DBMS LAB MANUAL

4TH SEMESTER IT

DBMS

Database

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

For example: The college Database organizes the data about the admin, staff, students and faculty etc.

Using the database, you can easily retrieve, insert, and delete the information.

Database Management System

- Database management system is a software which is used to manage the database. For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.
- DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

DBMS allows users the following tasks:

- **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
- **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
- **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
- **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

Characteristics of DBMS

- It uses a digital repository established on a server to store and manage the information.

- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

Advantages of DBMS

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

RDBMS

RDBMS

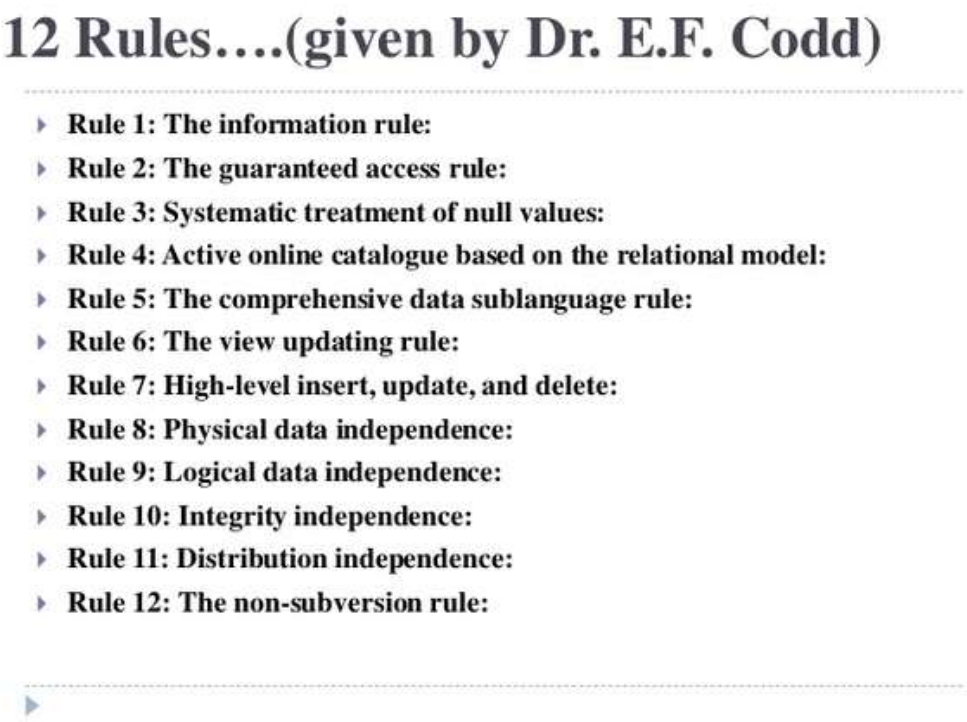
RDBMS stands for **R**elational **D**atabase **M**anagement **S**ystem. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

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.

Relational Database Characteristics

- Data in the relational database must be represented in tables, with values in columns within rows.
- Data within a column must be accessible by specifying the table name, the column name, and the value of the primary key of the row.
- The DBMS must support missing and inapplicable information in a systematic way, distinct from regular values and independent of data type.
- The DBMS must support an active on-line catalogue.
- The DBMS must support at least one language that can be used independently and from within programs, and supports data definition operations, data manipulation, constraints, and transaction management.
- Views must be updatable by the system.
- The DBMS must support insert, update, and delete operations on sets.
- The DBMS must support logical data independence.
- The DBMS must support physical data independence.
- Integrity constraints must be stored within the catalogue, separate from the application.
- The DBMS must support distribution independence. The existing application should run when the existing data is redistributed or when the DBMS is redistributed.
- If the DBMS provides a low level interface (row at a time), that interface cannot bypass the integrity constraints.

CODD'S RULE



Difference between DBMS and RDBMS

Although DBMS and RDBMS both are used to store information in physical database but there are some remarkable differences between them.

The main differences between DBMS and RDBMS are given below:

No.	DBMS	RDBMS
1)	DBMS applications store **data as file**.	RDBMS applications store **data in a tabular form**.
2)	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3)	**Normalization is not** present in DBMS.	**Normalization is** present in RDBMS.

4)	DBMS does **not apply any security** with regards to data manipulation.	RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
5)	DBMS uses file system to store data, so there will be **no relation between the tables**.	in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well.
6)	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7)	DBMS **does not support distributed database**.	RDBMS **supports distributed database**.
8)	DBMS is meant to be for small organization and **deal with small data**. it supports **single user**.	RDBMS is designed to **handle large amount of data**. it supports **multiple users**.
9)	Examples of DBMS are file systems, **xml** etc.	Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc.

