



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

**ACCC5001 Database Management
Systems
Assignment 2**

Alexandru Balasa
1814696

Lecture: **Koushik Modak**

Table of Contents

Introduction:.....	3
Report:.....	4
Task 2:.....	4
Conclusion:	14
Reference:	15

Introduction:

In this Database Management Systems assignment, I Applied well-established data analysis and design techniques as per my knowledge or understanding onto that topic to create prerequisites and determinations for information base application demonstrating an essential comprehension of the requirement for quality and information security. I learned about trigger its implementations and how it helps us to secure the database applications. Security is the major concern while adapting any application in this assignment user used to-rent-estate agent database and demand security onto that so that user aware from any ongoing activity.

In this assignment, I used Data definition level or data manipulation level trigger. I used SQL server 2019 for creating a table or running trigger also attached screenshot for the conclusion. In SQL Server basically, Triggers are executed after or before an INSERT, DELETE, or an UPDATE operation onto a table. We can also use these SQL triggers on Views, Schema or Tables if we want to perform any of the above-mentioned attributes. In case of a trigger, the most important point is that we can associate a trigger to a single table only at a time. In this assignment we have to make a report for task 2. Task 2 also includes some theoretical answer from these theoretical answers I understand about the transaction of database also know about the concurrency anomalies and how to resolve them. I explain all these answers with the help of example in the task 2 theory report. In task 1 of this assignment, I have to execute queries that were mentioned in the assignment in a SQLQUERY_1.sql file.

Report:

Task 2:

Part a:

The audit table stores the information like Audit ID, Date of transactions, User name who is performing Inserting, delete, Update operations onto the database to store these events information I make a details column and to shows the property that user /owner visit or sales I make a id column.

Query for inserting records into PRGPropertyAudit table:

Create table PRGPropertyAudit

```
(  
    AuditID int ,  
    id VARCHAR(10),  
    date_of_transaction VARCHAR(255),  
    user_name varchar(255),  
    details VARCHAR(255)  
)
```

I create a trigger that responds to Delete or Update on the sales table and for insert onto the visits. The DDL or DML trigger helps to know about DDL or DML transaction occurring into audit table because the audit table stores information about the date on which transaction occur also about the user name who is performing any DML actions like Insert, update or delete. While DDL statements include Create, Alter, Drop etc. that helps us to create table sales or audit.

This screen capture of PRGPropertyAudit table shows the detail of operations perform by user and various details that is obtained from trigger that is on insert in visit and delete or update on sales tables.

	AuditID	id	date_of_transaction	user_name	details
1	25	11	Nov 11 2020 6:03PM	dbo	Insert
2	11	11	Nov 11 2020 6:10PM	dbo	DELETE
3	16	16	Nov 11 2020 6:13PM	dbo	DELETE
4	10	25	Nov 11 2020 6:17PM	dbo	UPDATE

Part b:

Advantages to be gained from proposed solution:

In task 1 part a, I create a trigger that helps me to know about any table that is created into the database, alter into the database, drop from the database with its proper date time or user name. It helps property owner to track the records of their to-rent-estate agent database.

