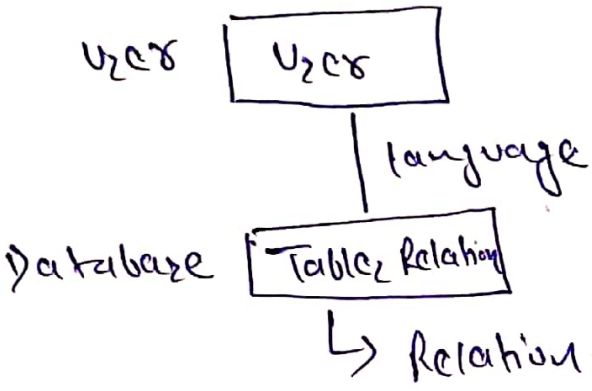
Structure Query Language (SQL)



1970 – EF Code [Father of Relational Algebra DBMS]

↳ Relation DB

→ SQL is domain-specific language.
→ SQL is declarative language
→ DDL, DML, DCL, TCL
→ Keys and Constraints
→ Operators: (like, between, in, Not in, conditional)
→ clauses. (Distinct, order by, group by, from having)
→ aggregate functions.
→ Join and Nested Queries] *** [important]
→ PL SQL (triggers function, Cursor, procedures)

"SQL Commands"

Data Definition language (DDL)

- create
- Alter
- Drop
- truncate
- Rename

- Remove the data.
- change the size/table.
- Remove particular structure

↓

Table

→ here we create the table

Data manipulation language (DML)

- Select
- Insert
- update
- Delete.

↓

we give the data in the table here.

Data Control language (DCL)

- Grant
- Revoke

↓

To control the data of the table

- To give the permission to change
- To take back the permission to change.

Transaction Control language (TCL)

- Commit
- Roll back
- Save point

↓

for transaction of data/money.

Constraints

- Primary key
- foreign key
- check
- unique
- Default
- Not NULL

↓

we use Constrains in the various Control language

=> Create table <'table-name> Command

```
create table <table-name>
(
    colum1_name  datatype,
    colum2name   datatype,
    colum2name   datatype
    );
desc table-name;
```

→ Create table emp

```
(  id int;
   name varchar2, (20),
    salary  number (10);
    );
desc emp;
```

=> Alter Command

→ Add column/s
→ Remove column/s
→ modify datatype
→ modify datatype
→ Add Contraints
→ Remove Contraints.
→ Rename column/table

→ Alter table student Add (addrss varchar(30));

Eg → alter table employee add address varchar(10);
" " " drop column address;
" " " modify id varchar(30);
" " " rename column id to roll-no;

Alter

→ DDL

→ Add/remove / change the datatype

→ change the coloumn name table name

→ change the size of contrains

It change the structure in a table

Update

→ DML

→ It change the data in a table.

Eg: Update Emp
Set salary = Salary * 2;

→ we can apply condition.

Update Emp
Set Salary = Salary * 2
where ID 2;

Delete

DML

→ Delete from tablename
Delete from student.

→ here data will be Deleted.

→ In this we delete row one by one

Delete from student where id = 1;

→ Roll back is possible

Drop

DDL

→ It will delete everything. of the table

→ This access is mainly avaible admin panel only.

Drop student table;

Truncate

DDL

→ It will also delete the row of the table.

→ It will delete all the row of table at once

Truncate student;

→ Roll back is not possible

⇒ Constraints in SQL :

↳ To give / provide the condition.
↳ provide the condition on attributes / colooms.

(I) Unique → [Data should be unique not repeated]

(II) Not NULL → [Neccessary to fill this coloum / mendatory]

(III)* Primary key → [unique + Not NULL]

(IV) check → [to fix the domain of the value like (age>18, addrss : Delhi, chandigan]

(V)* foreign key → [Reference integrity]

(VI) Default → [to provide by default value]

SQL

- **SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language.**
- **SQL is a language to operate databases; it includes database creation, deletion, fetching rows, modifying rows, etc. SQL is an ANSI (American National Standards Institute) standard language, but there are many different versions of the SQL language.**

What is SQL?

- SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.
- SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.
- Also, they are using different dialects, such as –
- MS SQL Server using T-SQL,
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc.

Why SQL?