After observing the differences between DBMS and RDBMS, you can say that RDBMS is an extension of DBMS. There are many software products in the market today who are compatible for both DBMS and RDBMS. Means today a RDBMS application is DBMS application and vice-versa.

DBA

Database administration is more of an operational or technical level **function** responsible for physical database design, security enforcement, and database performance. Tasks

include maintaining the data dictionary, monitoring performance, and enforcing organizational standards and security.

RESPONSIBILITY OF DBA

- Installation, configuration and upgrading of Database server software and related products.
- Evaluate Database features and Database related products.
- Establish and maintain sound backup and recovery policies and procedures.
- Take care of the Database design and implementation.
- Implement and maintain database security (create and maintain users and roles, assign privileges).
- Database tuning and performance monitoring.
- Application tuning and performance monitoring.
- Setup and maintain documentation and standards.
- Plan growth and changes (capacity planning).
- Work as part of a team and provide 24x7 support when required.
- Do general technical troubleshooting and give cons.
- Database recovery.

ORACLE

An Oracle **database** is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management.

STRUCTURE QUERY LANGUAGE

SQL

SQL stands for structured query language. SQL is the language used by RDBMS for database interactions. It enables the user to define access and manage the relational database.

All task related to relational data management creating table, quering database for information, modifying data in the database, deleting, granting access to users and show on can be easily performed using SQL.

SQL data type

Char - It is used to store character string of fixed length. Maximum size is 255 characters.

Varchar/varchar2 - It is used to store variable length alphanumeric data. Maximum size 8000 characters.

Date - It is used to represent date and time. The standard format is DD / MMM / YY.

Number - It is used to store numbers of any magnitude may be stored upto 38 digit of precission. Maximum size is 9.9 * 10 to the power 124.

Long - It is used to store variable length character string upto 2GB. Long data can be used to store array on binary data in ASCII format.

Raw/long raw - It is used to store binary data such as picture or image. Raw data type can have maximum 255 byte and maximum size of long raw is upto 2GB.

