Homework 5

Name: Cameron Stark

Problem 1 (10 Points): What is the difference between isolation levels: **REPEATABLE READ** and **READ COMMITTED**

Repeatable Read is a higher level that if the query reads a result that was already read, it deals with uncommitted, whereas Read Committed that guarantees that no data that is uncommitted can be read.

Problem 2:

Part A (20 Points): The relation R(x) consists of a set of integers --- that is, one-component tuples with an integer component.

Alice's transaction is a query:

SELECT SUM(x) FROM R;

COMMIT;

Betty's transaction is a sequence of inserts:

INSERT INTO R VALUES(10);

INSERT INTO R VALUES(20);

INSERT INTO R VALUES(30);

COMMIT;

Carol's transaction is a sequence of deletes:

DELETE FROM R WHERE x=30;

DELETE FROM R WHERE x=20;

COMMIT;

Before any of these transactions execute, the sum of the integers in R is 1000, and **none of these integers are 10, 20, or 30**. If Alice's, Betty's, and Carol's transactions run at **about the same time**, and each of them runs under isolation level **READ COMMITTED**, which sums could be produced by Alice's transaction? Identify one of those sums from the list below and **explain the reason**.

a) 1020

b) 950

c) 980

d) 1060

Because the delete from statements, won't delete anything because the values don't exist in the relation until the commit action is performed.

Part B (20 Points): Given the same scenario as Part A, which sums could be returned by Alice's transaction if all three transactions run under isolation level READ UNCOMMITTED, but not if they run under isolation level SERIALIZABLE? Identify one of those sums from the list below and **explain the reason**.

a) 1030

b) 1060

[✓ c) 1020]

d) 1000

Because the 20 will be deleted and the 30 will be added resulting in 1020

Problem 3 (20 Points): Given the following relation:

```
Movie(
        title varchar(255),
        ID int PRIMARY KEY,
        year int,
        length int,
        studioName varchar(255),
        genre varchar(100)
        )
```

1. Write sql scripts to create a view called “DisneyComedies” with title, length, year as attributes. The DisneyComedies only allows end users to query movies from studio “Disney” with genre as “comedy”.

```
CREATE VIEW DisneyComedies AS (
        SELECT
                title,
                length,
                year
        FROM Movie
        WHERE
                studioName = ‘Disney’ AND
                genre = ‘comedy’
)
```

2. Write sql scripts to create single index on attributes **title** and **year** for table “Movie” using BTREE.

CREATE INDEX movieIndex ON Movie (title, year) USING BTREE;

Problem 4 (30 Points): Suppose we have two relations

```
CourseGrades (
        StudentID int,
        CourseID varchar(10),
        Credits int,
        Grades double
)
GPA as (
        StudentID int,
        TotalCredits int,

        GPA double

)
```

Write sql scripts to create a **trigger** “UpdateGpa”, which updates the GPA table for a student when the student has a new record inserted in the CourseGrades table. The update should include:

GPA = (GPA* TotalCredits+ Grades*credits)/(TotalCredits+ credits)

TotalCredits = TotalCredits + credits

```
delimiter ||
CREATE TRIGGER UpdateGpa AFTER INSERT ON CourseGrades
  FOR EACH ROW
  BEGIN
    UPDATE GPA set gpa = ((GPA * TotalCredits  + new.Grades * new.credits) / (TotalCredits + new.credits)) WHERE
studentid = new.studentid;
    UPDATE TotalCredits = TotalCredits + new.credits WHERE studentid = new.studentid;
  END
||
```