Introduction to DBMS

Agenda

- What is DBMS?
- Why DBMS?
- RDBMS
- DataBase Terminologies

What is DBMS?

- DBMS stand for Database Management Systems
- These are systems to store, retrieve or ,sometimes, manipulate data
- Developed to handle large amount of data

Why DBMS?

- Consider a bank that maintains customer's account details, employee details, bank device details, etc.
- This details needs to be stored in such a way that it can be added, deleted, updated and retrieved from one place
- DBMS is a software designed for this type of operations

RDBMS

What is RDBMS?

- RDBMS stand for Relational Database Management Systems
- RDBMS allows to store, retrieve or manipulate data, but in a more efficient way than DBMS
- Apart from rows and columns the RDBMS table has following components
    - Domain
    - Instance
    - Schema
    - Keys

Database Terminologies

A database table

- A database consists of one or more tables
- A table is the most significant component in an RDBMS
- A table is where all data is stored
- A table constitutes of rows & columns
- Each column represents attributes of the entity

A database table

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

A record in a table

- Each row in a table is a record/tuple
- Each record is all of the information for each object, say a person or a product

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

A column in a table

- Each column in a table is an attribute
- This gives one piece of information about the attribute. For example, last name of a customer

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

Instance

- In RDBMS, there are lot of changes taking place in a table, over time
- Data get inserted, manipulated and deleted in parallel
- The data stored in a database at a particular moment of time is called instance

Keys

- A key is a data item (a column or a set of column) to uniquely identify a record in a table
- It is used to fetch a single or a set of records from a table
- Keys can also provide several types of useful constraints. For example, a unique key constraint can help avoid enter a duplicate value

Data Types

Built-in Data Types

- Numeric: TINYINT INT, BIGINT
- Floating numbers: DECIMAL, FLOAT
- Strings: CHAR, VARCHAR, BLOB, TEXT
- Date and Time: DATE, TIME, DATETIME

*Every relational database has its own **maximum and minimum size limit** for different-different data types which we do not need to remember. The objective is to have an idea of what data type to use in specific conditions.*

Numeric – Data Types

Datatype	From	To
Tinyint	0	255
int	-2,147,483,648	2,147,483,647
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

Floating Numbers

Datatype	From	To
Decimal	-10^38 +1	10^38 -1
Float	-1.79E + 308	1.79E + 308

Strings

DataType	Description
CHAR	Fixed length (Max - 8,000 characters)
VARCHAR	Variable length storage (Max – 8,000 characters)
BLOB	For binary large objects
TEXT	Variable length storage (Max Size – 1GB data)

Date and Time

Datatype	Description
DATE	Stores date (format YYYY-MM-DD)
TIME	Stores time (format HH:MI:SS)
DATETIME	Stores information of date and time (format YYYY-MM-DD HH:MI:SS)

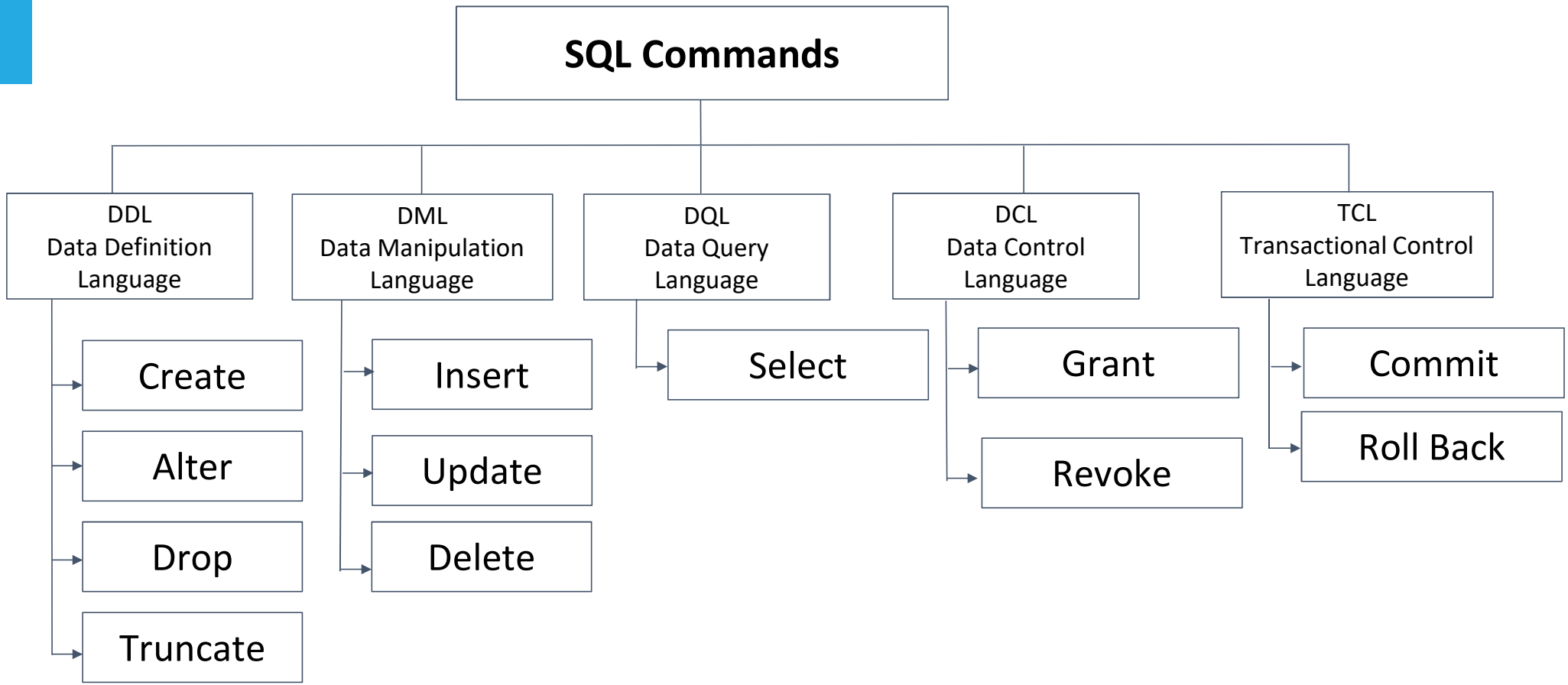
Getting started with Structured Query Language (SQL)

