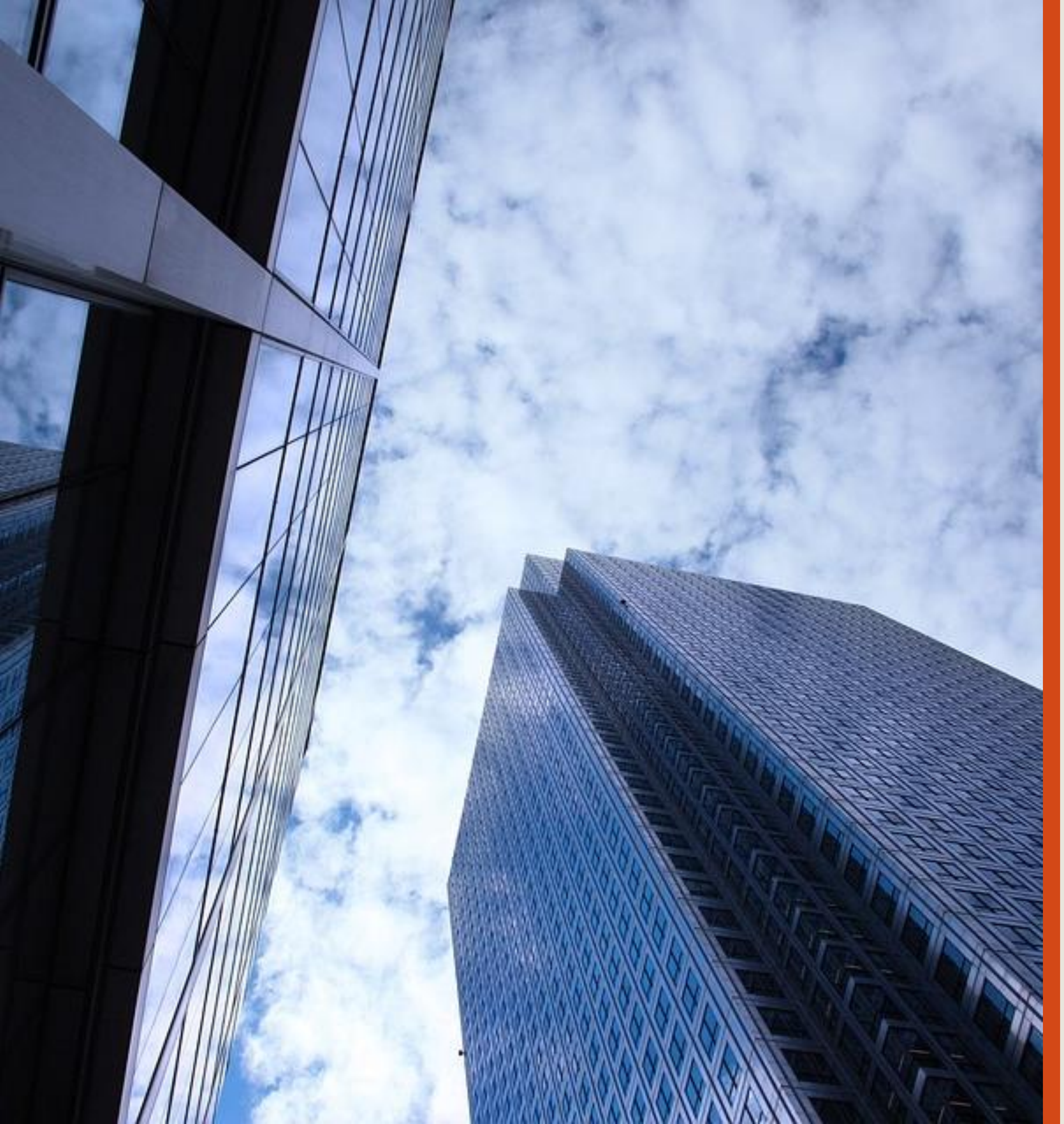


SQL

PART 06

Jayathma Chathurangani

ejc@ucsc.cmb.ac.lk

OUTLINE

✓ **TCL**

- **Transactions**
- **ACID Properties**
- **Related Keywords**

✓ **DCL**

- **Related Keywords**
- **Authorization Identifiers And Ownership**
- **Privileges**

✓ **Other areas for your Knowledge**

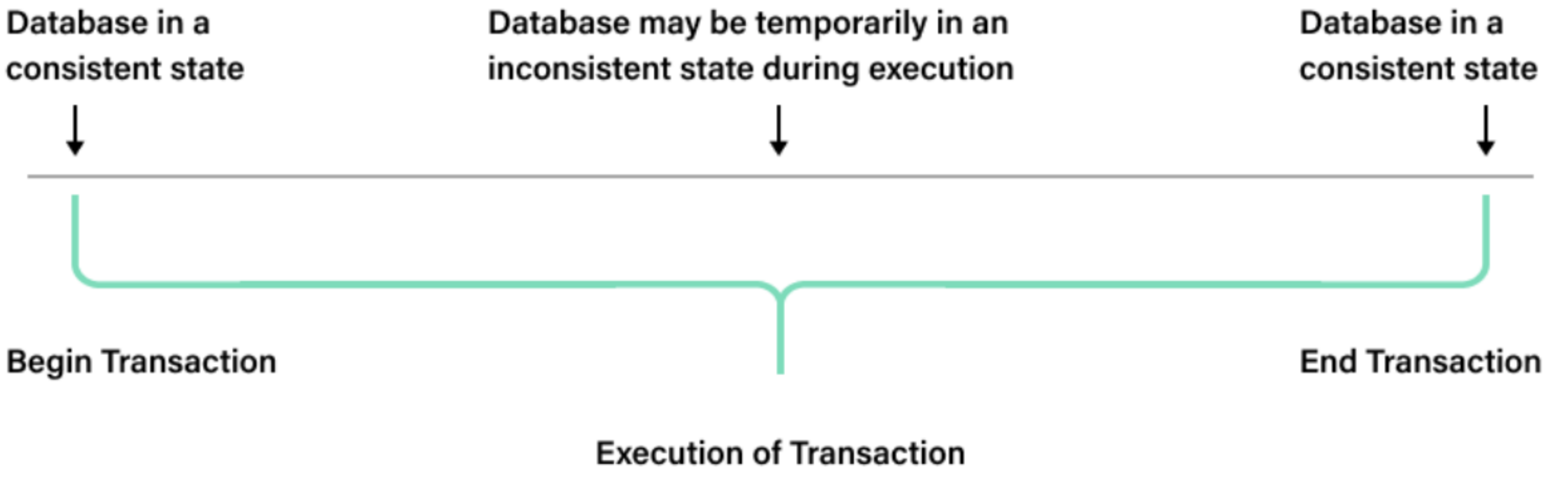
- **DML and DQL**
- **Imperative Vs Declarative**
- **SQL Vs PLSQL**