Query that I used for creating trigger:

```
use To_rent_DB
go
create or alter trigger to__rent_trigger on database for create_table, alter_table,
drop_table as
begin
    DECLARE @data XML
    DECLARE @DatabaseName VARCHAR(max)
    DECLARE @TSQLCommand VARCHAR(max)
    DECLARE @event_type VARCHAR(max)
    DECLARE @LoginName VARCHAR(max)
    DECLARE @SchemaName VARCHAR(max)
    DECLARE @PostTime VARCHAR(max)
    SET @data = EVENTDATA()
    insert into PRGDatabaseAudit (
        Database_Name,Table_Name,Event_Type,Login_Name,SQL_Command,
        Audit_Date_Time
    )
    values (
        @data.value('/EVENT_INSTANCE/DatabaseName')[1], 'nvarchar(100)'),
        @data.value('/EVENT_INSTANCE/SchemaName')[1], 'nvarchar(100)'),
        @data.value('/EVENT_INSTANCE/EventType')[1], 'nvarchar(100)'),
        @data.value('/EVENT_INSTANCE/LoginName')[1], 'nvarchar(100)'),
        @data.value('/EVENT_INSTANCE/TSQLCommand')[1], 'nvarchar(100)'
    ),
        @data.value('/EVENT_INSTANCE/PostTime')[1], 'nvarchar(100)')
);
end;
```

```

1  use To_rent_DB
2  go
3  create or alter trigger to__rent_trigger on database for create_table, alter_table, drop_table as
4  begin
5      DECLARE @data XML
6      DECLARE @DatabaseName VARCHAR(max)
7      Declare @TSQLCommand VARCHAR(max)
8      DECLARE @event_type VARCHAR(max)
9      DECLARE @LoginName VARCHAR(max)
10     DECLARE @SchemaName VARCHAR(max)
11     DECLARE @PostTime VARCHAR(max)
12     SET @data = EVENTDATA()
13     SET @DatabaseName = @data.value('(/EVENT_INSTANCE/EventType)[1]', 'nvarchar(100)')
14     -- print @data
15     --print @DatabaseName
16     insert into PRGDatabaseAudit (
17         Database_Name,Table_Name,Event_Type,Login_Name,SQL_Command,Audit_Date_Time
18     )
19     values (
20         @data.value('(/EVENT_INSTANCE/DatabaseName)[1]', 'nvarchar(100)'),
21         @data.value('(/EVENT_INSTANCE/SchemaName)[1]', 'nvarchar(100)'),
22         @data.value('(/EVENT_INSTANCE/EventType)[1]', 'nvarchar(100)'),
23         @data.value('(/EVENT_INSTANCE/LoginName)[1]', 'nvarchar(100)'),
24         @data.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'nvarchar(100)'),
25         @data.value('(/EVENT_INSTANCE/PostTime)[1]', 'nvarchar(100)')
26     );
27 end;

```

Commands completed successfully.

Commands completed successfully.

Total execution time: 00:00:00.005

In part b of task 1 I also create a table PRGDatabaseAudit that is responsible for tracking the records of following information:

Database_Name,Table_Name,Event_Type,Login_Name,SQL_Command,Audit_Date_Time

Query that I used for creating PRGDatabaseAudit table:

create table PRGDatabaseAudit (Database_Name VARCHAR(255),
Table_Name VARCHAR(255),Event_Type VARCHAR(255),Login_Name VARCHAR(255),SQL_Command VARCHAR(255),Audit_Date_Time VARCHAR(255))

	Database_Name	Table_Name	Event_Type	Login_Name	SQL_Command	Audit_Date_Time
1	To_rent_DB	dbo	CREATE_TABLE	DESKTOP-6A1ADSD\Zuco1	create table sid16(id int)	2020-11-10T17:00:39.413
2	To_rent_DB	dbo	DROP_TABLE	DESKTOP-6A1ADSD\Zuco1	drop table sid16	2020-11-10T17:04:00.110
3	To_rent_DB	dbo	CREATE_TABLE	DESKTOP-6A1ADSD\Zuco1	create table visit(id varch...	2020-11-11T10:42:56.420
4	To_rent_DB	dbo	CREATE_TABLE	DESKTOP-6A1ADSD\Zuco1	create table PRGPropertyAudi...	2020-11-11T12:08:17.363
5	To_rent_DB	dbo	DROP_TABLE	DESKTOP-6A1ADSD\Zuco1	drop table PRGPropertyAudit	2020-11-11T12:20:18.893
6	To_rent_DB	dbo	CREATE_TABLE	DESKTOP-6A1ADSD\Zuco1	create table PRGPropertyAudi...	2020-11-11T12:20:40.087
7	To_rent_DB	dbo	CREATE_TABLE	DESKTOP-6A1ADSD\Zuco1	create table Sales(id varch...	2020-11-11T12:27:08.063
8	To_rent_DB	dbo	DROP_TABLE	DESKTOP-6A1ADSD\Zuco1	drop table visit	2020-11-11T12:50:17.733
9	To_rent_DB	dbo	CREATE_TABLE	DESKTOP-6A1ADSD\Zuco1	create table visit(Audit_id...	2020-11-11T12:50:54.583

Answer for part a.

Transaction in DBMS:

Transaction as the name suggests it handles all transaction in database. Database transactions are performing read or write operations onto the database management system data objects so that user can view his transaction in proper way.

A transaction in DBMS follows ACID properties so that a transaction is completed without any inconsistency in database.