SQL Introduction

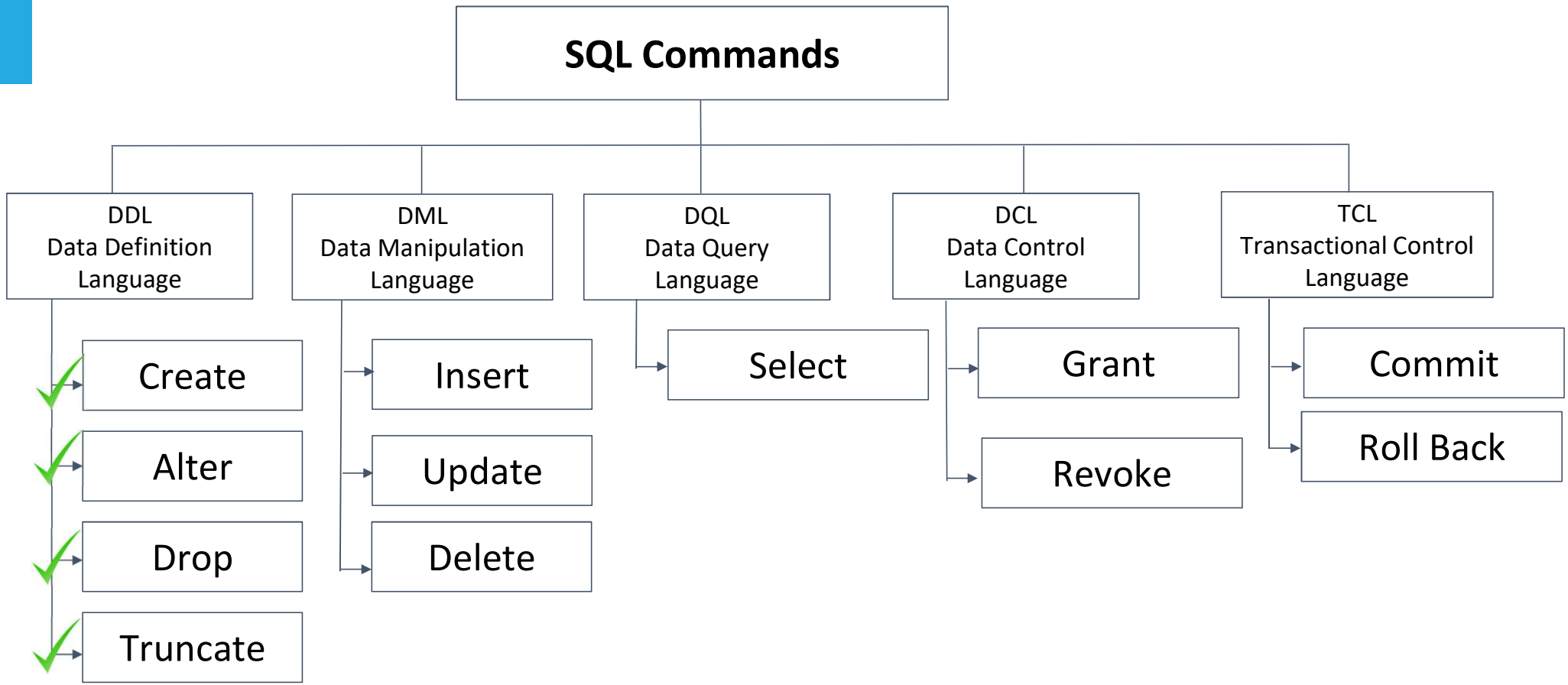
The commands available in SQL can be broadly categorised as follows:

- Data Definition Language (DDL)
- Data Manipulation Language(DML)
- Data Query Language (DQL)
- Data Control Language (DCL)
- Transactional Control Language (TCL)

Types of SQL Commands



Types of SQL Commands



Data Definition Language (DDL)

Data Definition Language (DDL)

- A database is a collection of many tables, and a database server can hold many of these databases

  Database Server —> Databases —> Tables (defined by columns) —> Rows

- Databases and tables are referred to as database objects

- Any operation, such as ***creating***, ***modifying***, or ***deleting*** database objects, is called **Data Definition Language (DDL)**

Data Definition Language (DDL)

- DDL is used to create a new schema as well as to modify an existing schema

- The typical commands available in DDL are:
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

Create Database

DDL - CREATE DATABASE- Syntax

- The CREATE DATABASE statement is used to create a new SQL database

Syntax:

```
CREATE DATABASE databasename;
```


The semicolon character (;) is a SQL statement terminator

- To create tables in the database you need to first select the database. Use the following syntax to select the database:

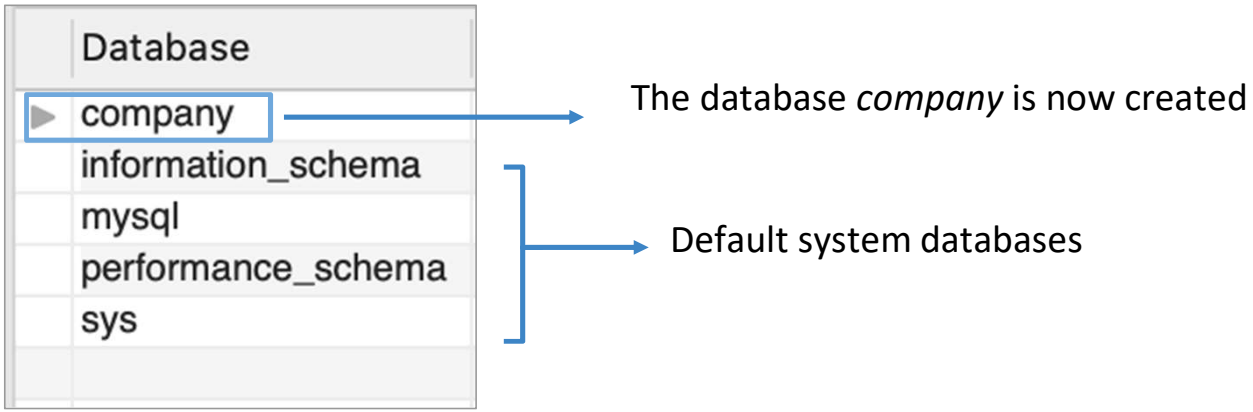
```
USE databasename;
```

DDL - CREATE DATABASE - Example

- We will create a database called company

```
CREATE DATABASE company;
```

- Use **SHOW DATABASES** to check if the database *company* has been **CREATED**



Drop Database

DDL - DROP DATABASE - Syntax

- The DROP DATABASE statement is used to drop an existing SQL database

Syntax:

```
DROP DATABASE databasename;
```


Database name

DDL - DROP DATABASE - Example

- The following SQL statement drops the existing database "company":

```
DROP DATABASE company;
```

- Use **SHOW DATABASES** to check if the database *company* has been **DROPPED**

Database
information_schema
mysql
performance_schema
sys

We see the database *company* is now dropped

Create Tables

Prerequisites for Creating Tables

- To create and maintain table, you need a database
- While defining columns in a table, you should mention:
  - the **name of the columns**,
  - **datatype** (integer, floating point, string, and so on), and
  - **default value** (if any)
- Let’s take a look at data types before we create table

CREATE TABLE - Syntax

- The CREATE TABLE statement is used to create a new table in a database

Syntax:

```
CREATE TABLE table_name (
       column1 datatype,
       column2 datatype,
       column3 datatype,
   ....);
```

- The column parameters specify the names of the columns of the table
- The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

CREATE TABLE - Example

- We will create a customers table, which will hold the customers information

- The table will have the following columns in it
  - CustomerID
  - FirstName
  - LastName and
  - Country

CREATE TABLE - Example

- We will create a table “customers” in the company database. Select the database ‘company’ using **USE company** command

```
CREATE TABLE customers(
CustomerId int,
first_name varchar(20),
last_name varchar(20),
country varchar(20)
);
```