- SQL is widely popular because it offers the following advantages –
- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

A Brief History of SQL

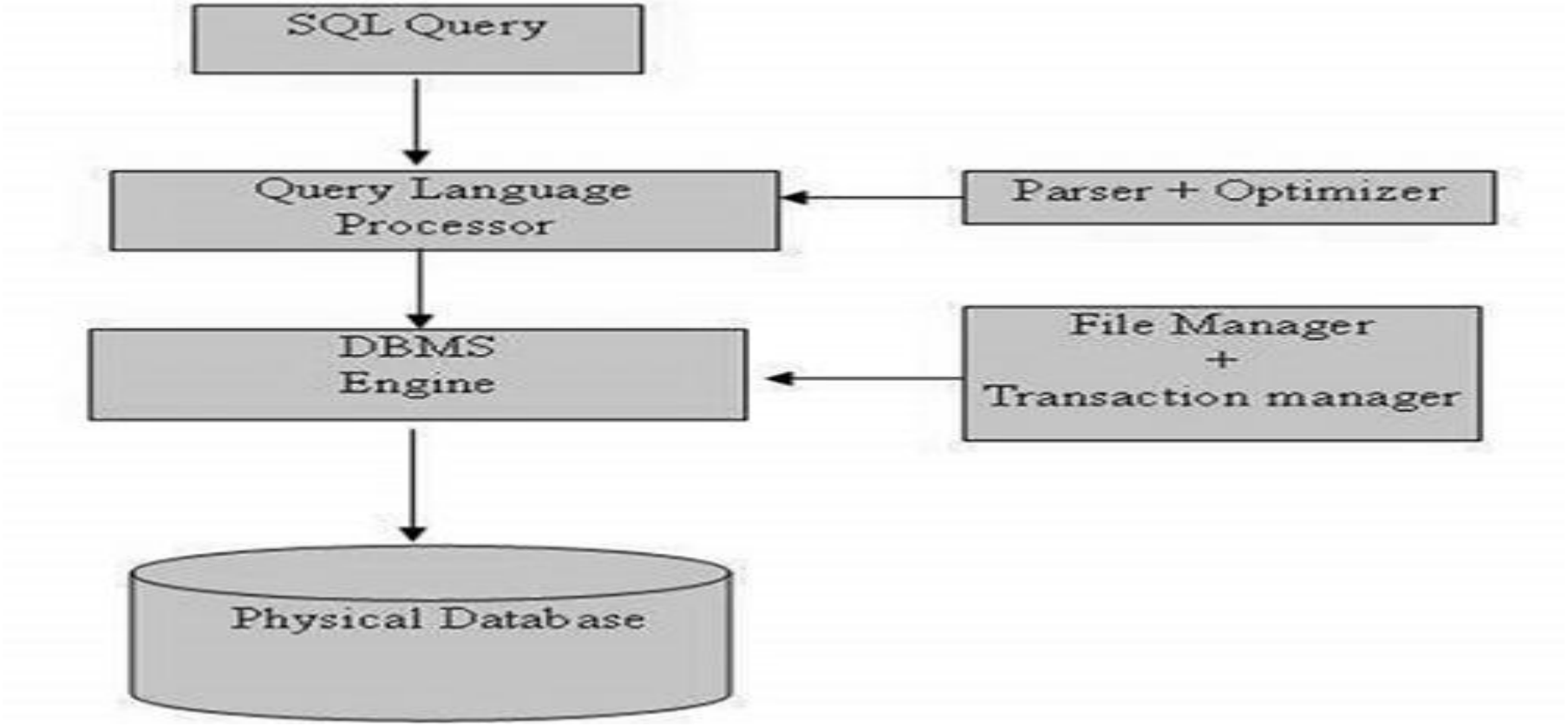
- 1970 – Dr. Edgar F. "Ted" Codd of IBM is known as the father of relational databases. He described a relational model for databases.
- 1974 – Structured Query Language appeared.
- 1978 – IBM worked to develop Codd's ideas and released a product named System/R.
- 1986 – IBM developed the first prototype of relational database and standardized by ANSI. The first relational database was released by Relational Software which later came to be known as Oracle.

SQL Process

- When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.
- There are various components included in this process.
- These components are −
- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.
- A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

SQL Process

- **Following is a simple diagram showing the SQL Architecture**



SQL Commands

- The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into the following groups based on their nature
- DDL - Data Definition Language
- DML - Data Manipulation Language
- DCL - Data Control Language

DDL - Data Definition Language

- **CREATE**

Creates a new table, a view of a table, or other object in the database.