1

TCL (TRANSACTION CONTROL LANGUAGE)

1.1 TRANSACTIONS

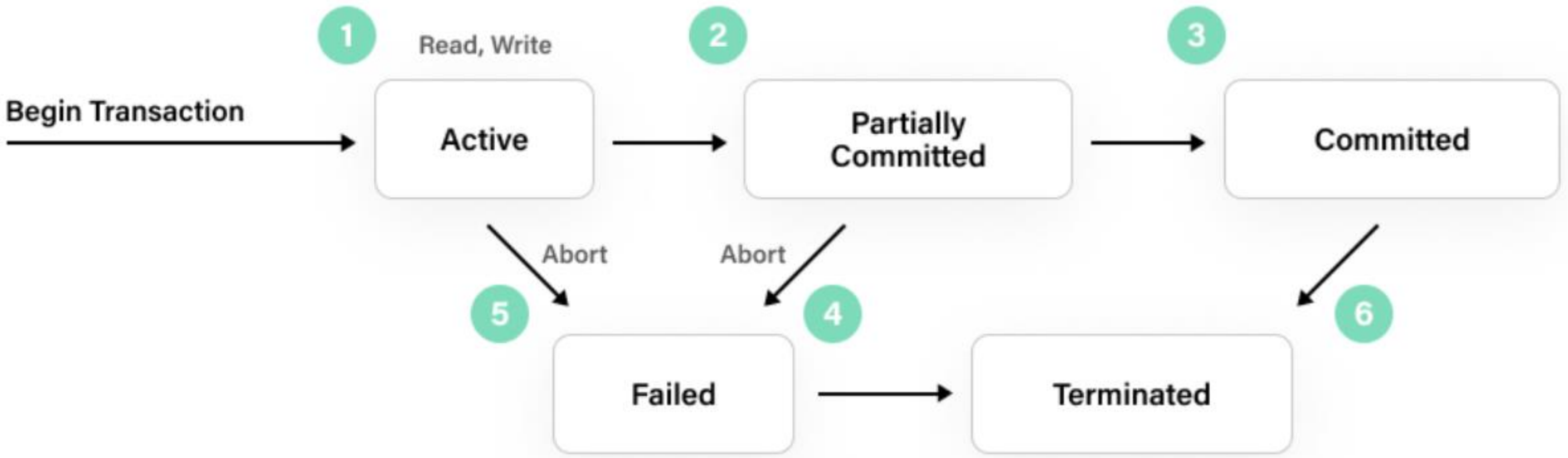
- A **transaction** is a single logical unit of work that accesses and possibly modifies the contents of a database.
- Transactions access data using read and write operations.


Database in a consistent state
Database may be temporarily in an inconsistent state during execution
Database in a consistent state
Begin Transaction
End Transaction
Execution of Transaction

1.2 STATES IN TRANSACTION LIFECYCLE

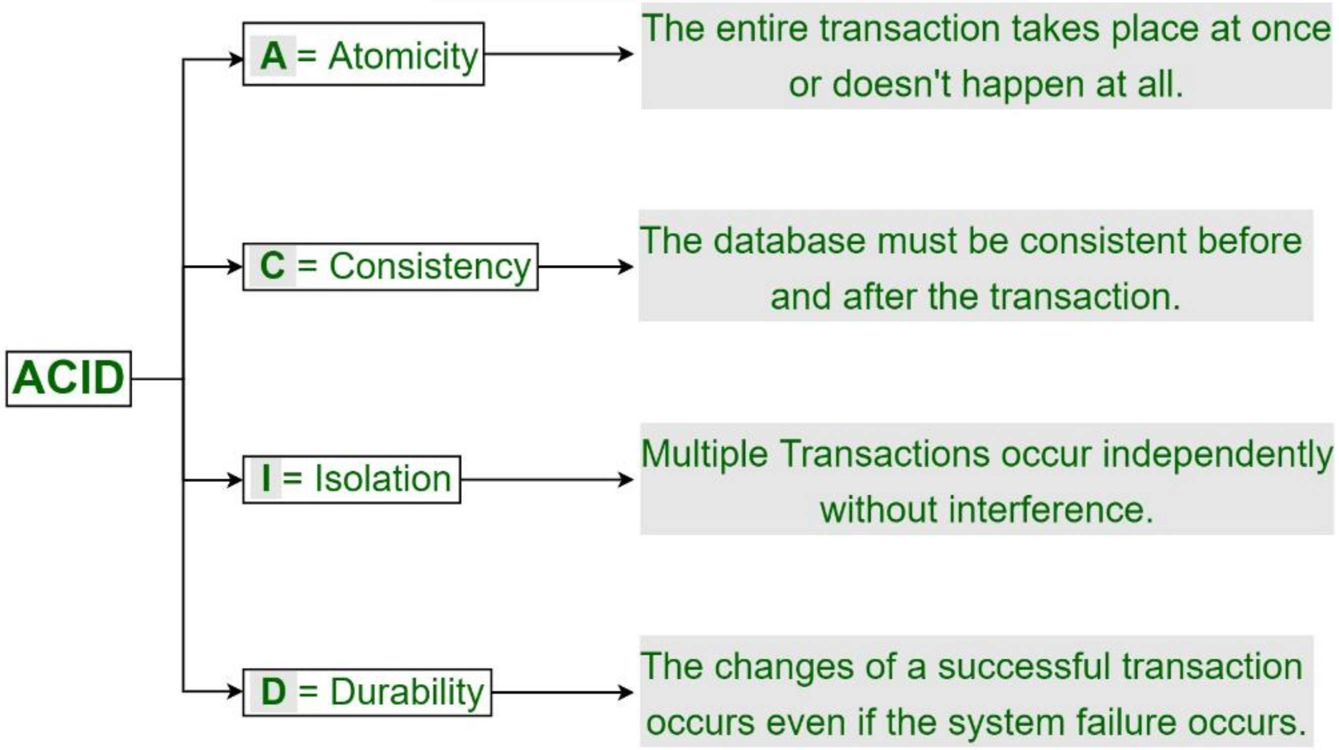
- **Active states**: It is the first state during the execution of a transaction. A transaction is active as long as its instructions (read or write operations) are performed.
- **Partially committed**: A change has been executed in this state, but the database has not yet committed the change on disk. In this state, data is stored in the memory buffer, and the buffer is not yet written to disk.
- **Committed:** In this state, all the transaction updates are permanently stored in the database. Therefore, it is not possible to rollback the transaction after this point.
- **Failed:** If a transaction fails or has been aborted in the active state or partially committed state, it enters into a failed state.
- **Terminated state:** This is the last and final transaction state after a committed or aborted state. This marks the end of the database transaction life cycle.