It is declared as an integer since it contains only integers

first_name, last_name, and country: They contain strings, so they are defined as varchar

CREATE TABLE - Example

- The CustomerID column is of type int and will hold an integer
- 'firstName', 'lastName', and 'country' columns are of type varchar with maximum length of 20 characters
- Check the definition of the table by using: describe *customers* :

Field	Type	Null	Key	Default	Extra
CustomerId	int(11)	YES		NULL	
first_name	varchar(20)	YES		NULL	
last_name	varchar(20)	YES		NULL	
country	varchar(20)	YES		NULL	

Alter Table

The Alter Query

- Sometimes we need to incorporate changes to an already existing tables. For example, renaming a field, changing the data-type, etc

- The ***alter*** command is used to make modification in an existing database/table

- Alter command is generally used with clauses such as change, modify, add, drop

The Alter Query - Change Clause

- To make changes in the column's definition we use the ***Change*** clause
- The change clause allows you to:
  - Change the name of the column
  - Change the column data type
  - Change column constraints

Syntax:

```
ALTER TABLE table_name CHANGE  old_column_name new_column_name data type;
```

The Alter Query - Change Clause

Changing Column Definition

- The *ALTER TABLE* command is used to specify the change in the structure of a table
- This is followed by the *CHANGE* clause that tells the MySql server that we want to change the column name
- The ***CHANGE*** clause is followed by an existing column name that needs to be changed
- And finally, we mention the new definition (new name, new data type, new constraint(optional))

The Alter Query - Change Clause

- Consider a table ***Customer*** with below fields

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
Second_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

- Here, we need to rename ‘Second_name’ as ‘last_name’ with increase in the number of characters

The Alter Query - Change Clause

- Use below *alter* query to change the name of the field ‘Second_name’ to ‘last_name’

```
ALTER TABLE Customer CHANGE Second_name last_name varchar(20);
```

- Use **describe *Customer*** to check if the column name has changed to the desired column name

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
last_name	varchar(20)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

The Alter Query - Modify Clause

- The ***Modify*** clause allows you to:
    - Modify Column Data Type
    - Modify Column Constraints

Syntax:

```
ALTER TABLE table_name MODIFY current_column_name data type constraint;
```

Modify clause CANNOT be used to rename a column

The Alter Query - Change Clause

Modifying Column Definition

- The *ALTER TABLE* command is used to specify the change in the structure of a table
- This is followed by the *MODIFY* clause that tells the MySql server that we want to modify a column
- The ***MODIFY*** clause is followed by an existing column name that needs to be changed
- And finally, we mention the new definition of that column (new data type, new constraint(optional))

The Alter Query - Modify Clause

- Consider a table ***Customer*** with below fields

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
Second_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

- Here, we need to increase the width of ‘First_name’ field from 10 to 25

The Alter Query - Modify Clause

- Use below *alter* query to change the width of 'First_name' to varchar(25) with a NOT NULL constraint

```
ALTER TABLE Customer MODIFY First_name varchar(25) NOT NULL;
```

- Use **describe *Customer*** to check if the column name has changed to the desired column name

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(25)	NO		NULL	
Second_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

Difference between Change and Modify Clause

- If you have already created your MySQL database, and decide after the fact that one of your columns is named incorrectly, you can simply rename it using ***CHANGE***

- ***MODIFY*** does everything ***CHANGE*** can, but without renaming the column

The Alter Query - Add Clause

- The ***Add*** clause allows you to:
    - Add a new column to an existing table
    - Add primary key constraint to an existing column

The Alter Query - Add Clause

Adding a new column to a table

- To add a new column to an existing table, we use the ***ADD COLUMN*** clause with the ***ALTER*** command in the following way

Syntax:

```
ALTER TABLE table_name ADD COLUMN column_name
```

The Alter Query - Add Clause

- Consider the previously created table ***Customer:***

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
last_name	varchar(20)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

- Here, we add a new column ‘Salary’ to this table

The Alter Query - Add Clause

- Use below *alter* query with the add clause:

```
ALTER TABLE Customer ADD COLUMN Salary int;
```

- Use **`describe`** ***`Customer`*** to check if a new column has been added to the table

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
last_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	
Salary	int(11)	YES		NULL	

*By default, the ADD clause adds a column at the end of the table. Use the **AFTER** keyword to add a column at a particular position in a table*

- For example: To add a 'Date_of_Birth' column after 'last_name' column in the table Customer, use the following query :

```
ALTER TABLE Customer ADD Date_of_Birth date AFTER 'last_name';
```

*By default, the ADD clause adds a column at the end of the table. Use the **AFTER** keyword to add a column at a particular position in a table*