SQL commands

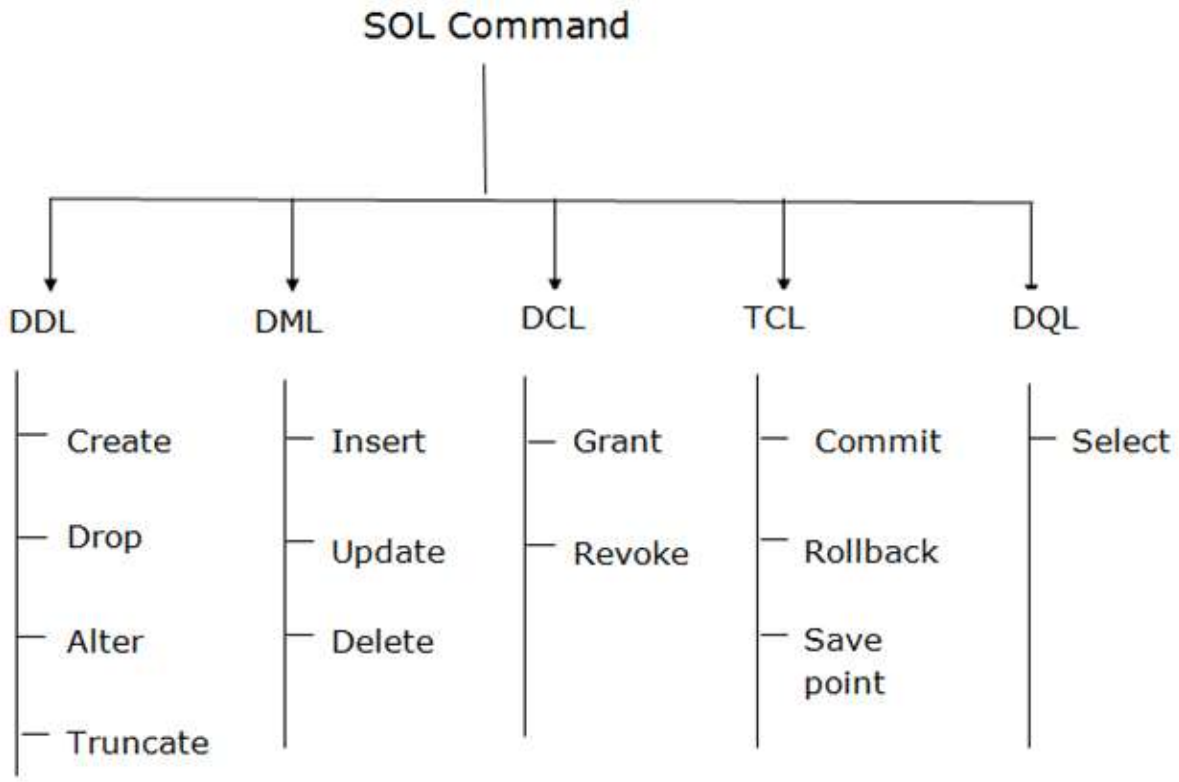
SQL command can be classified into –

DDL (data definition language)
DML (data manipulation language)
DQL (data query language)

DCL (data control language)

DAS (data administrator statement)

TCS (transaction control statement)



DDL (data definition language) -Database objects can be created, altered or deleted using DDL. The command used alter and drop. The major data definition statements are create table, create index, alter table, drop table, drop view, drop index.

DML (data manipulation language) -It allows user to insert modify and delete the data in the database. SQL provides 3 data manipulation statements - insert, update, delete.

DQL (data query language) -SQL has only one data query statement that is select it is one of the most commonly used SQL statement. It enables the user to query one or more tables to get the desired information.

DCL (data control language) -Data control language commands controls the user access to the database object. Does DCL is mainly related to the security issues that is determined who has access to the database object and what operation they can perform on them. It is responsible for preventing on 28 access to the data. DBA has the authority to give and take

privileges to a specific user. The DCL commands are grant and revoke.

DAS (data administrator statement) -It allows the user to perform audits and analysis on operation within the database which commands are also used to analyse the performance of the system. 2 Data communication commands are start audit and stop audit.

TCS (transaction control statement) -All the changes are made by the DML statements are managed by transaction control statements. Some of these commands are commit, rollback, save point, set transaction.

HOW TO OPEN ORACLE DATABASE

- click on start click
- on Oracle database 10g Express
- click on go to database homepage
- then database login
    - username system
    - password Tiger
- then click SQL
- then click SQL command

1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

CREATE TABLE COMMAND

It is used to create a new table in the database.

Syntax:

CREATE TABLE table name

(
column name data type (size),
column data type (size),
.
.
.
.) ;

Example:

CREATE TABLE student
(
roll number varchar (15),
name varchar (30),
branch varchar (5),
semester varchar (3)
);

DROP COMMAND:

It is used to delete both the structure and record stored in the table.

Syntax

DROP TABLE tablename ;

Example

DROP TABLE student;

ALTER COMMAND:

It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

SYNTAX –

```
ALTER TABLE TABLE NAME
ADD
(
    NEW COLUMN_NAME DATA_TYPE (SIZE),
    NEW COLUMN_NAME DATA_TYPE (SIZE),
    .
    );
```

EXAMPLE –

```
    ALTER TABLE MP7
    ADD
    (
    EMP TEL_No NUMBER (10),
    EMP FAX_No NUMBER (15),

);
```

To modify existing column in the table:

SYNTAX –

```
    ALTER TABLE TABLENAME
    MODIFY
    (
    COLUMN_NAME NEW_DATA_TYPE (SIZE)
    );
```

EXAMPLE –

```
    ALTER TABLE EMP7
    MODIFY
    (
    EMP FAX_No VARCHAR (25)
    );
```

To drop a column

SYNTAX –

ALTER TABLE TABLE_NAME

DROP COLUMN COLUMN_NAME;

Example –

ALTER TABLE EMP 7

DROP COLUMN EMPNM;

Change the name of the column

SYNTAX –

ALTER TABLE TABLE_NAME

RENAME COLUMN TO NEW_COLUMN_NAME

EXAMPLE –

ALTER TABLE EMP 7

RENAME COLUMN_NAME EMP_NM TO EMPLOYEE_NAME

TRUNCATE COMMAND:

It is used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table_name;

Example:

TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language(DML)

- DML 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

INSERT COMMAND:

The INSERT statement is a SQL query. It is used to insert data into the row of a table.

SYNTAX:

```
INSERT INTO TABLE_NAME
(col1, col2, col3,.... col N)
VALUES (value1, value2, value3, .... valueN);
```

Or

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

EXAMPLE:

```
INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");
```

UPDATE COMMAND:

This command is used to update or modify the value of a column in the table.