- **ALTER**

Modifies an existing database object, such as a table.

- **DROP**

Deletes an entire table, a view of a table or other objects in the database.

DML - Data Manipulation Language

- **SELECT**

Retrieves certain records from one or more tables

- **INSERT**

Creates a record.

- **UPDATE**

Modifies records.

- **DELETE**

Deletes records.

DCL - Data Control Language

- **GRANT**

Gives a privilege to user.

- **REVOKE**
- **Takes back privileges granted from user.**

What is RDBMS?

- **RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.**
- **A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.**

What is a table?

- The data in an RDBMS is stored in database objects which are called as tables. This table is basically a collection of related data entries and it consists of numerous columns and rows.
- Remember, a table is the most common and simplest form of data storage in a relational database

Table name

Attribute names

Tables in SQL

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Tuples or rows

What is a NULL value?

- **A NULL value in a table is a value in a field that appears to be blank, which means a field with a NULL value is a field with no value.**

- **It is very important to understand that a NULL value is different than a zero value or a field that contains spaces. A field with a NULL value is the one that has been left blank during a record creation.**

SQL Constraints

- **Constraints are the rules enforced on data columns on a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.**

- **Constraints can either be column level or table level. Column level constraints are applied only to one column whereas, table level constraints are applied to the entire table.**

SQL Constraints

- Following are some of the most commonly used constraints available in SQL −
- NOT NULL Constraint − Ensures that a column cannot have a NULL value.
- DEFAULT Constraint − Provides a default value for a column when none is specified.
- UNIQUE Constraint − Ensures that all the values in a column are different.
- PRIMARY Key − Uniquely identifies each row/record in a database table.
- FOREIGN Key − Uniquely identifies a row/record in any another database table.
- CHECK Constraint − The CHECK constraint ensures that all values in a column satisfy certain conditions.

SQL - NOT NULL Constraint

- By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such a constraint on this column specifying that NULL is now not allowed for that column.
- A NULL is not the same as no data, rather, it represents unknown data.

Example FOR NOT NULL

- For example, the following SQL query creates a new table called PERSON
- ```
  CREATE TABLE Persons (
      ID int NOT NULL,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255) NOT NULL,
      Age int
  );
  ```
  ```

SQL DEFAULT Constraint

- The DEFAULT constraint is used to provide a default value for a column.
- The default value will be added to all new records IF no other value is specified.
-
```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

SQL UNIQUE Constraint

- **The UNIQUE constraint ensures that all values in a column are different.**
- **Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.**
- **A PRIMARY KEY constraint automatically has a UNIQUE constraint.**
- **However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.**

SQL UNIQUE Constraint on CREATE TABLE

- CREATE TABLE Persons (
  ID int NOT NULL UNIQUE,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int
  );

SQL PRIMARY KEY Constraint

- **The PRIMARY KEY constraint uniquely identifies each record in a database table.**
- **Primary keys must contain UNIQUE values, and cannot contain NULL values.**
- **A table can have only one primary key, which may consist of single or multiple fields.**

SQL PRIMARY KEY on CREATE TABLE

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

SQL FOREIGN KEY Constraint

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
- The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

SQL FOREIGN KEY Constraint

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

SQL CHECK Constraint

- **The CHECK constraint is used to limit the value range that can be placed in a column.**
- **If you define a CHECK constraint on a single column it allows only certain values for this column.**
- **If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.**

SQL CHECK on CREATE TABLE

- **CREATE TABLE Persons (**
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int,
  CHECK (Age>=18)
  );

Data Integrity

- The following categories of data integrity exist with each RDBMS −
- Entity Integrity − There are no duplicate rows in a table.
- Domain Integrity − Enforces valid entries for a given column by restricting the type, the format, or the range of values.
- Referential integrity − Rows cannot be deleted, which are used by other records.
- User-Defined Integrity − Enforces some specific business rules that do not fall into entity, domain or referential integrity.

