

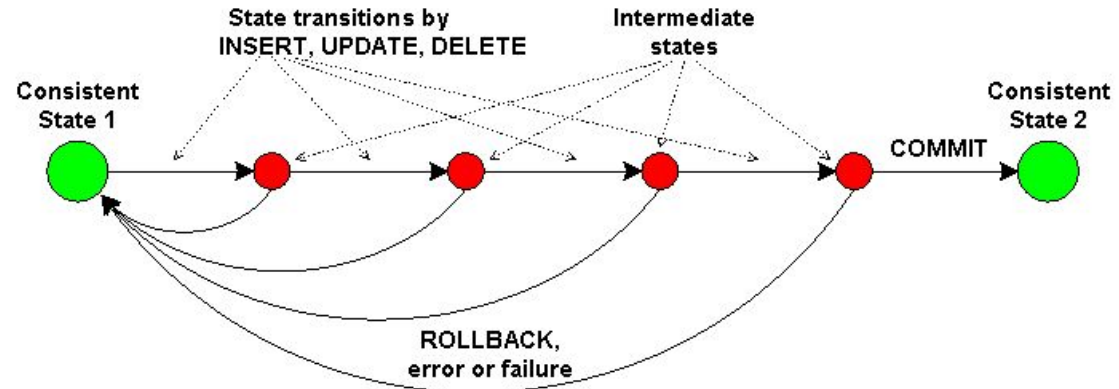
Databases - Tutorial 9

Concurrency Control- Transactions

Hamza Salem - Innopolis University

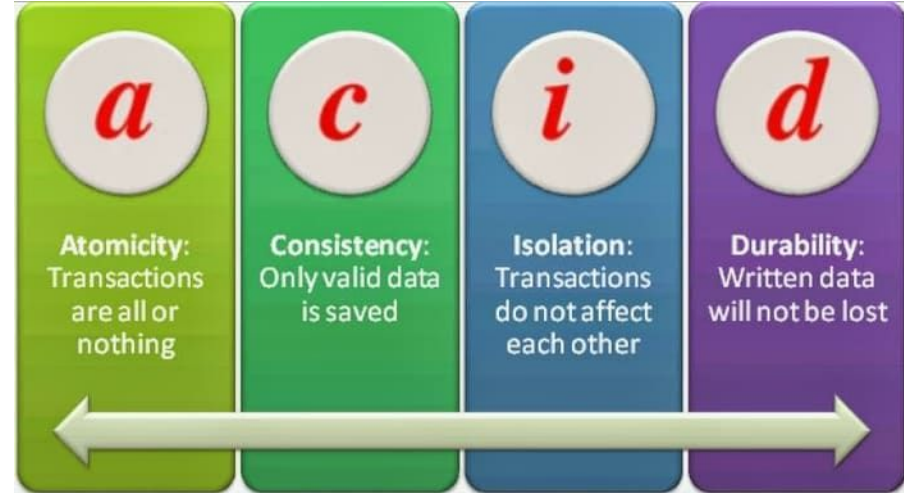
What is a transaction?

- A transaction is a **set of operations performed by an application that transfers a database from one correct state to another correct state (consistency)**, provided that the transaction is completed (**atomicity**) and without interference from other transactions (**isolation**).



Transactions

- **Atomicity** – ensures that all operations within the work unit are completed successfully. Otherwise, the transaction is aborted at the point of failure and all the previous operations are rolled back to their former state.
- **Consistency** – ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation** – enables transactions to operate independently of and transparent to each other.
- **Durability** – ensures that the result or effect of a committed transaction persists in case of a system failure.



Read Phenomena



DIRTY READ

A transaction **reads** data written by other concurrent **uncommitted** transaction

NON-REPEATABLE READ

A transaction **reads** the **same row twice** and sees different value because it has been **modified** by other **committed** transaction

