



DBMS

Lab 10

Rahul Barodia B20CS047

Q1)

Firstly we made some assumptions.

These assumptions are:

- Book_ID is unique for every record in Book_Details.
 - Author_ID is unique for every record in Author_Details.
 - Book_ID is unique for every record in Book__Purchase_Details.
 - Book_ID is unique with respect to Author_ID and Book.
 - Different books from different authors can have the same book name.
 - 2 Authors can have the same name but will have different author ids.
 - The domain of all attributes in all the tables is atomic values.
-

a)

The SQL query is

```

CREATE VIEW View1 AS
SELECT
    Author_Name, Book, Copies
FROM
    Author_Details AS AD,
    Book_Details AS BD,
    Book_Purchase_Details AS BPD
WHERE
    AD.Author_ID = BD.Author_ID
    AND
    BD.Book_ID = BPD.Book_ID
GROUP BY
    Book;

SELECT * FROM View1;

```

Output:

Author_Name	Book	Copies
Damasio	Self Comes to Mind	1
Minsky	Emotion Machine	2
Minsky	Society of Mind	2
Ramachandran	Phantoms in the Brain	2
Rowling	Fantastic Beasts and Where to Find Them	3
Rowling	Goblet of Fire_Harry Potter	3
Rowling	Philosopher's Stone_Harry Potter	3
Rowling	Prisoner of Azkaban_Harry Potter	3
Safina	Voyage of the Turtle	2
Safina	What Animals Think	2
Tolkien	Fellowship of the Rings_Lord of the Rings	3
Wodehouse	Wodehouse at the Wicket	1



b)

The select query takes significantly less time than VIEW creation + view retrieval, however, as observed, VIEWS can get updated on their own so for multiple updates views can be better for execution.

If we need only temporary results for a few times then simple select query should be used but for long run, once a view is made, its retrieval time is less than simple select as it acts like cache.

c)

The view will get updates accordingly if tuples are inserted or deleted from the main database.

The sql query for inserting tuple in main database is

```
INSERT INTO
    Book_Details(Author_ID,Book_ID,Book)
VALUES
    ("Da_002","Da002_Dig","Digital Fortress");

INSERT INTO
    Author_Details(Author_ID, Author_Name)
VALUES
    ("Da_002", "Dan Brown");

INSERT INTO
    Book_Purchase_Details(Book_ID, Purchase_Dt, Copies, Price)
VALUES
    ("Da002_Dig", "Sep 06, 2021", 26, 250);
```

The result of after insertion is

Author_Name	Book	Copies
Damasio	Self Comes to Mind	1
Minsky	Emotion Machine	2
Minsky	Society of Mind	2
Ramachandran	Phantoms in the Brain	2
Rowling	Fantastic Beasts and Where to Find Them	3
Rowling	Goblet of Fire_Harry Potter	3
Rowling	Philosopher's Stone_Harry Potter	3
Rowling	Prisoner of Azkaban_Harry Potter	3
Safina	Voyage of the Turtle	2
Safina	What Animals Think	2
Tolkien	Fellowship of the Rings_Lord of the Rings	3
Wodehouse	Wodehouse at the Wicket	1
Dan Brown	Digital Fortress	26

The sql query for deleting tuple in main database is

```

SET SQL_SAFE_UPDATES = 0;
DELETE
FROM
    Book_Details
WHERE
    Book = "Society of Mind";

```

The result of after deletion is

Author_Name	Book	Copies
Damasio	Self Comes to Mind	1
Minsky	Emotion Machine	2
Ramachandran	Phantoms in the Brain	2
Rowling	Fantastic Beasts and Where to Find Them	3
Rowling	Goblet of Fire_Harry Potter	3
Rowling	Philosopher's Stone_Harry Potter	3
Rowling	Prisoner of Azkaban_Harry Potter	3
Safina	Voyage of the Turtle	2
Safina	What Animals Think	2
Tolkien	Fellowship of the Rings_Lord of the Rings	3
Wodehouse	Wodehouse at the Wicket	1
Dan Brown	Digital Fortress	26

Thus, it can be clearly seen that when we insert or delete values from the main database then the changes are made in the view.

d)

No, I cannot update the above view.