DDL - Data Definition Language

- **SQL CREATE TABLE Statement**
- **CREATE TABLE Persons (**
  PersonID int,
  LastName varchar(255),
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255)
  );

DDL - Data Definition Language

- **SQL ALTER TABLE Statement**
- **The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.**
- **The ALTER TABLE statement is also used to add and drop various constraints on an existing table.**
- **ALTER TABLE *table_name***
  MODIFY COLUMN *column_name datatype*;

SQL DROP TABLE Statement

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

DROP TABLE *table_name*;

DML - Data Manipulation Language

- **SELECT**

Retrieves certain records from one or more tables

- **INSERT**

Creates a record.

- **UPDATE**

Modifies records.

- **DELETE**

Deletes records.

SELECT COMMAND

- **The SELECT statement is used to select data from a database.**
- **The data returned is stored in a result table, called the result-set.**

SELECT Syntax

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

SELECT COMMAND

ASCRIPT | SQL | PHP | MORE ▾ | REFERENCES ▾ | EXAMPL

1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-9

SELECT CustomerName, City FROM Customers;

SQL> SELECT * FROM CUSTOMERS;

SQL Query

Basic form: (plus many many more bells and whistles)

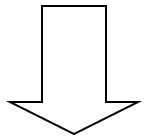
```
SELECT  <attributes>
FROM    <one or more relations>
WHERE   <conditions>
```

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   *
FROM     Product
WHERE    category='Gadgets'
```



PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

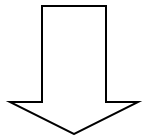
"selection"

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

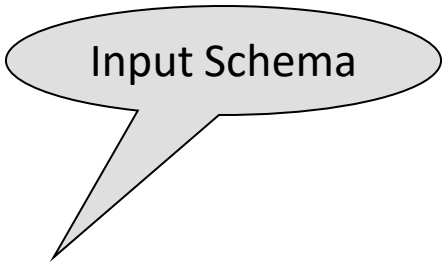
```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```



"selection" and "projection"

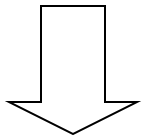
PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

Notation

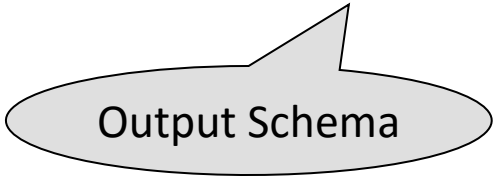


Product(PName, Price, Category, Manfacturer)

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```



Answer(PName, Price, Manfacturer)



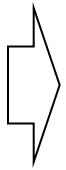
Details

- **Case insensitive:**
  - **Same: SELECT Select select**
  - **Same: Product product**
  - **Different: ‘Seattle’ ‘seattle’**

- **Constants:**
  - **‘abc’ - yes**
  - **“abc” - no**

Eliminating Duplicates

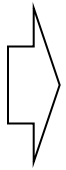
```
SELECT   DISTINCT category
FROM     Product
```



Category
Gadgets
Photography
Household

Compare to:

```
SELECT   category
FROM     Product
```



Category
Gadgets
Gadgets
Photography
Household

Ordering the Results

```
SELECT   pname, price, manufacturer
FROM     Product
WHERE    category='gizmo' AND price > 50
ORDER BY  price, pname
```

Ties are broken by the second attribute on the ORDER BY list, etc.

Ordering is ascending, unless you specify the DESC keyword.

SQL - INSERT Query

- **The INSERT INTO statement is used to insert new records in a table.**
- **INSERT INTO Syntax**
- **It is possible to write the INSERT INTO statement in two ways.**
- **The first way specifies both the column names and the values to be inserted:**
- **INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)**
  VALUES (*value1*, *value2*, *value3*, ...); \or
- **INSERT INTO *table_name***
  VALUES (*value1*, *value2*, *value3*, ...);

CustomerID	CustomerName	ContactName	Address	City	Posta
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-01

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

SQL SELECT DISTINCT Statement

- **The SELECT DISTINCT statement is used to return only distinct (different) values.**
- **Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.**
- **The SELECT DISTINCT statement is used to return only distinct (different) values.**
- **SELECT DISTINCT Syntax**
- **SELECT DISTINCT *column1*, *column2, ...***
  FROM *table_name*;

SQL SELECT DISTINCT Statement

CustomerID	CustomerName	ContactName	Address	City	Posta
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-01

```
SELECT Country FROM Customers;
```

```
SELECT DISTINCT Country FROM Customers;
```

SQL AND, OR and NOT Operators

- The WHERE clause can be combined with AND, OR, and NOT operators.
- The AND and OR operators are used to filter records based on more than one condition:
- The AND operator displays a record if all the conditions separated by AND is TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.
- The NOT operator displays a record if the condition(s) is NOT TRUE.