- Use **`describe Customer`** to check the table definition

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	YES		NULL	
last_name	varchar(10)	YES		NULL	
Date_of_Birth	date	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	
Salary	int(11)	YES		NULL	

The Alter Query - Drop Clause

Dropping a column from the table

- Suppose you no longer need a column from a table for your analysis
- In this scenario we use the ***ALTER*** command with the ***DROP*** clause to remove a column from the table

Syntax:

```
ALTER TABLE table_name DROP COLUMN column_name
```

The Alter Query - Drop Clause

- Consider a table ***Customer*** with below fields:

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	NO		NULL	
last_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	
Salary	int(11)	YES		NULL	

- Here, we don’t need the column ‘Salary’ from the table

The Alter Query - Drop Clause

- Use below *alter* query to drop the 'Salary' column from the table Customer

```
ALTER TABLE Customer DROP COLUMN Salary;
```

- Use **`describe`** ***`Customer`*** to check if the column has been drop from the table

Field	Type	Null	Key	Default	Extra
customer_id	int(11)	YES		NULL	
First_name	varchar(10)	NO		NULL	
last_name	varchar(10)	YES		NULL	
City	varchar(20)	YES		NULL	
Total_exp	varchar(10)	YES		NULL	

Drop Table

DROP TABLE - Syntax

- The DROP TABLE statement is used to drop an existing table in a database

Syntax:

```
DROP TABLE table_name;
```

Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table!!!!

DROP and TRUNCATE TABLE - Example

- The following SQL statement drops the existing table "company":

```
DROP TABLE customers;
```

- The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself

```
TRUNCATE TABLE customers;
```

The Rename Query

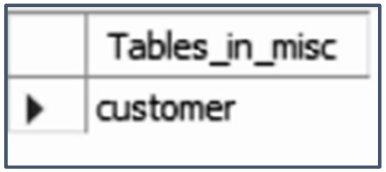
- The rename command is used to change the name of an existing database table to a new name
- Renaming a table does not make it to lose any data is contained within it

Syntax:

```
RENAME TABLE current_table_name TO new_table_name
```

The Rename Query

- Rename the current Customer table to Customer_info



	Tables_in_misc
▶	customer

You can use **`show tables`** command to retrieve the name of all the tables present in a database. ‘misc’ is the name of the database

The Rename Query

- Below command changes the name of the table Customer to Customer_info:

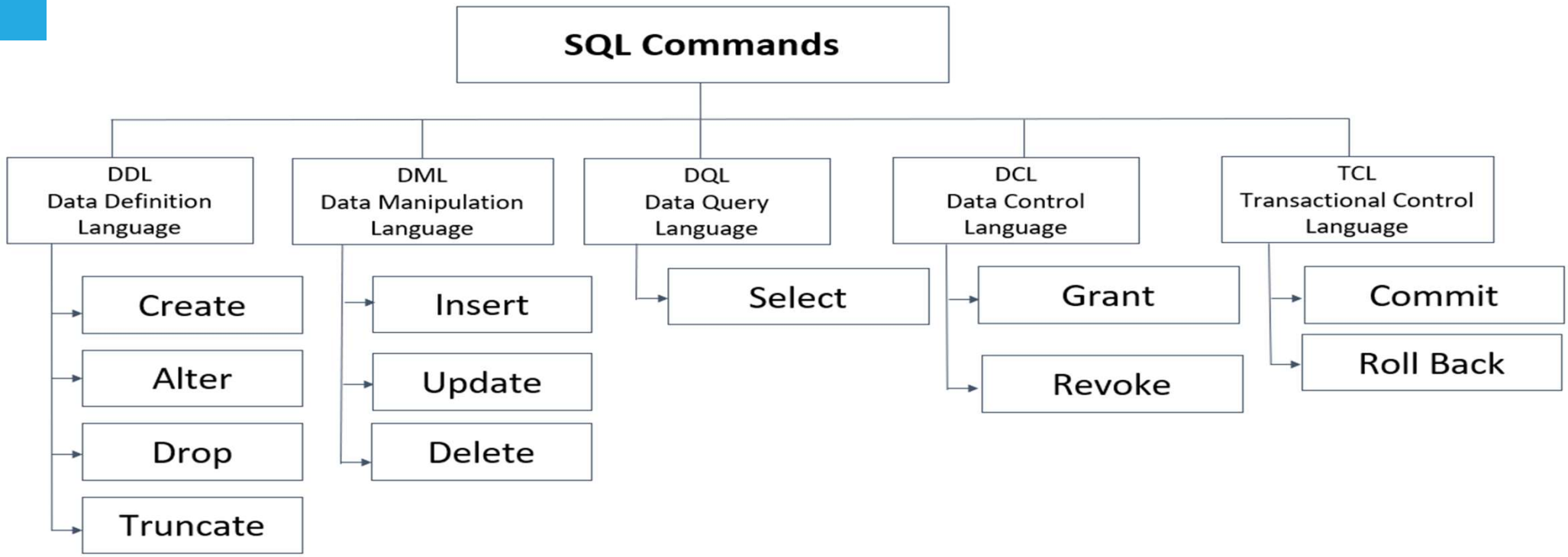
```
RENAME TABLE Customer TO Customer_info
```