1.2 STATES IN TRANSACTION LIFECYCLE (CONTINUED)



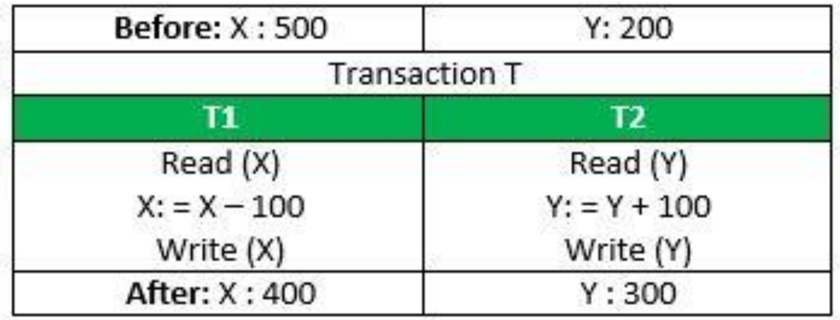
1.3 TRANSACTIONS AND ACID PROPERTIES

- In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.


ACID
A = Atomicity
The entire transaction takes place at once or doesn't happen at all.
C = Consistency
The database must be consistent before and after the transaction.
I = Isolation
Multiple Transactions occur independently without interference.
D = Durability
The changes of a successful transaction occurs even if the system failure occurs.

1.3.1 ATOMICITY

- By this, we mean that either the entire transaction takes place at once or doesn’t happen at all.
- There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all.
- **If the transaction fails, the entire transaction is rolled back. Atomicity prevents partial and incomplete transactions.**
- It involves the following two operations.
  - **Abort**: If a transaction aborts, changes made to the database are not visible.
  - **Commit**: If a transaction commits, changes made are visi



Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

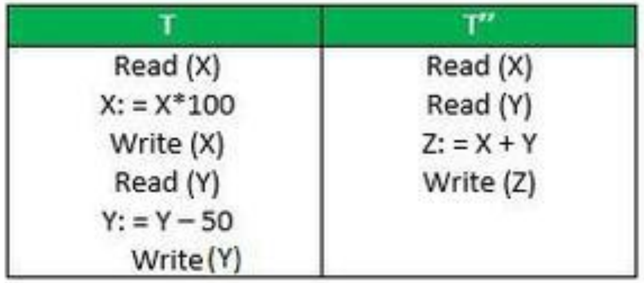
1.3.2 CONSISTENCY

- Consistency means that integrity constraints must be maintained so that the database is consistent before and after the transaction.
- **It refers to the correctness of a database.**
- **Consistency enforcement prevents data corruption and invalid entries.**
- Referring to the example,
  - The total amount before and after the transaction must be maintained.

  Total before **T** occurs = **500 + 200 = 700**.

  Total after **T** occurs = **400 + 300 = 700**.

  - Therefore, the database is consistent.
- Inconsistency occurs in case T1 completes but T2 fails. As a result, T is incomplete.



T	T’’
Read (X)	Read (X)
X: = X*100	Read (Y)
Write (X)	Z: = X + Y
Read (Y)	Write (Z)
Y: = Y – 50	
Write(Y)	

1.3.3 ISOLATION

- This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of the database state.
- Transactions occur independently without interference.
- **Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed.**
- This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.
- **Isolation maintains the independence of database transactions. Uncommitted transactions are isolated with locking mechanisms to prevent dirty reads** (Reading uncommitted data from other transactions) **or lost updates ().**
- Example: Let **X**= 500, **Y** = 500. Consider two transactions **T** and **T”.** Overwriting another transaction's uncommitted updates

1.3.4 DURABILITY

- This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs.
- These updates now become permanent and are stored in non-volatile memory.
- The effects of the transaction, thus, are never lost.
- **Durability ensures that transactions, once committed, will survive permanently.**
- Failed hardware, power loss, and even database crashes will not undo committed transactions due to durability support.
- *Durability is achieved with database backups, transaction logs, and disk storage*.

1.4 TCL

- The standard specifies that an SQL transaction automatically begins with a transaction-initiating SQL statement executed by a user or program (for example, SELECT, INSERT, UPDATE).
- Changes made by a transaction are not visible to other concurrently executing transactions until the transaction completes.
- **A transaction can complete in one of four ways:**
  - A **COMMIT** statement ends the transaction successfully, making the database changes permanent. A new transaction starts after COMMIT with the next transaction-initiating statement.
  - A **ROLLBACK** statement aborts the transaction, backing out any changes made by the transaction. A new transaction starts after ROLLBACK with the next transaction-initiating statement.
  - For programmatic SQL, successful program termination ends the final transaction successfully, even if a COMMIT statement has not been executed.
  - For programmatic SQL, abnormal program termination aborts the transaction.