The sql query is

```
UPDATE View1  
SET Book = "Da Vinci Code"  
Where Author_Name = "Minsky";
```

The result of query is

Error Code: 1288. The target table View1 of the UPDATE is not updatable

Explanation : The reason is the above view is non updatable. Non updatable views are those views which are made by joining values from multiple tables, but the views made from a single table are updatable. Only updatable views can be updated.

e)

The sql query is


```
CREATE TRIGGER trigger1 BEFORE
INSERT ON
    Book_Purchase_Details
FOR EACH ROW
SET
    new.Purchase_Price = new.Copies * new.Price;

INSERT INTO
    Book_Purchase_Details(Book_ID, Purchase_Dt, Copies, Price, Book_Value)
VALUES
    ("Da002_DaV", "Sep 06, 2021", 26, 250, 150);

SELECT * FROM Book_Purchase_Details;
```

The result of query is

Book_ID	Purchase_Dt	Copies	Price	Purchase_Price	Book_Value
Da001_Sel	Sep 1, 2021	1	50	50	505
Da001_Sel	Sep 7, 2021	5	50	250	505
Mi009_Emo	Sep 2, 2021	2	50	100	527
Mi009_Soc	Sep 1, 2021	2	50	100	531
Ra001_Pha	Sep 2, 2021	2	50	100	508
Ro015_Fan	Sep 1, 2021	3	50	150	523
Ro015_Fan	Sep 9, 2021	12	35	420	523
Ro015_Gob	Sep 1, 2021	3	50	150	526
Ro015_Phi	Sep 1, 2021	3	50	150	535
Ro015_Phi	Sep 10, 2021	20	75	1500	535
Ro015_Pri	Sep 1, 2021	3	50	150	545
Sa001_Voy	Sep 2, 2021	2	50	100	546
Sa001_Wha	Sep 2, 2021	2	50	100	516
To015_Fel	Sep 1, 2021	3	50	150	527
To015_Fel	Sep 12, 2021	9	55	495	527
Wo015_Wod	Sep 5, 2021	1	50	50	549
Da002_Dig	Sep 06, 2021	26	250	NULL	NULL
Da002_DaV	Sep 06, 2021	26	250	6500	150



f) Yes, triggers on views can be created using `INSTEAD OF` keyword in dbms. Unfortunately, MYSQL doesn't support triggers on views.

g)

```
CREATE TRIGGER stopAlterTrig  
ON Book_Purchase_Details  
FOR ALTER_TABLE AS  
BEGIN  
    PRINT 'Table alteration is not allowed'  
    ROLLBACK TRANSACTION  
END
```

In case we want to keep the field constraints fixed , that is , the data fields' constraints shouldn't be changed , then we can use DDL triggers

For example, if someone tries to set Purchase_Price to NOT NULL constraint , we can create a DDL trigger which fires on ALTER command on Book_Purchase_Details table and restricts the alter as well as displays a warning message : Table alteration is not allowed

h) The sql query is

```
CREATE TRIGGER trigger_Book_Val BEFORE
INSERT
  ON Book_Purchase_Details
FOR EACH ROW
SET
  new.Book_Value = (100 + SUBSTRING(new.Purchase_Dt, 5, 1)) * (new.Purchase_Price / new.Copies) / 100;

INSERT INTO
  Book_Purchase_Details(Book_ID, Purchase_Dt, Copies, Price)
VALUES
  ("XYZ", "Sep 8, 2021", 3, 39);

SELECT * FROM Book_Purchase_Details;
```

The result of query is

Book_ID	Purchase_Dt	Copies	Price	Purchase_Price	Book_Value
Da001_Sel	Sep 1, 2021	1	50	50	505
Da001_Sel	Sep 7, 2021	5	50	250	505
Mi009_Emo	Sep 2, 2021	2	50	100	527
Mi009_Soc	Sep 1, 2021	2	50	100	531
Ra001_Pha	Sep 2, 2021	2	50	100	508
Ro015_Fan	Sep 1, 2021	3	50	150	523
Ro015_Fan	Sep 9, 2021	12	35	420	523
Ro015_Gob	Sep 1, 2021	3	50	150	526
Ro015_Phi	Sep 1, 2021	3	50	150	535
Ro015_Phi	Sep 10, 2021	20	75	1500	535
Ro015_Pri	Sep 1, 2021	3	50	150	545
Sa001_Voy	Sep 2, 2021	2	50	100	546
Sa001_Wha	Sep 2, 2021	2	50	100	516
To015_Fel	Sep 1, 2021	3	50	150	527
To015_Fel	Sep 12, 2021	9	55	495	527
Wo015_Wod	Sep 5, 2021	1	50	50	549
Da002_Dig	Sep 06, 2021	26	250	NULL	NULL
Da002_DaV	Sep 06, 2021	26	250	6500	150
XYZ	Sep 8, 2021	3	39	117	42