PHANTOM READ

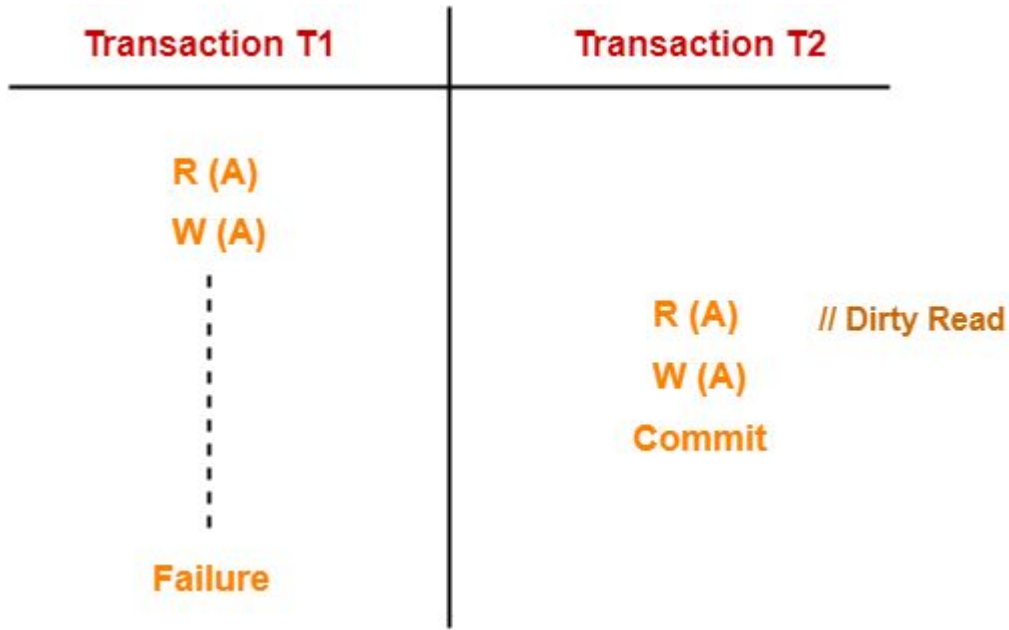
A transaction **re-executes** a query to **find rows** that satisfy a condition and sees a **different set** of rows, due to changes by other **committed** transaction

SERIALIZATION ANOMALY

The result of a **group** of concurrent **committed transactions** is **impossible to achieve** if we try to run them **sequentially** in any order without overlapping

Non-repeatable read vs Phantom read

- User A runs the same query twice.
- In between, User B runs a transaction and commits.
- Non-repeatable read: The A row that user A has queried has a different value the second time.
- Phantom read: All the rows in the query have the same value before and after, but **different rows are being selected** (because B has deleted or inserted some).



1. T1 reads the value of A.
2. T1 updates the value of A in the buffer.
3. T2 reads the value of A from the buffer.
4. T2 writes the updated the value of A.
5. T2 commits.
6. T1 fails in later stages and rolls back.

Transaction T1	Transaction T2
R (X)	
	R (X)
W (X)	
	R (X) // Unrepeated Read

1. T1 reads the value of X (= 10 say).
2. T2 reads the value of X (= 10).
3. T1 updates the value of X (from 10 to 15 say) in the DB.
4. T2 again reads the value of X (but = 15).

Transaction T1	Transaction T2
R (X)	
	R (X)
Delete (X)	
	Read (X)

Phantom Read Problem

1. T1 reads X.
2. T2 reads X.
3. T1 deletes X.
4. T2 tries reading X but does not find it.



4 Standard Isolation Levels



American National Standards Institute - ANSI

Low

1

2

3

4

High



READ UNCOMMITTED

Can see data written by
uncommitted transaction

READ COMMITTED

Only see data written by
committed transaction

REPEATABLE READ

Same read query always
returns same result

SERIALIZABLE

Can achieve same result
if execute transactions
serially in some order
instead of concurrently

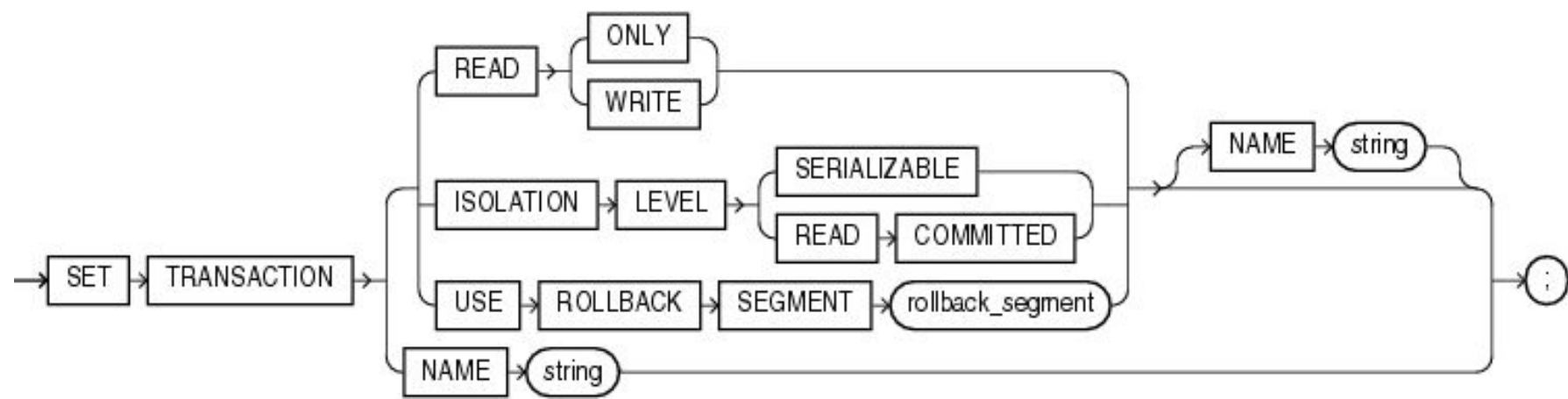
Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

ERROR: could not serialize access due to read/write dependencies among transactions

Read committed vs Repeatable read

Read committed is an isolation level that guarantees that any data read was committed at the moment it is read. It simply restricts the reader from seeing any intermediate, uncommitted, 'dirty' read. It makes no promise whatsoever that if the transaction re-issues the read, it will find the same data, as data is free to change after it was read.