SQL AND, OR and NOT Operators

CustomerID	CustomerName	ContactName	Address	City	Posta
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-01

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin'
```

```
SELECT * FROM Customers
WHERE City='Berlin' OR City='München';
```

SQL ORDER BY Keyword

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

CustomerID	CustomerName	ContactName	Address	City	Posta
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-01

```
SELECT * FROM Customers
ORDER BY Customerid
```

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

SQL NULL Values

- **A field with a NULL value is a field with no value.**
- **If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value**
- **IS NULL Syntax**
- **SELECT *column_names***
  FROM *table_name*
  WHERE *column_name* IS NULL;

D	LastName	FirstName	Address	City
	Doe	John	542 W. 27th Street	New York
	Bloggs	Joe		London
	Roe	Jane		New York
	Smith	John	110 Bishopsgate	London

```
SELECT LastName, FirstName, Address FROM Persons
WHERE Address IS  NULL;
```

LastName	FirstName	Address
Bloggs	Joe	
Roe	Jane	

SQL UPDATE Statement

- **The UPDATE statement is used to modify the existing records in a table.**

UPDATE Syntax

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

UPDATE Table

- **UPDATE Customers**
  SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
  WHERE CustomerID = 1;

CustomerID	CustomerName	ContactName	Address	City	Posta
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-01

The SQL DELETE Statement

- **The DELETE statement is used to delete existing records in a table.**

- **DELETE Syntax**
- **DELETE FROM *table_name***
  WHERE *condition*;

SQL DELETE Example

CustomerID	CustomerName	ContactName	Address	City	Posta
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-01

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

```
DELETE FROM table_name;
```

```
DELETE * FROM table_name;
```

SQL MIN() and MAX() Functions

- **The MIN() function returns the smallest value of the selected column.**
- **The MAX() function returns the largest value of the selected column.**
- **MIN() Syntax**
- **SELECT MIN(*column_name*)**
  FROM *table_name*
  WHERE *condition*;
- **MAX() Syntax**
- **SELECT MAX(*column_name*)**
  FROM *table_name*
  WHERE *condition*;

SQL MIN() and MAX() Functions

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.3

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

```
SELECT MAX(Price) AS LargestPrice
FROM Products;
```

SQL COUNT(), AVG() and SUM()

- **The COUNT() function returns the number of rows that matches a specified criteria.**
- **The AVG() function returns the average value of a numeric column.**
- **The SUM() function returns the total sum of a numeric column.**
- **COUNT() Syntax**
- **SELECT COUNT(*column_name*)**
  FROM *table_name*
  WHERE *condition*;
- **AVG() Syntax**
- **SELECT AVG(*column_name*)**
  FROM *table_name*
  WHERE *condition*;
- **SUM() Syntax**
- **SELECT SUM(*column_name*)**
  FROM *table_name*
  WHERE *condition*;

SQL COUNT(), AVG() and SUM()

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.3

```
SELECT COUNT(ProductID)
FROM Products;
```

```
SELECT AVG(Price)
FROM Products;
```

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

SQL LIKE Operator

- **The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.**
- **There are two wildcards used in conjunction with the LIKE operator:**
- **% - The percent sign represents zero, one, or multiple characters**
- **_ - The underscore represents a single character**
- **LIKE Syntax**
- **SELECT *column1, column2, ...***
  FROM *table_name*
  WHERE *columnN* LIKE *pattern*;

SQL LIKE Operator

ASCRIPT | SQL | PHP | MORE | REFERENCES | EXAMPLE

WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

CustomerID	CustomerName	ContactName	Address	City	Po
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	W/
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%a';
```

The SQL IN Operator

- **The IN operator allows you to specify multiple values in a WHERE clause.**
- **The IN operator is a shorthand for multiple OR conditions.**
- **IN Syntax**
- **SELECT *column_name(s)***
  FROM *table_name*
  WHERE *column_name* IN (*value1*, *value2*, ...);