i)

A DML trigger will be used for this as a DML trigger is used in response to a variety of Data Definition Language (DDL) events.

The sql query is

```
CREATE VIEW Result3 AS
SELECT Book_Id, sum(Purchase_Price) as Total_by_Book
FROM
    Book_Purchase_Details AS BPD
GROUP BY Book_Id;

SELECT * FROM Result3;
```

Book_Id	Total_by_Book
Da001_Sel	300
Mi009_Emo	100
Mi009_Soc	100
Ra001_Pha	100
Ro015_Fan	570
Ro015_Gob	150
Ro015_Phi	1650
Ro015_Pri	150
Sa001_Voy	100
Sa001_Wha	100
To015_Fel	645
Wo015_Wod	50
Da002_Dig	10000
Da002_DaV	6500
XYZ	117

j)

Yes, both after and before triggers can be used as we have already done this in the previous parts. We calculated purchase price and book value using before triggers and applied after trigger on the same values for calculating total prices.

Note that we can use them in different triggers, but not in the same trigger.

This can be seen from the trigger we used in question 1)h) which is an example of 'before' triggers.

Example for use of 'After' triggers is

```
CREATE TRIGGER Total_Price_trigger AFTER  
INSERT ON  
    Book_Purchase_Details  
FOR EACH ROW  
UPDATE Result  
SET Total = new.Purchase_Price + Total;
```

This trigger will output net expenditure of the library on all books combined.

Q2)

a)

Important assumptions:

Trigger should be triggered whenever a new application is received.

It should check the necessary conditions and then if accepted, insert value into the rank table.

Following is the Trigger:

```
create trigger Application_check
After INSERT on APPLICATION
for each row
Begin
    UPDATE Application A1
    FROM
        SELECT SUM(C.Credits) as Cred, AVG(E.Grade) as avg_score
        FROM
            APPLICATION A, COURSE C, EXAM E
        WHERE New.SID = EXAM.SID AND EXAM.CID = Course.CID
    IF Cred>45 AND avg_score>25
        Set A1.Status ="accepted"
        INSERT INTO RANK values(A1.SID, avg_score, Cred,
                                SELECT row_number() over (order by AVG DESC))
    ELSE
        Set A1.Status = "rejected"
End
```

Q3)

a)

Following is the trigger that maintains a symmetry in friendship relations. This ensures that if (X,Y) is deleted from the Friends table (Y,X) should also be deleted and vice versa.

```
create trigger f1_Del
after delete on Friend
for each row
when exists ( select * from Friends where ID1= Old.ID2 and ID2= Old.ID1 )
begin
    delete from Friend
    where ( ID1= Old.ID2 and ID2= Old.ID1 );
End
```

This is the Friend table:

ID1	ID2
2	5
5	2
3	2
7	5

Now I will delete row 1, with value 2,5.

ID1	ID2
3	2
7	5

After deleting 2,5

5,2 also gets deleted according to the trigger.

b) Following is the trigger whose function is to delete students that have graduated (their class increased by more than 12)

```
create trigger Graduation
After update of class on Student
for each row
when New.grade > 12
begin
    delete from Student
    where ID=New.ID; //Old.ID also works
end
```

c) Following is the trigger that increases the grade of all the friends of a student, whose class is increased by one.

```
create trigger friend_inc
After update of class on Student
for each row
begin
    UPDATE student
    INNER JOIN Friend ON
        Student.ID1 = Friend.ID2
    where Friend.ID1=New.ID1
    Set Student.class = Student.class+1
end
```