Repeatable read is a higher isolation level, that in addition to the guarantees of the read committed level, it also guarantees that any data read cannot change, if the transaction reads the same data again, it will find the previously read data in place, unchanged, and available to read.



Transaction Control

The following commands are used to control transactions.

- **COMMIT** – to save the changes.
- **ROLLBACK** – to roll back the changes.
- **SAVEPOINT** – creates points within the groups of transactions in which to ROLLBACK.
- **SET TRANSACTION** – Places a name on a transaction +isolation .

```
SET TRANSACTION transaction_mode [, ...]
```

```
ISOLATION LEVEL { SERIALIZABLE |  
REPEATABLE READ | READ COMMITTED |  
READ UNCOMMITTED }
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

```
SQL> DELETE FROM CUSTOMERS
      WHERE AGE = 25;
SQL> COMMIT;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
3	kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

```
SQL> DELETE FROM CUSTOMERS
```

```
WHERE AGE = 25;
```

```
SQL> ROLLBACK;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

```
SQL> SAVEPOINT SP1;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=1;
1 row deleted.
SQL> SAVEPOINT SP2;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=2;
1 row deleted.
SQL> SAVEPOINT SP3;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=3;
1 row deleted.
```

```
SQL> ROLLBACK TO SP2;
Rollback complete.
```






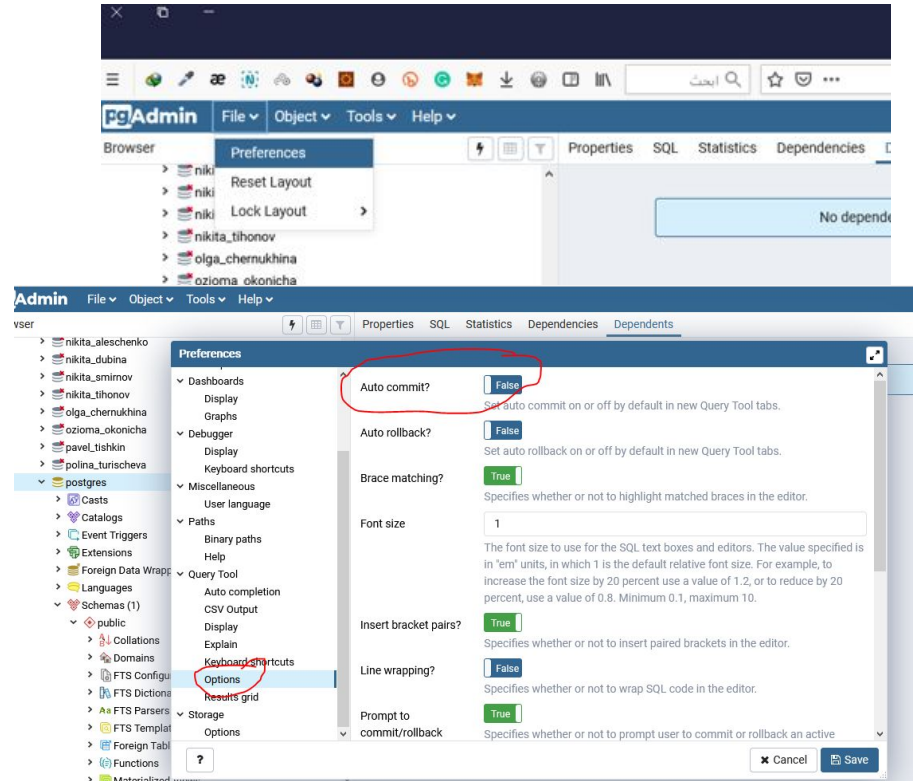
```
SQL> SELECT * FROM CUSTOMERS;
```

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

```
6 rows selected.
```

How to deactivate auto commit in pgAdmin?

- File  preference
- Query tool  Options  Auto commit (False)



Useful Links

- <https://www.tutorialspoint.com/sql/sql-transactions.htm>
- <https://habr.com/en/company/postgrespro/blog/467437>
- <https://www.gatevidyalay.com/concurrency-problems-in-transaction/>
- <https://www.postgresqltutorial.com/postgresql-show-tables/>
- <https://www.enterprisedb.com/postgres-tutorials/how-work-postgresql-transactions>
- <https://develloppaper.com/postgresql-set-transaction-snapshot/>
- <https://www.postgresql.org/docs/current/transaction-iso.html>