Operator

CustomerID	CustomerName	ContactName	Address	City	Po
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	W/
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

```
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

The SQL BETWEEN Operator

- **The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.**
- **The BETWEEN operator is inclusive: begin and end values are included.**
- **BETWEEN Syntax**
- **SELECT *column_name(s)***
  FROM *table_name*
  WHERE *column_name* BETWEEN *value1* AND *value2;*

The SQL BETWEEN Operator

CustomerID	CustomerName	ContactName	Address	City	Po
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	W/
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

SQL Aliases

- **SQL Aliases**
- **SQL aliases are used to give a table, or a column in a table, a temporary name.**
- **Aliases are often used to make column names more readable.**
- **An alias only exists for the duration of the query.**
- **Alias Column Syntax**
- **SELECT *column_name* AS *alias_name***
  FROM *table_name;*

SQL Aliases

CustomerID	CustomerName	ContactName	Address	City	Po
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	W/
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-

```
SELECT CustomerID as ID, CustomerName AS Customer
FROM Customers;
```

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]
FROM Customers;
```

SQL GROUP BY Statement

- **The SQL GROUP BY Statement**
- **The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.**
- **GROUP BY Syntax**
- **SELECT *column_name(s)***
  FROM *table_name*
  WHERE *condition*
  GROUP BY *column_name(s)*
  ORDER BY *column_name(s);*

SQL GROUP BY Statement

CustomerID	CustomerName	ContactName	Address	City	Po
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	W/
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

SQL HAVING Clause

- **The SQL HAVING Clause**
- **The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.**

HAVING Syntax

- **SELECT *column_name(s)***
  FROM *table_name*
  WHERE *condition*
  GROUP BY *column_name(s)*
  HAVING *condition*
  ORDER BY *column_name(s);*

SQL HAVING Clause

- **SELECT COUNT(CustomerID), Country**
  FROM Customers
  GROUP BY Country
  HAVING COUNT >5
- **The following SQL statement lists the number of customers in each country, sorted high to low (Only include countries with more than 5 customers):(CustomerID) > 5;**

SQL CREATE VIEW Statement

- In SQL, a view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- CREATE VIEW Syntax
- CREATE VIEW view_name AS
  SELECT column1, column2, ...
  FROM table_name
  WHERE condition;

SQL CREATE VIEW Examples

- **CREATE VIEW [Current Product List] AS**
  SELECT ProductID, ProductName
  FROM Products
  WHERE Discontinued = No;

DATA CONTROL LANGUAGE

- DCL Commands are used to restrict the operations on a database object for a particular user.DBA GRANTS and REVOKE the user privileges using grant and revoke commands.
- Privileges represents the rights to a particular user to access ,manipulate,destroy various objects
- Note-A user can be granted all the privileges or some specific privileges

Creating Data Base users

- **Note-Before a privilege can be granted to user ,it is mandatory to create a user first**

How to Create Data Base user

Syntax- Create user username

Identified by password;

Example-Create user abc

Identified by hello;

Output – User created

Note-user abc does not have any privileges at this point,so he cannot do anything with the database right now

- **Privilages can be either system or object privilages**

Object Privilages-

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
ALTER	.Allows the user to change the table defination
DELETE	Allows the user to delete the record
Index	Allows the user to create index
With Grant option	Allows the user to grant privileges to some other users

Oracle SQL – Joins

1. The purpose of a join is to combine the data across tables.
2. A join is actually performed by the where clause which combines the specified rows of tables.
3. If a join involves in more than two tables then Oracle joins first two tables based on the joins condition and then compares the result with the next table and so on.

TYPES

1 Equi join
2 Non-equi join
3 Self join
4 Natural join
5 Cross join
6 Outer join

- Left outer
- Right outer
- Full outer

7 Inner join
8 Using clause
9 On clause

Assume that we have the following tables.
SQL> select * from dept;

DEPTNO	DNAME	LOC
10	INVENTORY	HYBD
20	FINANCE	BGLR
30	HR	MUMBAI

SQL> select * from emp;

EMPNO	ENAME	JOB	MGR	DEPTNO
111	saketh	analyst	444	10
222	sudha	clerk	333	20
333	jagan	manager	111	10
444	madhu	engineer	222	40

1. EQUI JOIN

A join which contains an equal to '=' operator in the joins condition.