- The name of the table Customer is now changed to customer_info:

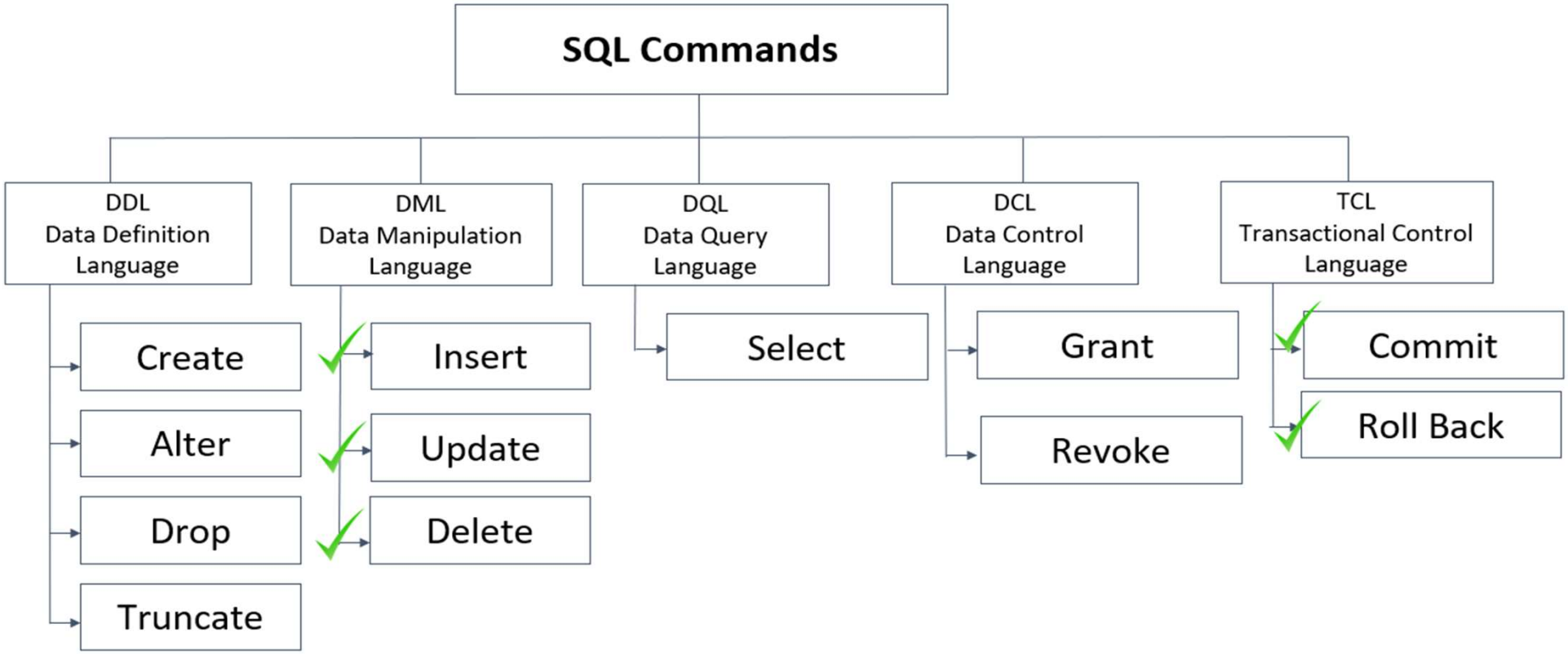
	Tables_in_misc
▶	customer_info

Data Manipulation Language (DML)

Types of SQL Commands



Types of SQL Commands



Data Manipulation Language (DML)

- The typical commands available in DML are:
  - INSERT
  - UPDATE
  - DELETE
  - SELECT

INSERT

SQL INSERT - Syntax

- The INSERT INTO statement helps to insert new records in a table
- We can write the INSERT INTO statement in two ways
- First way specifies both the column names and the values which are required to be inserted

Syntax:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

SQL INSERT - Syntax

- Second way to insert values can be in the following manner where we do not use the column names:

Syntax:

```
INSERT INTO table_name VALUES (value1, value2, value3,...);
```

- Order values needs to be in the same order as the columns in table.