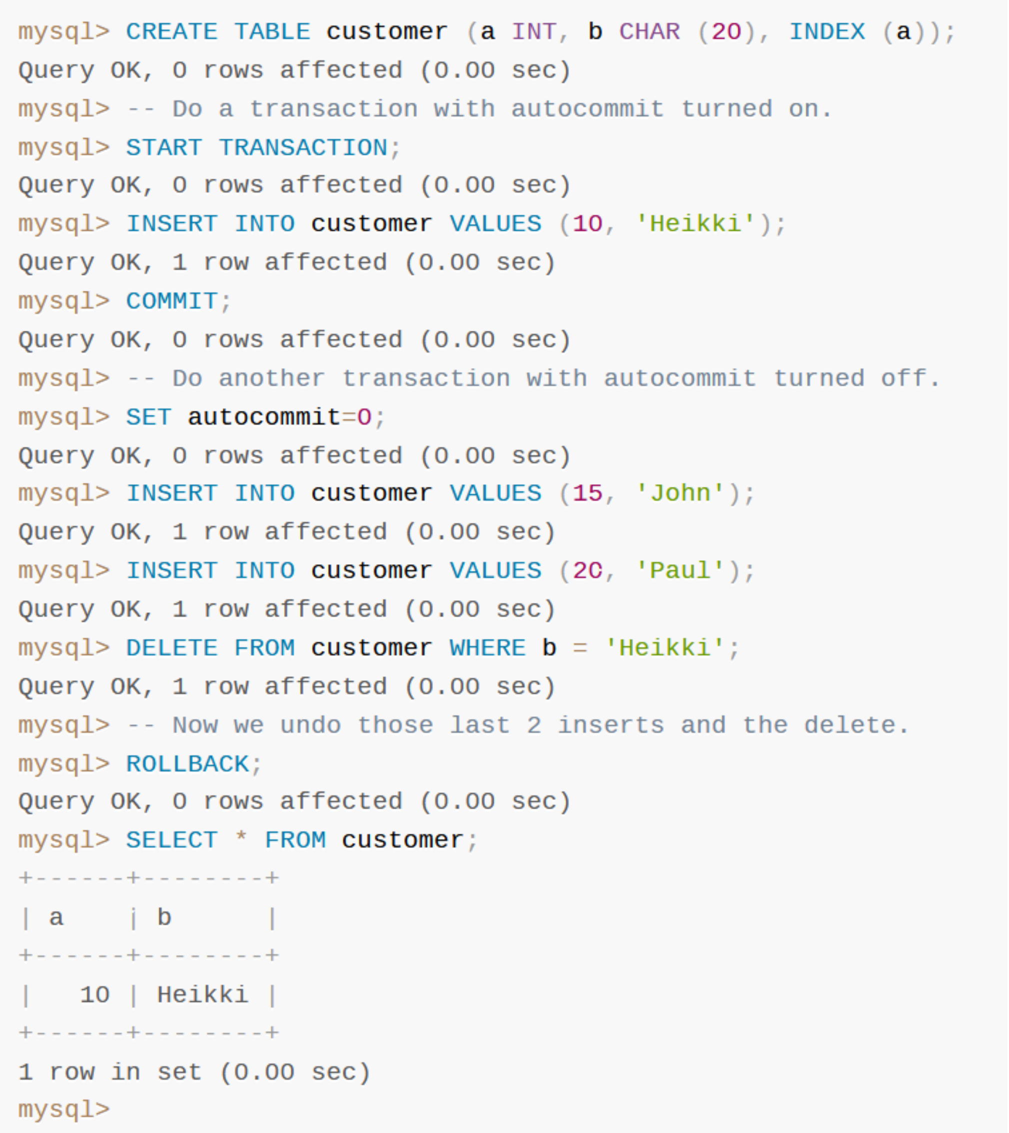
1.4 TCL (CONTINUED)

- **TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.**
- These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.
- Here are some commands that come under TCL:
  - **COMMIT** - Commit command is used to save all the transactions to the database.
  - **ROLLBACK** - Rollback command is used to undo transactions that have not already been saved to the database. (Reverts the changes after the last commit)
  - **SAVEPOINT** - It is used to roll the transaction back to a certain point without rolling back the entire transaction.
  - **AUTOCOMMIT** - This means that, when not otherwise inside a transaction, *each statement is atomic, as if it were surrounded by START TRANSACTION and COMMIT*. You cannot use ROLLBACK to undo the effect; however, if an error occurs during statement execution, the statement is rolled back.

1.4 TCL (CONTINUED)

- Consider the SQL statements executed in the mysql console.

•**Auto-commit** transfers all changes that you make immediately to the database.
•**Manual commit** requires your confirmation before committing a change to the database or rolling it back.

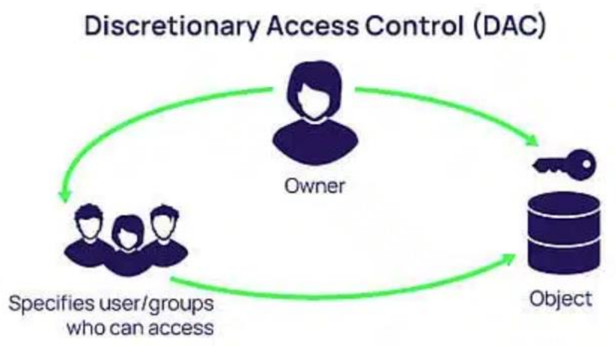


```
mysql> CREATE TABLE customer (a INT, b CHAR (20), INDEX (a));
Query OK, 0 rows affected (0.00 sec)
mysql> -- Do a transaction with autocommit turned on.
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO customer VALUES (10, 'Heikki');
Query OK, 1 row affected (0.00 sec)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
mysql> -- Do another transaction with autocommit turned off.
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO customer VALUES (15, 'John');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO customer VALUES (20, 'Paul');
Query OK, 1 row affected (0.00 sec)
mysql> DELETE FROM customer WHERE b = 'Heikki';
Query OK, 1 row affected (0.00 sec)
mysql> -- Now we undo those last 2 inserts and the delete.
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT * FROM customer;
+------+--------+
| a    | b      |
+------+--------+
|   10 | Heikki |
+------+--------+
1 row in set (0.00 sec)
mysql>
```