SYNTAX:

```
UPDATE table_name
    SET [column_name1= value1,...column_nameN = valueN]
    [WHERE CONDITION]
```

EXAMPLE:

```
UPDATE students
SET User_Name = 'Sonoo'
WHERE Student_Id = '3'
```

DELETE COMMAND:

It is used to remove one or more row from a table.

SYNTAX:

```
DELETE FROM table_name [WHERE condition];
```

EXAMPLE:

```
DELETE FROM javatpoint
WHERE Author="Sonoo";
```

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

GRANT COMMAND:

It is used to give user access privileges to a database.

EXAMPLE

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

REVOKE COMMAND:

It is used to take back permissions from the user.

EXAMPLE

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

4. Transaction Control Language

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
- ROLLBACK
- SAVEPOINT

COMMIT COMMAND:

Commit command is used to save all the transactions to the database.

SYNTAX:

COMMIT;

EXAMPLE:

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

ROLLBACK COMMAND:

Rollback command is used to undo transactions that have not already been saved to the database.

SYNTAX:

ROLLBACK;

EXAMPLE:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
ROLLBACK;
```

SAVEPOINT COMMAND:

It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

```
SAVEPOINT SAVEPOINT_NAME;
```

5. Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

SELECT COMMAND:

This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

SYNTAX:

SELECT * FROM TABLE NAME

Example –

SELECT * FROM STUDENT

RETRIEVAL OF SPECIFIC COLUMN FROM A TABLE:

SYNTAX –

SELECT column name column name
FROM table name;

EXAMPLE –

SELECT ROLL_NUMBER, NAME FROM STUDENT;

ELIMINATION OF DUPLICATES FROM THE TABLE

SYNTAX –

SELECT DISTINCT COLUMN NAME, COLUMN NAME
FROM TABLE NAME

EXAMPLE –

SELECT DISTINCT STUDENT, NAME, ROLL NUMBER
FROM STUDENT;

SORTING OF DATA IN A TABLE

SYNTAX –

SELECT COLUMN NAME, COLUMN NAME
FROM TABLE NAME
ORDER BY COLUMN NAME;

EXAMPLE –

SELECT ROLLNO, NAME, BRANCH, SEM
FROM STUDENT ORDER BY ROLLNO;

SELECT A DATA SET FROM TABLE DATA

SYNTAX –

SELECT COLUMN NAME, COLUMN NAME
FROM TABLE NAME
WHERE SEARCH CONDITION;

EXAMPLE –

SELECT ROLLNO, NAME
FROM STUDENT WHERE ROLLNO=F18011024009;

Change the name of the table

SYNTAX –

RENAME TABLE TABLE_NAME TO NEW_TABLE_NAME

EXAMPLE –

RENAME TABLE EMP7 TO EMPLOYEE7

Range Searching

SELECT EMP_ID, EMP_NM, DEPT,JOB,SAL
FROM EMP7
WHERE SAL BETWEEN 10000 AND 20000

Pattern Matching

- The use of the **LIKE** predicate

  SELECT EMPNM FROM EMP7
  WHERE EMPNM LIKE A%;

The IN and NOTIN predicates

IN:

SELECT EMPNM , DEPT , JOB , SAL
FROM EMP7
WHERE EMPNM IN('BILL GATES','AMIT BADANA');

NOTIN

SELECT EMPNM , DEPT , JOB , SAL
FROM EMP7
WHERE EMPNM NOTIN('BILL GATES','AMIT BADANA');

Grouping Data From Tables In Sql

```
SELECT EMPNM , DEPT , JOB , SAL
FROM EMP7
GROUP BY DEPT;
```

Having Clause

A condition can be imposed on the group by clause, using “HAVING”.

```
SELECT EMPNM , DEPT , JOB , SAL
FROM EMP7
GROUP BY DEPT
HAVING DEPT = 'IT' OR 'CSE';
```

DATA CONSTRAINTS

SQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints 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 table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

EXAMPLE

```
CREATE TABLE persons (
        Pid VARCHAR (10) PRIMARY KEY,
        name VARCHAR(30),
        birth_date DATE,
        phone VARCHAR(15)
);
```

UNIQUE KEY Constraint

The UNIQUE constraint restricts one or more columns to contain unique values within a table.

Although both a UNIQUE constraint and a PRIMARY KEY constraint enforce uniqueness, use a UNIQUE constraint instead of a PRIMARY KEY constraint when you want to enforce the uniqueness of a column, or combination of columns, that is not the primary key.

The following SQL statement creates a table named *persons* and specifies the *phone* column as unique. That means this field does not allow duplicate values.

EXAMPLE