SQL INSERT - Example

- Here we insert values into the customers table created earlier

- The INSERT statement inserts new records in a table

```
INSERT INTO customers (CustomerId,first_name,last_name,country)
VALUES
(1,'Mike', 'Christensen', 'USA'),
(2, 'Andy', 'Hollands', 'Australia'),
(3, 'Ravi', 'Vedantam', 'India');
```

INSERT - Example

- Now, the “customer” table would be:

CustomerId	First_name	Last_name	Country
1	Mike	Christensen	USA
2	Andy	Hollands	Australia
3	Ravi	Vedantam	India

UPDATE

SQL UPDATE - Syntax

- The UPDATE statement modifies the records exist in a table

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

We need to be very careful while updating the records in a table, You can notice the WHERE clause in the UPDATE statement. This clause specifies that which records to be updated. All records in the table will be updates if we omit the WHERE clause.

SQL UPDATE - Example

- Here we are updating the first row of the customers table

- The first_name and the last_name will be updated to John, Kent from the country ‘USA’

```
UPDATE customers
SET first_name = 'John', last_name= 'Kent'
WHERE country = 'USA';
```

UPDATE - Example

- Now, the “customer” table would be:

CustomerId	First_name	Last_name	Country
1	John	Kent	USA
2	Andy	Hollands	Australia
3	Ravi	Vedantam	India

DELETE

SQL DELETE - Syntax

- The DELETE statement deletes existing records in a table

Syntax:

```
DELETE FROM table_name WHERE condition;
```

We should be very careful while deleting the records in a table. You can notice the WHERE clause in the DELETE statement. . This clause specifies that which records to be deleted. All records in the table will be deleted if we omit the WHERE clause.

SQL DELETE - Example

- We are deleting the first row where, first name is John

```
DELETE FROM customers WHERE first_name='John';
```

- Now, the “customer” table would be:

CustomerId	First_name	Last_name	country
2	Andy	Hollands	Australia
3	Ravi	Vedantam	India

Data Transaction Language (TCL)

- The typical commands available in TCL are:
  - COMMIT
  - ROLLBACK

COMMIT

- COMMIT ends the current transaction.
- It permanently saves the transaction effected changes to the physical data files and make others visible to the data changes.
- After the COMMIT has been issued, the current transaction will exit and savepoint will disappear.
- There is no option of database that will "undo" the transaction after committing the data.

ROLLBACK

- In a running transaction which is not yet physically committed to database, the transactional changes can be rollbacked and retain the original status of the data.

- ROLLBACK plays an important roll during interactive operations when user tries to restrain from modifying the data.

Select commands

SELECT Statements – Syntax

- The SELECT statement is used to retrieve data from a table.
- The data returned is stored in a result table, is known as the result-set.

Syntax:

```
SELECT column1, column2, … FROM table_name;
```

SELECT Statement - Syntax

- column1, column2, ... are the column names of the table we want to select data from

- To select all the fields available in the table , follow the given syntax

```
SELECT * FROM table_name;
```

SELECT Statement - Data

Here are selections from "Customers" table in the “company” database:

CustomerID	FirstName	LastName	Country
1	Mike	Christensen	USA
2	Andy	Hollands	Australia
3	Rahul	Vedantam	India
4	Jeevan	Sharma	India

SELECT Column - Example

- Below statement selects the "first_name“ and “country” columns from the "Customers" table:

```
SELECT first_name, Country FROM Customers;
```

Output:

FirstName	Country
Mike	USA
Andy	Australia
Rahul	India
Jeevan	India

Selecting all Columns - Example

- Below statement selects the "first_name"and “country” columns from the "Customers" table:

```
SELECT * FROM Customers;
```

Output:

CustomerID	FirstName	LastName	Country
1	Mike	Christensen	USA
2	Andy	Hollands	Australia
3	Rahul	Vedantam	India
4	Jeevan	Sharma	India

WHERE Clause - Syntax

- WHERE clause is used to filter records
- WHERE clause is also used to return only those records that fulfill a specified condition

Syntax:

```
SELECT column1, column2, … FROM table_name WHERE condition;
```

Apart from SELECT statement, WHERE Clause is also used in UPDATE, DELETE statement, etc.!

WHERE Clause - Syntax

- Below statement selects all the customers from the country "India", in the "Customers" table:

```
SELECT * FROM Customers WHERE Country='India';
```

Output:

CustomerID	FirstName	LastName	Country
3	Rahul	Vedantam	India
4	Jeevan	Sharma	India

LIKE/NOT LIKE Commands

LIKE/NOT LIKE Clause - Syntax

- The LIKE clause is used to do a pattern match
- It extracts only those records that fulfill a given condition

Output:

```
SELECT column1, column2, … FROM table_name WHERE condition LiKE value;
```

Missing Data

Checking Missing Data

- The SQL NULL represents missing values.
- A NULL value in a table means a blank value in a field.
- IS NULL or IS NOT NULL operators are used to check for a NULL value