2

DCL (DATA CONTROL LANGUAGE)

2.1 WHAT IS IT?


Discretionary Access Control (DAC)
Owner
Object
Specifies user/groups who can access

- SQL supports only discretionary access control (DAC) through the GRANT and REVOKE statements.
- In DAC, each system object (file or data object) has an owner, and each initial object owner is the subject that causes its creation. Thus, an object's access policy is determined by its owner.
- **GRANT:** It is used to give user access privileges to a database.
  - **GRANT SELECT, UPDATE ON** MY_TABLE **TO** SOME_USER, ANOTHER_USER;
  - Eg- GRANT SELECT ON Users TO 'Tom'@'localhost;
- **REVOKE:** It is used to take back permissions from the user.
  - **REVOKE** privilege_name**ON** object_name**FROM** {user_name |PUBLIC |role_name}
  - Eg- **REVOKE SELECT, UPDATE ON** student **FROM** BCA, MCA;
- The mechanism is based on the concepts of authorization *identifiers, ownership, and privileges*
- *(To create a user:* **CREATE USER** user_name **IDENTIFIED BY** password)

2.2 AUTHORIZATION IDENTIFIERS AND OWNERSHIP

- An authorization identifier is a normal SQL identifier that is used to establish the identity of a user.
- Each database user is assigned an authorization identifier by the DBA.
- **Usually, the identifier has an associated password, for obvious security reasons.**
- Every SQL statement that is executed by the DBMS is performed on behalf of a specific user.
- The authorization identifier is used to determine which database objects the user may reference and what operations may be performed on those objects.
- Each object that is created in SQL has an owner.
- The owner is identified by the authorization identifier defined in the **AUTHORIZATION** clause of the schema to which the object belongs

2.3 PRIVILEGES

- Privileges are the actions that a user is permitted to carry out on a given base table or view.
- The privileges defined by the ISO standard are:
  - SELECT—the privilege to retrieve data from a table;
  - INSERT—the privilege to insert new rows into a table;
  - UPDATE—the privilege to modify rows of data in a table;
  - DELETE—the privilege to delete rows of data from a table;
  - REFERENCES—the privilege to reference columns of a named table in integrity constraints;
  - USAGE—the privilege to use areas like domains

ACTIVITY:

1) Database Language That Allows You To Access Or Maintain Data In Database
a) DCL b) DML c) DDL d) All of the Mentioned
2) A transaction completes its execution is said to be
a) Committed b) Aborted c) Rolled back d) Failed

3) A Database Language Concerned With The Definition Of The Whole Database Structure And Schema Is ________
a) DCL b) DML c) DDL d) All of the Mentioned

4) Which of the following keyword is used with Data Control Language (DCL) statements?
a) SELECT b) INSERT c) DELETE d) GRANT

5) What Is TRUE About SAVEPOINT?
a) Following the completion of a transaction, it must be executed to save all the operations performed in the transaction.
b) A transaction can be rolled back to its last saved state.
c) A specific part of a transaction can be given a name
d) None of the above