```
CREATE TABLE persons (
    Pid  VARCHAR2(6) CONSTRAINT U_KEY UNIQUE,
    Name VARCHAR(30) NOT NULL,
    birth_date DATE,
    phone VARCHAR(15)
);
```

NOT NULL Constraint

The NOT NULL constraint specifies that the column does not accept NULL values.

This means if NOT NULL constraint is applied on a column then you cannot insert a new row in the table without adding a non-NULL value for that column.

The following SQL statement creates a table named *persons* with four columns, out of which three columns, *id*, *name* and *phone* do not accept NULL values.

```
CREATE TABLE persons (
   Pid VARCHAR(10) NOT NULL,
    name VARCHAR(30) NOT NULL,
    birth_date DATE,
    phone VARCHAR(15) NOT NULL
);
```

DEFAULT Constraint

The DEFAULT constraint specifies the default value for the columns.

A column default is some value that will be inserted in the column by the database engine when an `INSERT` statement doesn't explicitly assign a particular value.

The following SQL statement creates a default for the *country* column.

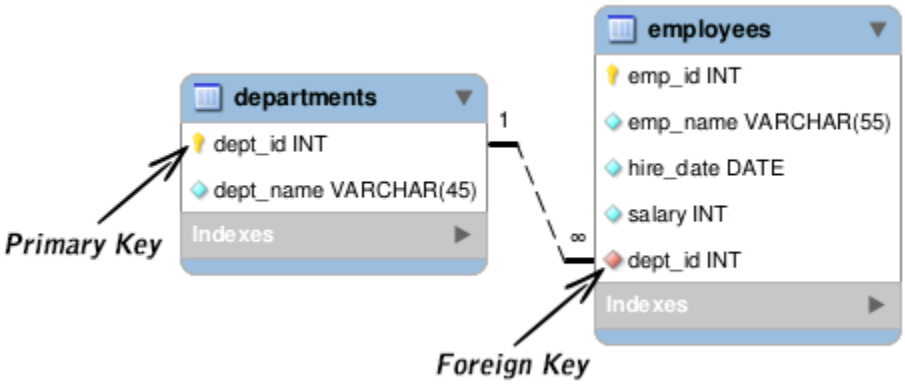
EXAMPLE

```
CREATE TABLE persons (
    Pid VARCHAR (10) PRIMARY KEY,
    name VARCHAR(30),
    birth_date DATE,
    phone VARCHAR(15),
    country VARCHAR(30) DEFAULT 'Australia'
);
```

FOREIGN KEY Constraint

A foreign key (FK) is a column or combination of columns that is used to establish and enforce a relationship between the data in two tables.

Here's a sample diagram showing the relationship between the **employees** and **departments** table. If you look at it carefully, you will notice that the **dept_id** column of the **employees** table matches the primary key column of the **departments** table. Therefore, the **dept_id** column of the **employees** table is the foreign key to the **departments** table.



The following statement establishes a foreign key on the *dept_id* column of the *employees* table that references the *dept_id* column of the *departments* table.

```
CREATE TABLE employees (
    emp_id VARCHAR(10) PRIMARY KEY,
```

```
    emp_name VARCHAR(55) NOT NULL,
    hire_date DATE NOT NULL,
    salary NUMBER,
    dept_id NUMBER,
    FOREIGN KEY (dept_id) REFERENCES departments (dept_id)
);
```

CHECK Constraint

The `CHECK` constraint is used to restrict the values that can be placed in a column.

For example, the range of values for a salary column can be limited by creating a `CHECK` constraint that allows values only from 3,000 to 10,000. This prevents salaries from being entered beyond the regular salary range. Here's an example:

```
CREATE TABLE employees (
    emp_id VARCHAR(10) PRIMARY KEY,
    emp_name VARCHAR(55) NOT NULL,
    hire_date DATE NOT NULL,
    salary NUMBER(5) CHECK (salary >= 3000 AND salary <= 10000),
    dept_id NUMBER(3)
);
```

FUNCTIONS

Oracle Built in Functions

There are two types of functions in Oracle.

1) Single Row Functions:

Single row or Scalar functions return a value for every row that is processed in a query.

2) Group Functions:

These functions group the rows of data based on the values returned by the query. This is discussed in SQL GROUP Functions. The group functions are used to calculate aggregate values like total or average, which return just one total or one average value after processing a group of rows.

There are four types of single row functions. They are:

1) Numeric Functions:

These are functions that accept numeric input and return numeric values.

2) Character or Text Functions:

These are functions that accept character input and can return both character and number values.

3) Date Functions:

These are functions that take values that are of datatype DATE as input and return values of datatype DATE, except for the MONTHS_BETWEEN function, which returns a number.

4) Conversion Functions:

These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.

You can combine more than one function together in an expression. This is known as nesting of functions.

1) Numeric Functions:

Numeric functions are used to perform operations on numbers. They accept numeric values as input and return numeric values as output. Few of the Numeric functions are:

FUNCTION NAME	PURPOSE	EXAMPLES	RETURN VALUE

ABS (x)	Absolute value of the number '*x*'	ABS (1) ABS (-1)	1 -1
CEIL (x)	Integer value that is Greater than or equal to the number '*x*'	CEIL (2.83) CEIL (2.49) CEIL (-1.6)	3 3 -1
FLOOR (x)	Integer value that is Less than or equal to the number '*x*'	FLOOR (2.83) FLOOR (2.49) FLOOR (-1.6)	2 2 -2
TRUNC (x, y)	Truncates value of number '*x*' up to '*y*' decimal places	ROUND (125.456, 1) ROUND (125.456, 0) ROUND (124.456, -1)	125.4 125 120
ROUND (x, y)	Rounded off value of the number '*x*' up to the number '*y*' decimal places	TRUNC (140.234, 2) TRUNC (-54, 1) TRUNC (5.7) TRUNC (142, -1)	140.23 54 5 140

2) Character or Text Functions:

Character or text functions are used to manipulate text strings. They accept strings or characters as input and can return both character and number values as output.

Few of the character or text functions are as given below:

Function Name	**Return Value**	**Examples**	**Return Value**
LOWER (string_value)	All the letters in *'string_value'* is converted to lowercase.	LOWER('Good Morning')	good morning
UPPER (string_value)	All the letters in *'string_value'* is converted to uppercase.	UPPER('Good Morning')	GOOD MORNING
INITCAP (string_value)	All the letters in *'string_value'* is converted to mixed case.	INITCAP('GOOD MORNING')	Good Morning
LTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the left of *'string_value'*.	LTRIM ('Good Morning', 'Good)	Morning
RTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the right of *'string_value'* .	RTRIM ('Good Morning', ' Morning')	Good
TRIM (trim_text FROM string_value)	All occurrences of *'trim_text'* from the left and right of *'string_value'* , *'trim_text'* can	TRIM ('o' FROM 'Good Morning')	Gd Mrning

	also be only one character long .		
SUBSTR (string_value, m, n)	Returns *'n'* number of characters from *'string_value'* starting from the '*m'* position.	SUBSTR ('Good Morning', 6, 7)	Morning
LENGTH (string_value)	Number of characters in *'string_value'* in returned.	LENGTH ('Good Morning')	12
LPAD (string_value, n, pad_value)	Returns '*string_value'* left-padded with *'pad_value'* . The length of the whole string will be of *'n'* characters.	LPAD ('Good', 6, '*')	**Good
RPAD (string_value, n, pad_value)	Returns '*string_value'* right-padded with *'pad_value'* . The length of the whole string will be of *'n'* characters.	RPAD ('Good', 6, '*')	Good**

3) Date Functions:

These are functions that take values that are of datatype DATE as input and return values of datatypes DATE, except for the MONTHS_BETWEEN function, which returns a number as output.

Few date functions are as given below.

Function Name	**Return Value**
ADD_MONTHS (date, n)	Returns a date value after adding *'n'* months to the date *'x'*.
MONTHS_BETWEEN (x1, x2)	Returns the number of months between dates x1 and x2.
ROUND (x, date_format)	Returns the date *'x'* rounded off to the nearest century, year, month, date, hour, minute, or second as specified by the *'date_format'*.
TRUNC (x, date_format)	Returns the date *'x'* lesser than or equal to the nearest century, year, month, date, hour, minute, or second as specified by the 'date_format'.
NEXT_DAY (x, week_day)	Returns the next date of the *'week_day'* on or after the date *'x'* occurs.
LAST_DAY (x)	It is used to determine the number of days remaining in a month from the date *'x'* specified.

SYSDATE	Returns the systems current date and time.
NEW_TIME (x, zone1, zone2)	Returns the date and time in zone2 if date 'x' represents the time in zone1.

The below table provides the examples for the above functions

Function Name	**Examples**	**Return Value**
ADD_MONTHS ( )	ADD_MONTHS ('16-Sep-81', 3)	16-Dec-81
MONTHS_BETWEEN( )	MONTHS_BETWEEN ('16-Sep-81', '16-Dec-81')	3
NEXT_DAY( )	NEXT_DAY ('01-Jun-08', 'Wednesday')	04-JUN-08
LAST_DAY( )	LAST_DAY ('01-Jun-08')	30-Jun-08
NEW_TIME( )	NEW_TIME ('01-Jun-08', 'IST', 'EST')	31-May-08

4) Conversion Functions:

These are functions that help us to convert a value in one form to another form. For Ex: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE.

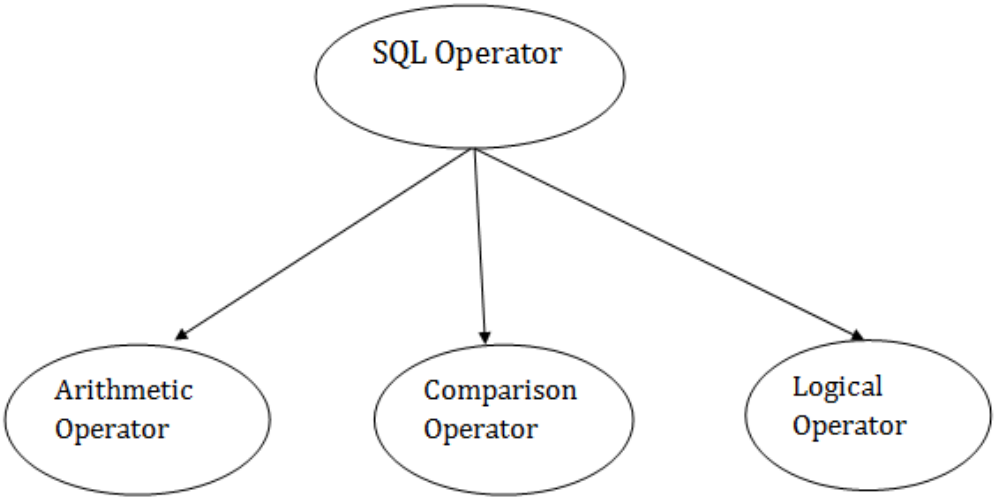
Few of the conversion functions available in oracle are:

Function Name	**Return Value**	**Examples**	**Return Value**
TO_CHAR (x [,y])	Converts Numeric and Date values to a character string value. It cannot be used for calculations since it is a string value.	TO_CHAR (3000, '$9999') TO_CHAR (SYSDATE, 'Day, Month YYYY')	$3000 Monday, June 2008