For understanding transaction I explained it with a simple example in which a student transfers some money into another student account. This very simple and small transaction involves several low-level tasks.

Student 1 Account Balance:

Amount = student_1.Amount

Current_Amount = Amount – 100

Student_1.Amount = Current_Amount

Quit_Account (Student_1)

Student 2 Account Balance:

Amount = student_2.Amount

Current_Amount = Amount + 100

Student_2.Amount = Current_Amount

Quit_Account (Student_2)

Concurrency in DBMS:

It takes care of all types of upgradation in data objects to provide non redundant or inconsistency experience to the end users. In DBMS multiple transactions are occur in parallel simultaneously at a time

concurrency control ensure that all type of data object updating in proper way.

To ensure concurrency in database it uses protocols that are classified into two categories namely:

1. Lock- based protocol
2. Timestamp- based protocol

Answer for part b.

ACID properties in Transaction with example.

ACID properties of transaction:

The word ACID itself consist of all the information why we used it in transaction. Each character of the word ACID demonstrates a purpose. The transaction is DBMS is certain and secure because of An ACID property of DBMS even now a day it is considered as major feature of Database management system.

Atomicity: It ensures either transaction is happening completely or not there is no intermediate stage regarding the transaction.

Example:

For understanding the atomicity behaviour consider, the arrangement of activities are

- Debited the measure of \$10 from Tom account.
- credited the measure of \$10 to Jerry account.

All activities in this set ought to be finished. On the off chance that the framework neglects to add the sum in Jerry account subsequent to deducting from Tom account, we need the system that reverts the operation on Tom account and Tom needs to make a transaction operation again.

Consistency: It ensures the stability of the database. The requirements that we put on the information worth should be consistent when the execution of the exchange. On the off chance that the framework falls flat on account of the invalid information while doing an activity, in this circumstance framework needs to return to its past state.

Example:

The aggregate sum in Tom and Jerry record ought to be the equivalent when the exchange. The amount of the cash in Tom and Jerry account when the exchange is \$20 for saving exchange consistency.

Isolation: This property guarantees that the execution, all things considered, ought to be detached from different exchanges. At a time multiple transactions are starting on a solitary information base, activity from any exchange ought not to meddle with activity in different exchanges.

Example:

If there is some other exchange (among Mac and Tom) going, it ought not to make any impact on the exchange among Tom and Jerry. Both of the exchanges ought to be confined.

Durability: The entirety of the above three properties ought to be fulfilled while the exchange in progress. Be that as it may, the sturdiness of the exchange is checked after the exchange is finished. The progressions made during the exchange should exist after fruition of the exchange.

In some cases, all the activities during the time spent the fruitful exchange might be finished however the framework flops right away. All things considered, changes are made while exchanges should persevere. Thus, the framework should re-visitation of its past stable state.

Example:

It might occur if a framework gets slammed after finishing of the apparent multitude of tasks or availability is a misfortune. All things considered, if the framework restarts it should protect the steady-state. A sum in Tom and Jerry record ought to be the equivalent when the framework gets a restart.

Answer for Part c.

Explanation for the three anomalies with example namely Dirty Read, Unrepeatable Read and Lost Update

Dirty Read:

This concurrency problem in transactions occurs when the data objects reads the data from an uncommitted transaction so it leads for inconsistency because it might be a chance with uncommitted transaction that they fail in later stage so they roll back to its previous stage.

Here, R stands for Read while W stands for write everywhere.

Example-

Transaction 1:	Transaction 2:
R(A) W(A) Failure	R(A) // Dirty Read (read value from uncommitted transaction done by A which fails in later stage) W(A) Commit

Unrepeatable Read:

This concurrency problem in transaction occurs whenever any other transactions trying to access the same value from a transaction more than one. In other word we can say that here transaction perform read operation repeatedly without updating its value in last transaction.