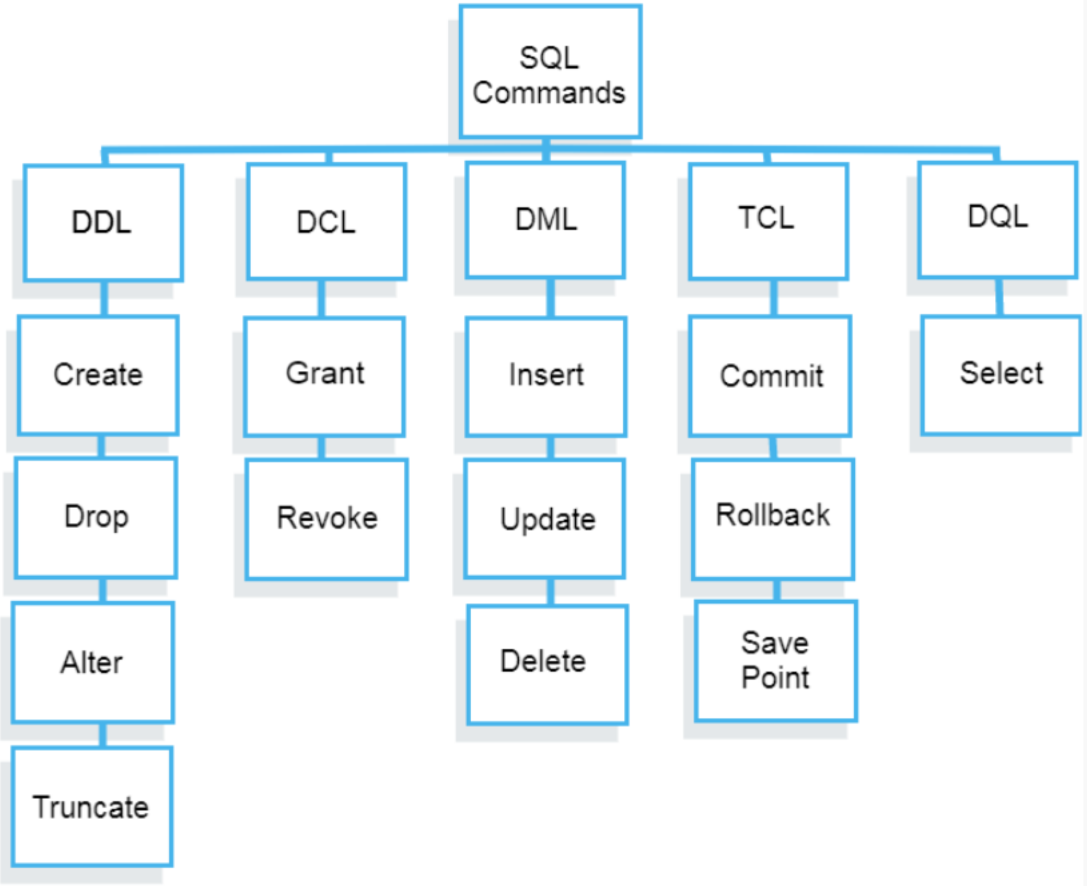
2

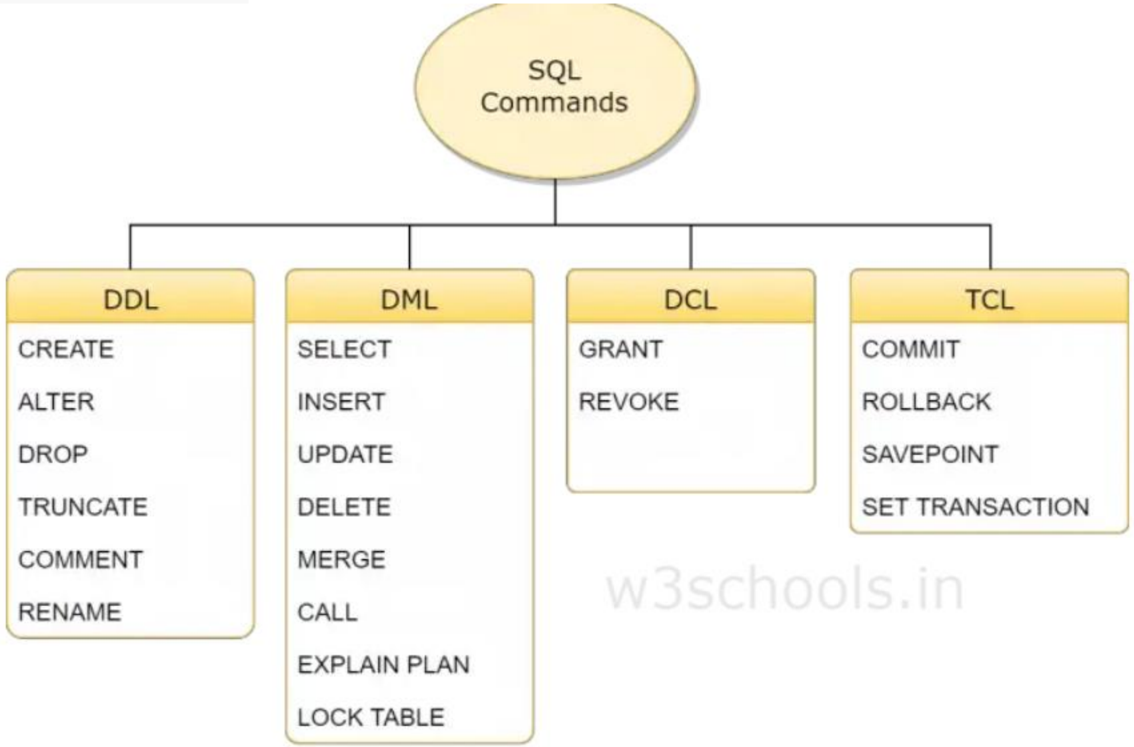
FOR YOUR KNOWLEDGE

3.1 DML AND DQL

You might find some refers SELECT under DQL not in DML. Refer below.

- **DML** (Data Manipulation Language) commands are used to modify the database.
  - It is responsible for all form of changes in the database.
  - The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.
  - Here are some commands that come under DML: INSERT, UPDATE, DELETE
- DQL (Data Query Language) is used to fetch the data from the database.
  - It uses only one command: SELECT


SQL Commands
DDL
DCL
DML
TCL
DQL
Create
Drop
Alter
Truncate
Grant
Revoke
Insert
Update
Delete
Commit
Rollback
Save Point
Select


SQL Commands
DDL
CREATE
ALTER
DROP
TRUNCATE
COMMENT
RENAME
DML
SELECT
INSERT
UPDATE
DELETE
MERGE
CALL
EXPLAIN PLAN
LOCK TABLE
DCL
GRANT
REVOKE
TCL
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION

3.2 IMPERATIVE VS DECLARATIVE

- There are several sub-paradigms of the **imperative programming paradigm**, such as the procedural or the object-oriented programming paradigms.

- *In the imperative programming paradigm, you describe the algorithm step-by-step, at various degrees of abstraction.*

- Examples of programming languages which support the procedural paradigm:
  - C (and most other legacy languages),PLSQL