Ex:

SQL> select empno,ename,job,dname,loc from emp e,dept d where e.deptno=d.deptno;

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	INVENTORY	HYBD
333	jagan	manager	INVENTORY	HYBD
222	sudha	clerk	FINANCE	BGLR

Using clause

SQL> select empno,ename,job ,dname,loc from emp e join dept d using(deptno);

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	INVENTORY	HYBD
333	jagan	manager	INVENTORY	HYBD
222	sudha	clerk	FINANCE	BGLR

On clause

SQL> select empno,ename,job,dname,loc from emp e join dept d on(e.deptno=d.deptno);

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	INVENTORY	HYBD
333	jagan	manager	INVENTORY	HYBD
222	sudha	clerk	FINANCE	BGLR

2. NON-EQUI JOIN

A join which contains an operator other than equal to '=' in the joins condition.

Ex:

SQL> select empno,ename,job,dname,loc from emp e,dept d where e.deptno > d.deptno;

EMPNO	ENAME	JOB	DNAME	LOC
222	sudha	clerk	INVENTORY	HYBD
444	madhu	engineer	INVENTORY	HYBD
444	madhu	engineer	FINANCE	BGLR
444	madhu	engineer	HR	MUMBAI

3. SELF JOIN

Joining the table itself is called self join.

Ex:

SQL> select e1.empno,e2.ename,e1.job,e2.deptno from emp e1,emp e2 where e1.empno=e2.mgr;

EMPNO	ENAME	JOB	DEPTNO
111	jagan	analyst	10
222	madhu	clerk	40
333	sudha	manager	20
444	saketh	engineer	10

4. NATURAL JOIN

Natural join compares all the common columns.

Ex:

SQL> select empno,ename,job,dname,loc from emp natural join dept;

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	INVENTORY	HYBD
333	jagan	manager	INVENTORY	HYBD
222	sudha	clerk	FINANCE	BGLR

5. CROSS JOIN

This will gives the cross product.

Ex:

SQL> select empno,ename,job,dname,loc from emp cross join dept;

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	INVENTORY	HYBD
222	sudha	clerk	INVENTORY	HYBD
333	jagan	manager	INVENTORY	HYBD
444	madhu	engineer	INVENTORY	HYBD
111	saketh	analyst	FINANCE	BGLR
222	sudha	clerk	FINANCE	BGLR
333	jagan	manager	FINANCE	BGLR
444	madhu	engineer	FINANCE	BGLR
111	saketh	analyst	HR	MUMBAI
222	sudha	clerk	HR	MUMBAI
333	jagan	manager	HR	MUMBAI
444	madhu	engineer	HR	MUMBAI

6. OUTER JOIN

Outer join gives the non-matching records along with matching records.

LEFT OUTER JOIN

This will display the all matching records and the records which are in left hand side table those that are not in right hand side table.

Ex:

SQL> select empno,ename,job,dname,loc from emp e left outer join dept d on(e.deptno=d.deptno);

Or

SQL> select empno,ename,job,dname,loc from emp e,dept d where e.deptno=d.deptno(+);

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	INVENTORY	HYBD
333	jagan	manager	INVENTORY	HYBD
222	sudha	clerk	FINANCE	BGLR
444	madhu	engineer		

RIGHT OUTER JOIN

This will display the all matching records and the records which are in right hand side table those that are not in left hand side table.

Ex:

SQL> select empno,ename,job,dname,loc from emp e right outer join dept d on(e.deptno=d.deptno);

Or

SQL> select empno,ename,job,dname,loc from emp e,dept d where e.deptno(+) = d.deptno;

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	INVENTORY	HYBD
333	jagan	manager	INVENTORY	HYBD
222	sudha	clerk	FINANCE	BGLR
			HR	MUMBAI

FULL OUTER JOIN

This will display the all matching records and the non-matching records from both tables.

Ex:

SQL> select empno,ename,job,dname,loc from emp e full outer join dept d on(e.deptno=d.deptno);

EMPNO	ENAME	JOB	DNAME	LOC
333	jagan	manager	INVENTORY	HYBD
111	saketh	analyst	INVENTORY	HYBD
222	sudha	clerk	FINANCE	BGLR
444	madhu	engineer		
			HR	MUMBAI

7. INNER JOIN

This will display all the records that have matched.

Ex:

SQL> select empno,ename,job,dname,loc from emp inner join dept using(deptno);

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	INVENTORY	HYBD
333	jagan	manager	INVENTORY	HYBD
222	sudha	clerkx`	FINANCE	BGLR