TO_DATE (x [, date_format])	Converts a valid Numeric and Character values to a Date value. Date is formatted to the format specified by *'date_format'*.	TO_DATE ('01-Jun-08')	01-Jun-08
NVL (x, y)	If *'x'* is NULL, replace it with *'y'*. *'x'* and *'y'* must be of the same datatype.	NVL (null, 1)	1
DECODE (a, b, c, d, e, default_value)	Checks the value of *'a'*, if *a = b*, then returns *'c'*. If *a = d*, then returns *'e'*. Else, returns *default_value*.		

SQL OPERATOR

OPERATOR

There are various types of SQL operator:



SQL Arithmetic Operators

Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.

Operator	Description	Example
+	It adds the value of both operands.	a+b will give 30
-	It is used to subtract the right-hand operand from the left-hand operand.	a-b will give 10
*	It is used to multiply the value of both operands.	a*b will give 200
/	It is used to divide the left-hand operand by the right-hand operand.	a/b will give 2
%	It is used to divide the left-hand operand by the right-hand operand and returns reminder.	a%b will give 0

SQL Comparison Operators:

Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.

Operator	Description	Example
=	It checks if two operands values are equal or not, if the values are queal then condition becomes true.	(a=b) is not true
!=	It checks if two operands values are equal or not, if values are not equal, then condition becomes true.	(a!=b) is true
<>	It checks if two operands values are equal or not, if values are not equal then condition becomes true.	(a<>b) is true
>	It checks if the left operand value is greater than right operand value, if yes then condition becomes true.	(a>b) is not true
<	It checks if the left operand value is less than right operand value, if yes then condition becomes true.	(a<b) is true
>=	It checks if the left operand value is greater than or equal to the right operand value, if yes then condition becomes true.	(a>=b) is not true
<=	It checks if the left operand value is less than or equal to the right operand value, if yes then condition becomes true.	(a<=b) is true
!<	It checks if the left operand value is not less than the right operand value, if yes then condition becomes true.	(a!=b) is not true
!>	It checks if the left operand value is not greater than the right operand value, if yes then condition becomes true.	(a!>b) is true

SQL Logical Operators

There is the list of logical operator used in SQL:

Operator	Description
ALL	It compares a value to all values in another value set.
AND	It allows the existence of multiple conditions in an SQL statement.
ANY	It compares the values in the list according to the condition.
BETWEEN	It is used to search for values that are within a set of values.
IN	It compares a value to that specified list value.
NOT	It reverses the meaning of any logical operator.
OR	It combines multiple conditions in SQL statements.
EXISTS	It is used to search for the presence of a row in a specified table.
LIKE	It compares a value to similar values using wildcard operator.

JOIN

SQL JOIN

JOIN means to combine something. In case of SQL, JOIN means "to combine two or more tables".

In SQL, JOIN clause is used to combine the records from two or more tables in a database.

Types of SQL JOIN

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. FULL JOIN

Sample Table

EMPLOYEE

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

PROJECT

PROJECT_NO	EMP_ID	DEPARTMENT
101	1	Testing
102	2	Development
103	3	Designing
104	4	Development

INNER JOIN

In SQL, INNER JOIN selects records that have matching values in both tables as long as the condition is satisfied. It returns the combination of all rows from both the tables where the condition satisfies.

SYNTAX

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```

EXAMPLE

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
INNER JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development

Christian	Designing
Kristen	Development

LEFT JOIN

The SQL left join returns all the values from left table and the matching values from the right table. If there is no matching join value, it will return NULL.

SYNTAX

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

EXAMPLE

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
LEFT JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development
Russell	NULL

Marry	NULL

RIGHT JOIN

In SQL, RIGHT JOIN returns all the values from the values from the rows of right table and the matched values from the left table. If there is no matching in both tables, it will return NULL.

SYNTAX

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

EXAMPLE

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
RIGHT JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

OUTPUT

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development

FULL JOIN

In SQL, FULL JOIN is the result of a combination of both left and right outer join. Join tables have all the records from both tables. It puts NULL on the place of matches not found.

SYNTAX

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

EXAMPLE

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
FULL JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development
Russell	NULL
Marry	NULL

INDEX

SQL Index

- Indexes are special lookup tables. It is used to retrieve data from the database very fast.
- An Index is used to speed up select queries and where clauses. But it shows down the data input with insert and update statements. Indexes can be created or dropped without affecting the data.
- An index in a database is just like an index in the back of a book.
- **For example:** When you reference all pages in a book that discusses a certain topic, you first have to refer to the index, which alphabetically lists all the topics and then referred to one or more specific page numbers.

Create Index

It is used to create an index on a table. It allows duplicate value.

SYNTAX

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

EXAMPLE