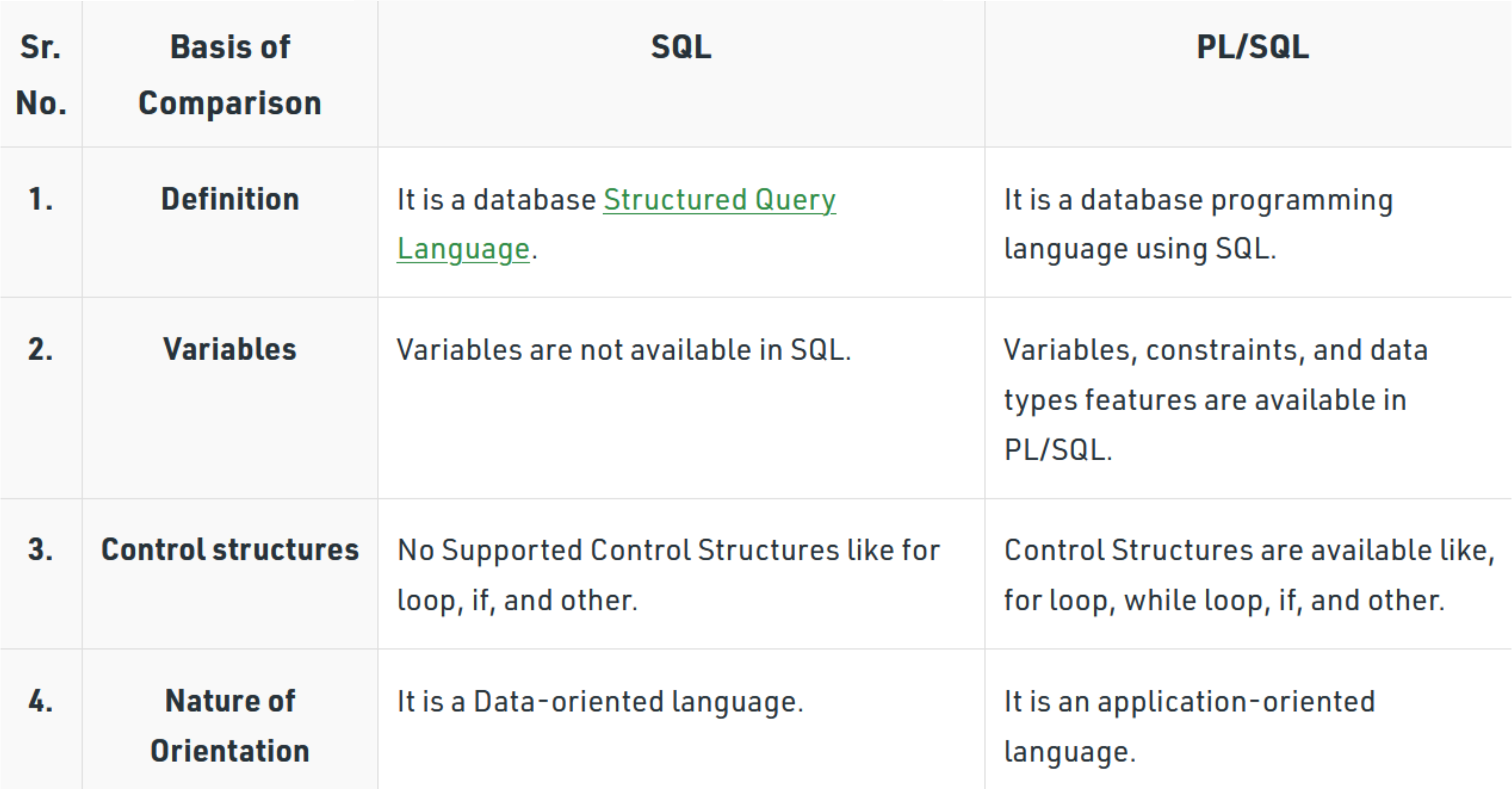
- There are several sub-paradigms of the **declarative programming paradigm**, such as the functional or the logic programming paradigms.

- *In the declarative programming paradigm, you describe a result or a goal, and you get it via a "black box". The opposite of imperative.*

- Examples of programming languages which support the declarative programming paradigm:
  - Yacc,Treetop,SQL

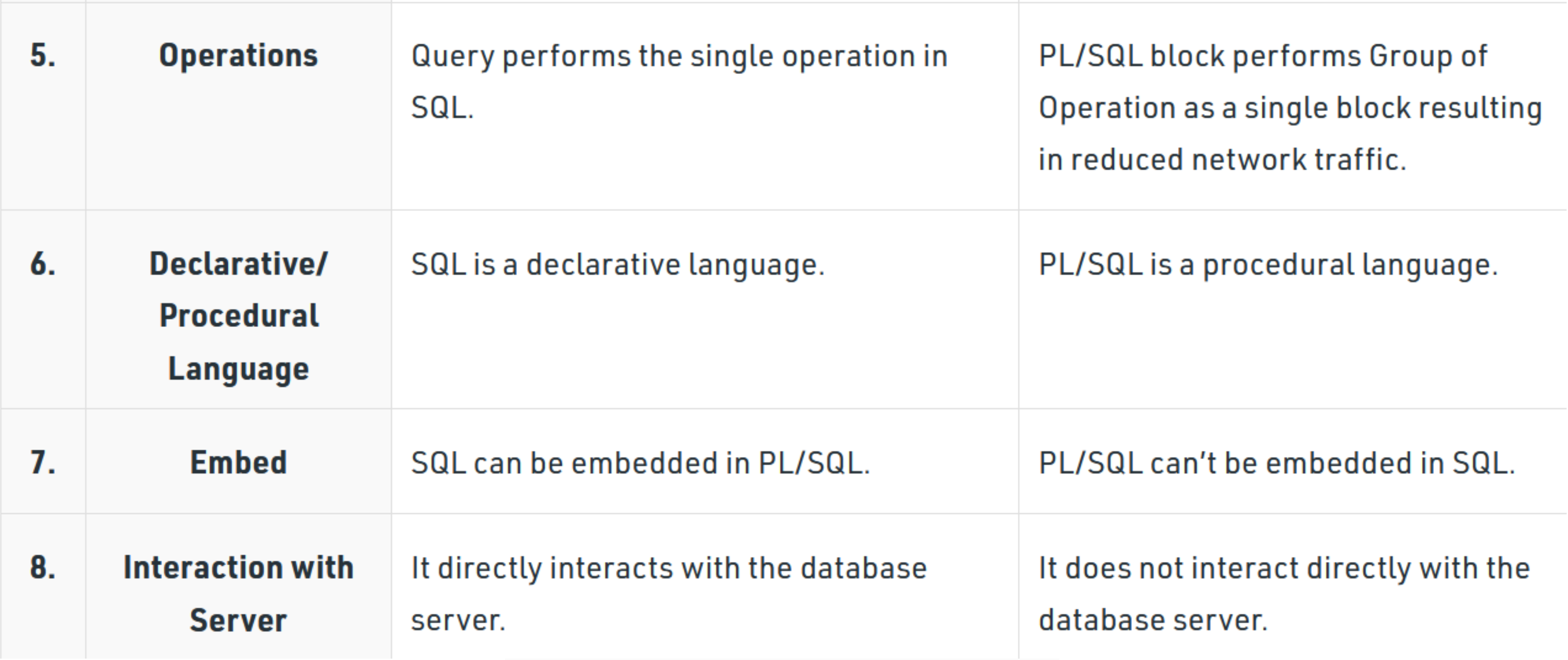
3.3 DIFFERENCE BETWEEN SQL AND PLSQL

The **PL/SQL** (Stands for Procedural Language extensions to SQL) programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database.