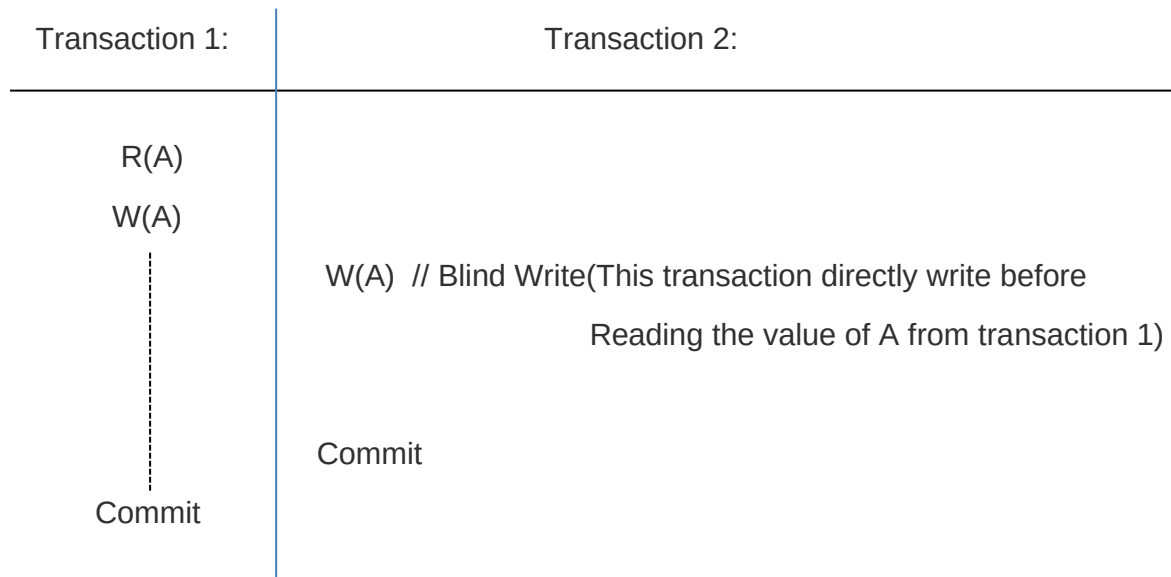
Example-

Transaction 1:	Transaction 2:
R(A)	R(A)
W(A)	R(A) // unrepeatable write

Lost Update:

This concurrency problem occurs in the situation when updates from one or more transactions get lost because multiple transactions occur simultaneously so it writes them blindly.

Example-



Answer for Part D.

Two phase locking:

This protocol is also known as 2PL. As the name suggests here locking and unlocking is done in two phase. In this a transaction acquires a lock when transaction is initiated and released a lock when transaction is completed. The two phases used by 2PL is:

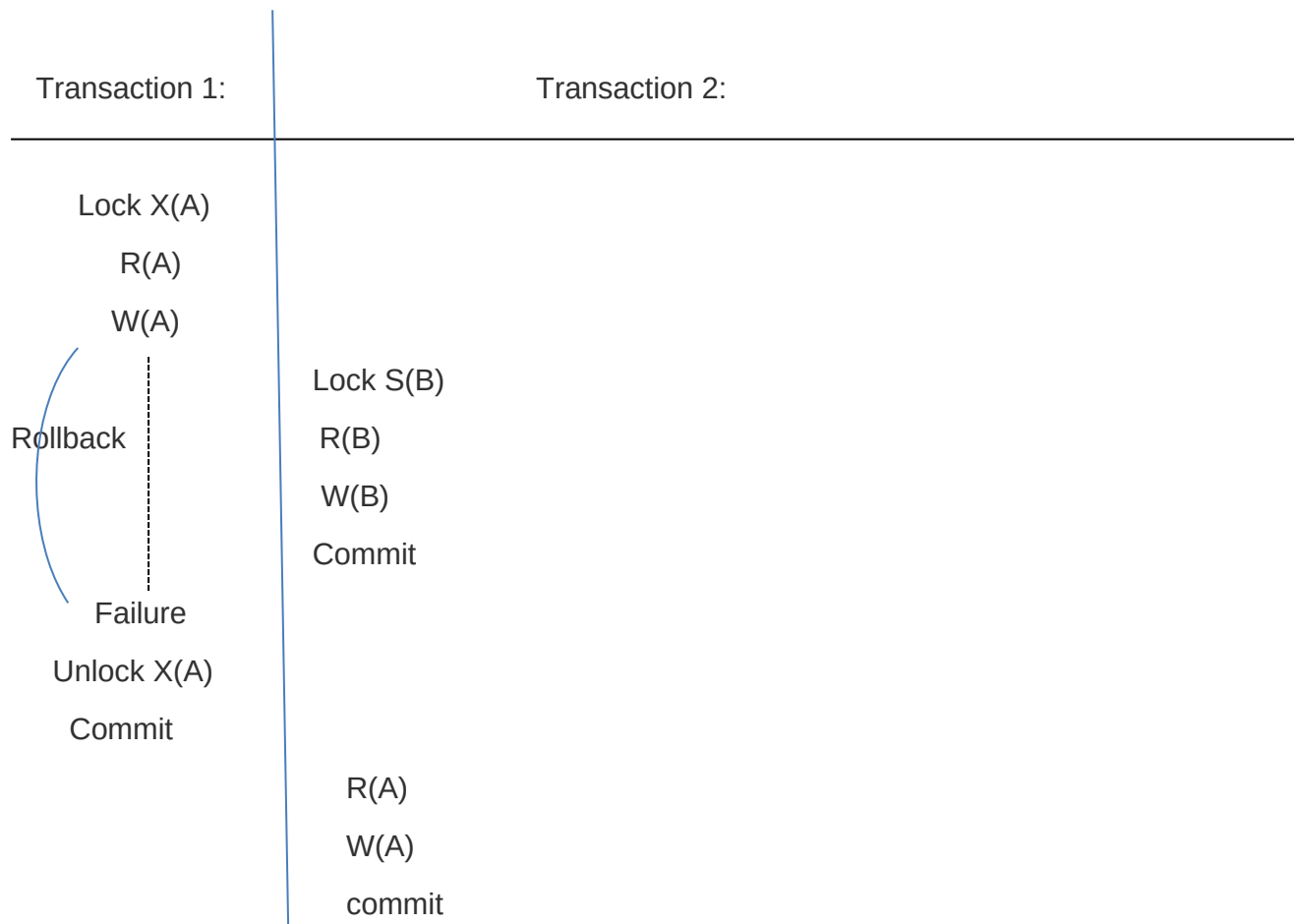
- Growing phase: in this phase locks are acquiring by a transaction.
- Shrinking phase: In this phase locks are releasing by a transaction.

The functioning of these two phase is fixed they can't adapt each other functionality.

2PL is used to achieve Serializability with the help of growing or shrinking phase.

2PL is used to solved the problem caused by dirty read with the help of shared S (A) or exclusive X(A) lock. Now, I used above dirty read example here.

In this example I explain how two phase locking protocol resolved dirty read issue.



Here we can see that the problem of dirty read is resolved because here on the failure, transaction 1 rollback and other transaction are not able to read its value at that time because it is executing all operation during his lock time. When transaction is fail it rollback to previous stable state and again try to execute if user wants otherwise it release his lock so that other transactions get the update. It leads to remove inconsistency that occur in dirty read.

Conclusion:

From this assignment I learned how we can track any ongoing changes into a database with the help of triggers in a table. I learned about trigger and see its benefit by executing its query into SQL server. I also learned about database transaction and its ACID properties. I also go through with various concurrency anomalies like Dirty read, Unrepeatable Read, Lost Update and deals with the two phase locking protocol to resolve the dirty read problem. I learned lots of new things from this assignment. This assignment helps me to understanding the security concepts. I learned lots of things from this assignment. This assignment was not easy for me I tried a lot for completing this assignment as of my best.

Reference:

Babu, R. (2019, May, 15). Triggers in SQL Server. Retrieved from <https://www.sqlshack.com/triggers-in-sql-server/>

Chapple, M. (2020, Jan, 03). The ACID Database Model. Retrieved from <https://www.lifewire.com/the-acid-model-1019731>

Factor, P. & Page, R. (2007, May, 25). SQL Server 2005 DDL Trigger Workbench. Retrieved from <https://www.red-gate.com/simple-talk/sql/t-sql-programming/sql-server-2005-ddl-trigger-workbench/>

Le, T. (2015, Jan, 12). Transactions and Concurrency Control in DBMS. Retrieved from <https://www.linkedin.com/pulse/transactions-concurrency-control-dbms-thiensi-le>

Javaram, P. (2018, Aug, 22). How to set up a DDL and DML SQL Server database transactional replication solution. Retrieved from <https://www.sqlshack.com/how-to-set-up-a-ddl-and-dml-sql-server-database-transactional-replication-solution/>