```
CREATE INDEX idx_name
ON Persons (LastName, FirstName);
```

Create Unique Index

It is used to create a unique index on a table. It does not allow duplicate value.

SYNTAX

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

Example

```
CREATE UNIQUE INDEX websites_idx
ON websites (site_name);
```

Drop Index

It is used to delete an index in a table.

SYNTAX

```
DROP INDEX index_name;
```

EXAMPLE

```
DROP INDEX websites_idx;
```

VIEW

Views in SQL

- Views in SQL are considered as a virtual table. A view also contains rows and columns.
- To create the view, we can select the fields from one or more tables present in the database.
- A view can either have specific rows based on certain condition or all the rows of a table.

Sample table:

Student_Detail

STU_ID	NAME	ADDRESS
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

Student_Marks

STU_ID	NAME	MARKS	AGE
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

Creating view

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

SYNTAX:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

EXAMPLE

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM Student_Details
WHERE STU_ID < 4;
```

Just like table query, we can query the view to view the data.

```
SELECT * FROM DetailsView;
```

Output:

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

Creating View from multiple tables

View from multiple tables can be created by simply include multiple tables in the SELECT statement.

In the given example, a view is created named MarksView from two tables Student_Detail and Student_Marks.

EXAMPLE

CREATE VIEW MarksView AS

SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS
FROM Student_Detail, Student_Mark
WHERE Student_Detail.NAME = Student_Marks.NAME;

To display data of View MarksView:

SELECT * FROM MarksView;

NAME	ADDRESS	MARKS
Stephan	Delhi	97
Kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

Deleting View

A view can be deleted using the Drop View statement.

SYNTAX

DROP VIEW view_name;

Example:

DROP VIEW MarksView;