BIT=0

If (@model IN (select model from products WHERE type IN('Printers')))

SET @Flag =1

RETURN @Flag

END

ALTER TABLE Printers

ADD CONSTRAINT CheckConstraint741d\_Printer CHECK(dbo.Constraint741d\_Printer(model)=1);

Here we will create 3 different constraint each for PCs, Laptops, and Printers which will check if the entry that is about to be entered is already in the Products Table. If it finds that model in the Products table we also user to insert the data but if it is not in there then we raise an error stopping user to insert that entry into that specific table without first inserting into Products Table.

D. Explain how the implementation C of the constraints in B may change if the following database schema is used:

**PCs**(*maker*, *model, speed, ram, hd, price*)

**Laptops**(*maker*, *model, speed, ram, hd, screen, price*)

**Printers**(*maker*, *model, color, type, price*)

If the above database schema is used, the implementation of C will change since we do not have the Products table and in every constraint in C we are using the Product table to check the type of product. If we were to remove the Products table then we do not need to check the type of the products since the table name will itself represent the type of that product, on top of that the maker entry is also available within all tables so we do not need to join any tables to take information about a product. And lastly, for the B sub question d “If the relation Products mentions a model and its type, then the model must also appear in the relation appropriate to that type.” We do not need to check this constraint because of the absent of the Products table.