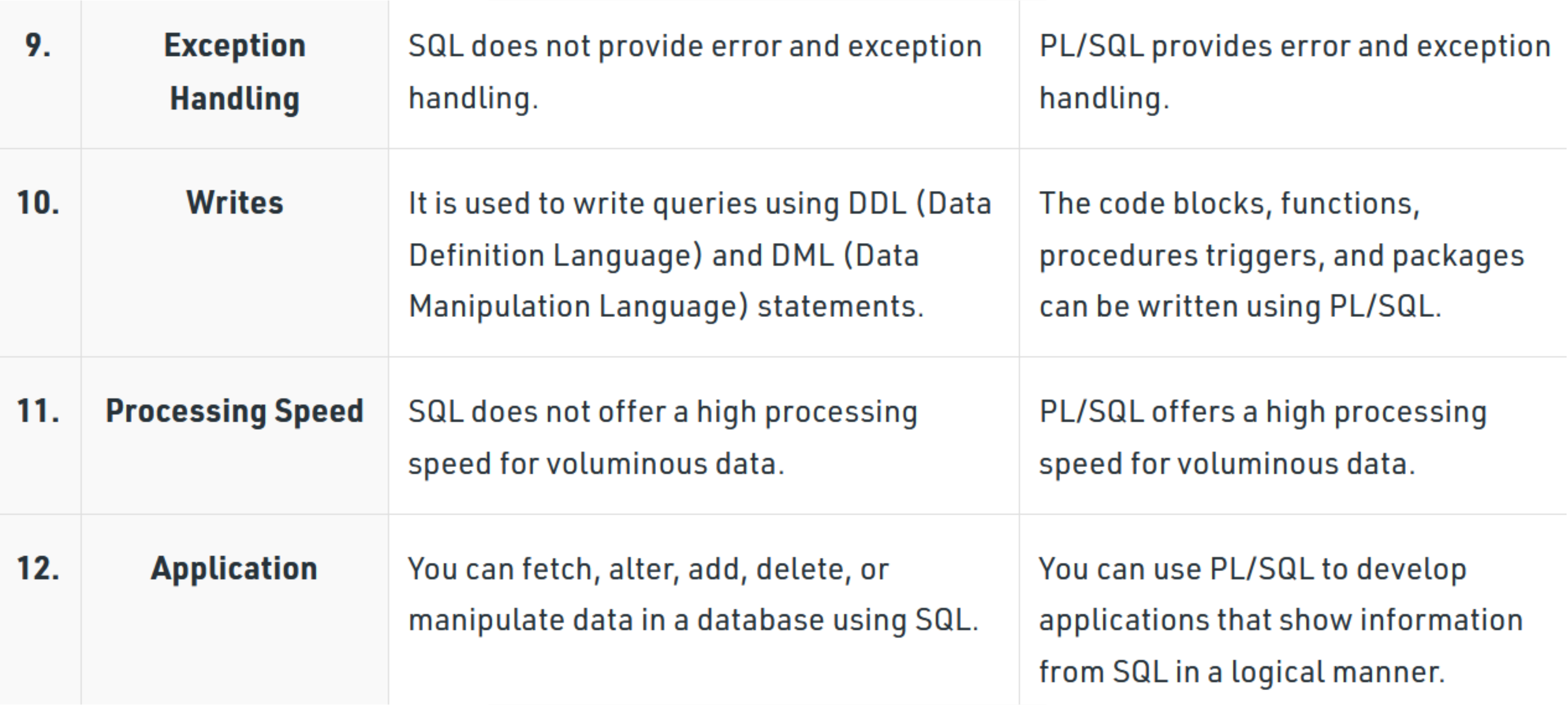
Sr. No.	Basis of Comparison	SQL	PL/SQL
1.	Definition	It is a database Structured Query Language.	It is a database programming language using SQL.
2.	Variables	Variables are not available in SQL.	Variables, constraints, and data types features are available in PL/SQL.
3.	Control structures	No Supported Control Structures like for loop, if, and other.	Control Structures are available like, for loop, while loop, if, and other.
4.	Nature of Orientation	It is a Data-oriented language.	It is an application-oriented language.

3.2 DIFFERENCE BETWEEN SQL AND PLSQL (CONTINUED)



5.	**Operations**	Query performs the single operation in SQL.	PL/SQL block performs Group of Operation as a single block resulting in reduced network traffic.
6.	**Declarative/ Procedural Language**	SQL is a declarative language.	PL/SQL is a procedural language.
7.	**Embed**	SQL can be embedded in PL/SQL.	PL/SQL can't be embedded in SQL.
8.	**Interaction with Server**	It directly interacts with the database server.	It does not interact directly with the database server.

3.2 DIFFERENCE BETWEEN SQL AND PLSQL (CONTINUED)



9.	**Exception Handling**	SQL does not provide error and exception handling.	PL/SQL provides error and exception handling.
10.	**Writes**	It is used to write queries using DDL (Data Definition Language) and DML (Data Manipulation Language) statements.	The code blocks, functions, procedures triggers, and packages can be written using PL/SQL.
11.	**Processing Speed**	SQL does not offer a high processing speed for voluminous data.	PL/SQL offers a high processing speed for voluminous data.
12.	**Application**	You can fetch, alter, add, delete, or manipulate data in a database using SQL.	You can use PL/SQL to develop applications that show information from SQL in a logical manner.

WRAP UP