Checking Missing Data

Consider the following Employee table having the records as shown below

ID	NAME	AGE	ADDRESS	SALARY
1	Kellie	32	California	2000
2	Pete	25	Texas	1500
3	Popy	23	Boston	2000
4	Sam	25	Florida	
5	John	27	Hawaii	

IS NOT NULL Operator

Syntax:

```
SELECT  ID, NAME, AGE, ADDRESS, SALARY FROM Employee
WHERE SALARY IS NOT NULL;
```

Output:

ID	NAME	AGE	ADDRESS	SALARY
1	Kellie	32	California	2000
2	Pete	25	Texas	1500
3	Popy	23	Boston	2000

IS NULL Operator

Syntax:

```
SELECT  ID, NAME, AGE, ADDRESS, SALARY FROM Employee
WHERE SALARY IS NULL;
```

Output:

ID	NAME	AGE	ADDRESS	SALARY
4	Sam	25	Florida	
5	John	27	Hawaii	

Logical Operators

- The following logical operators can be used in WHERE clause

where Condition1 and condition2

or

Operator	Description
✓ AND	Both Conditions must be satisfied
✓ OR	Any one condition must be satisfied
✓ NOT	Negation of the given condition

Relational Operators

- The following relational operators can be used in WHERE clause

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

!=

ORDER BY Clause

- The Order by clause helps us to sort the data
- The data can be sorted in any order (ascending or descending).

default desc.

Syntax :

```
SELECT column1, column2, … FROM table_name
Order by column1 ( asc/desc);
```

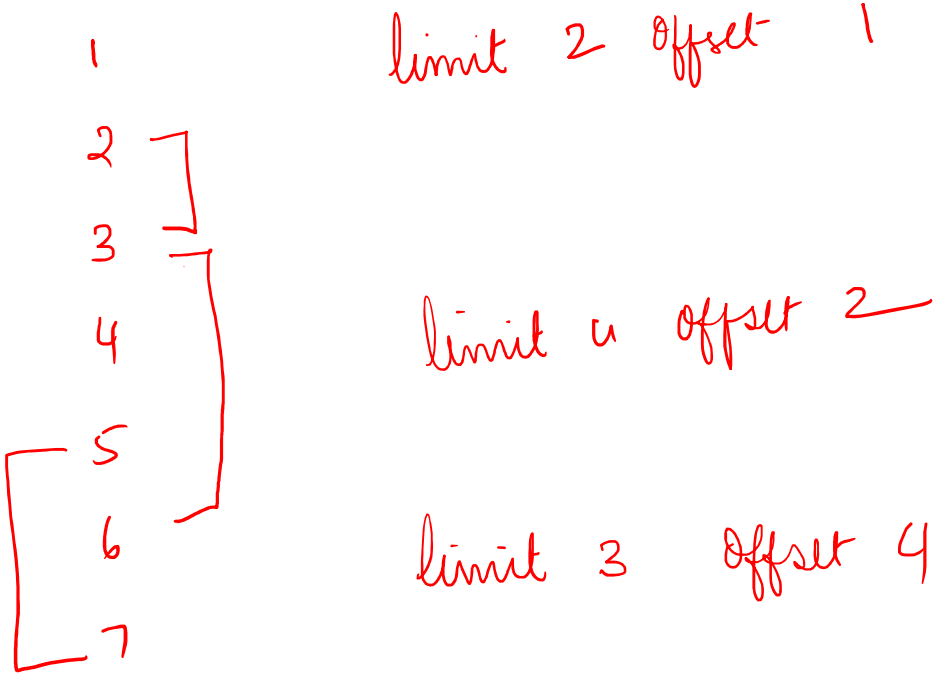
LIMIT/OFFSET Clause

- The Limit by clause is used to limit and print the rows
- Offset mentioned if rows need to be fetched in between

Syntax:

SELECT *column1, column2, …* **FROM** *table_name*
Limit 5 – *fetches first 5 rows*

LIMIT/OFFSET Clause



Arithmetic Operation Clause

- We can perform arithmetic operations on any numerical columns

Syntax:

```
SELECT column1, column2*0.1, … FROM table_name
```

Set Operations

Set Operations

- The Set Operations are used to fetch information from more than one table .
- The Set operations are

  UNION

  UNION ALL

Syntax:

```
SELECT column1, column2, … FROM table_name1
UNION/UNION ALL
SELECT column1, column2, … FROM table_name 1
```

Set Operations

table 1

id	name
1	John
2	Alex
3	Ram

table 2

id	name
1	John
2	Steven.

```
select id, name from table 1
union all / UNION
select id, name from table 2
```

union all

→	1	John
	2	Alex
	3	Ram
→	1	John
	2	Steven

UNION

1	John
2	Alex
3	Ram
2	Steven

Thank You