**SQL history:**

The **SQL** programming language was first developed in the 1970s by IBM researchers Raymond Boyce and Donald Chamberlin. The programming language, known then as SEQUEL, was created following the publishing of Edgar Frank Todd's paper, "A Relational Model of Data for Large Shared Data Banks," in 1970.

SQL is the only one language which use to interact with **RDMS.**

**SQL** port No: **3306**

**Assignment:**

**Codd's Rule for Relational DBMS**

E.F Codd was a Computer Scientist who invented the **Relational model** for Database management. Based on relational model, the **Relational database** was created. Codd proposed 13 rules popularly known as **Codd's 12 rules** to test DBMS's concept against his relational model. Codd's rule actualy define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS). Till now, there is hardly any commercial product that follows all the 13 Codd's rules. Even **Oracle** follows only eight and half(8.5) out of 13. The Codd's 12 rules are as follows.

Rule zero

This rule states that for a system to qualify as an **RDBMS**, it must be able to manage database entirely through the relational capabilities.

**Rule 1: Information rule**

All information(including metadata) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.

**Rule 2: Guaranted Access**

Each unique piece of data(atomic value) should be accesible by : **Table Name + Primary Key(Row) + Attribute(column)**.

**NOTE:** Ability to directly access via POINTER is a violation of this rule.

**Rule 3: Systematic treatment of NULL**

Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Also, Primary key must not be null, ever. Expression on NULL must give null.

**Rule 4: Active Online Catalog**

Database dictionary(catalog) is the structure description of the complete **Database** and it must be stored online. The Catalog must be governed by same rules as rest of the database. The same query language should be used on catalog as used to query database.

**Rule 5: Powerful and Well-Structured Language**

One well structured language must be there to provide all manners of access to the data stored in the database. Example: **SQL**, etc. If the database allows access to the data without the use of this language, then that is a violation.

**Rule 6: View Updation Rule**

All the view that are theoretically updatable should be updatable by the system as well.

**Rule 7: Relational Level Operation**

There must be Insert, Delete, Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

**Rule 8: Physical Data Independence**

The physical storage of data should not matter to the system. If say, some file supporting table is renamed or moved from one disk to another, it should not effect the application.

**Rule 9: Logical Data Independence**

If there is change in the logical structure(table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

**Rule 10: Integrity Independence**

The database should be able to enforce its own integrity rather than using other programs. Key and Check constraints, trigger etc, should be stored in Data Dictionary. This also make **RDBMS** independent of front-end.

**Rule 11: Distribution Independence**

A database should work properly regardless of its distribution across a network. Even if a database is geographically distributed, with data stored in pieces, the end user should get an impression that it is stored at the same place. This lays the foundation of **distributed database**.

**Rule 12: Nonsubversion Rule**

If low level access is allowed to a system it should not be able to subvert or bypass integrity rules to change the data. This can be achieved by some sort of looking or encryption

**DDL:**

1. **CREATE**
2. **ALTER**
3. **DROP**

**DROP DATABASE:** drop database DB\_NAME;

**Drop Column:** alter table table\_name drop column column\_name;

1. **RENAME**

rename table old\_table\_name to new\_table\_name;

1. **TRUNCATE**

truncate table table\_name;

**DML:**

|  |  |  |
| --- | --- | --- |
| Product\_id | Product\_name | Product\_quant |
|  |  |  |
|  |  |  |
|  |  |  |

**Insert Values:**

Insert into table\_name(column\_name2,column\_name2….) values(values1,values2….);

Insert into table\_name (column\_name2,column\_name2….) values(values1,values2….), values(values1,values2….),

**Update Values:**

update table\_name set column\_name=values;

**Delete:**

delete from table-name;

**DQL**[Data Query Language]:

**Select** is the only one comment prsent inside the **DQL**.

Select \* from table\_name;

With the help of DQL we can perform following retrieval operations from the table:

* **Projection:**

Select column\_name from table\_name;

We can retrieve row wise data as well as column wise data.

* **Selection:**
* **Joins:**

**From table\_name;**

**Select \***

\* is a **WILD-CARD** Character

**select** helps us to prepare the result set**.**

**Order of Execuation: from --> select**

**Creating A Table:**

create table employee(eid int(100),ename varchar(100),mobile\_no int(100) unique,d\_o\_b date not null,age int,d\_0\_j date,blood\_group varchar(100),deptno int, salary int,email varchar(100),gender varchar(100),primary key(eid));

**DISTINCT:**

It helps to give us the Unique Value;

Select ditnict column1, column3, column3.. from table\_name;

Select designation from employee;

Select designation,dept\_no from employee[It returns the combination of Uniqueness]

**ALIAS:**

It is a temporary name.

**Table Syntax:** SELECT column\_name as

alias\_name from table\_name;

**CLAUSE:**

It is a filteration logic.

Select \* from table\_name where condition…;

Update product set quantity=10 where p\_id=01;

**\*\*Where condition is only used with Update,Delete and Select.**

**Order Of The Execuation Of Where Clause:**

select \* from table\_name where column\_name=value;

**3**

**2**

**1**

**EXPRESSIONS:**

Operators

Operands

**a + b**

Statement is a combination of operator and operands.

Operands are two types:

1. Columns
2. Literals- i. Numbers

ii. Character

iii. Date

**\*Characters and Date must be passed inside Single Quote ( ‘ ’ )**

**\*\*Date must be on YYYY-MM-DD**

**Write a query to display the salary of the employee that is incremented by 5000.**

select salary+5000 incremented from employee;

**Write a query to display the annual salary of the employee.**

select salary\*12 annual\_salary from employee;

**Write a query to display the perday salary of the employee.**

select salary/30 monthly\_salary from employee;

**Write a query to display the result form the table whose job designation is Softwere Devloper.**

select \* from employee where designation = 'softwere developer';

**Write a query to display all the record of employee excluding who is working in the dept. no 20.**

select \* from employee where deptno !=20;

**Write a query to display the record of the employee whose salary is greater 50000.**

select \* from employee where salary>50000;

**Write a query to display the record of the employee whose age is less than 30.**

select \* from employee where age<30000;

**Write a query to display name,salary,mobile no,email id whose age is greater then 28.**

Select e\_name,sal,email\_id ,mobile\_no from employee where age>=28;

**Write a query to display name,salary,mobile no,email id whose salary is less than equal to 40000.**

Select e\_name,sal,email\_id ,mobile\_no from employee where sal>=40000;

**LOGICAL OPERATOR:**

1. **AND :**

O/P

(A . B)

B

A

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **0/P** |
| **T** | **F** | **F** |
| **T** | **T** | **T** |
| **F** | **T** | **F** |
| **F** | **F** | **F** |

Select \* from table\_name where condition1 and condition 2

1. **OR :**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **0/P** |
| **T** | **F** | **T** |
| **T** | **T** | **T** |
| **F** | **T** | **T** |
| **F** | **F** | **F** |

Select \* from table\_name where condition1 and condition 2

1. **NOT :**

|  |  |
| --- | --- |
| **A** | **O/P** |
| **T** | **F** |
| **F** | **T** |

**IN Operator:**

Select \* from table\_name where column\_name in(values 1,values 2,….values n);

Select \* from employee where dept\_no in(10,20,70,50);

**Not-in Operator:**

Select \* from table\_name where column\_name not in(values 1,values 2,….values n);

**Between Operator:**

**Select \* from table\_name where column\_name between value 1 and value 2 ;**

Higher Value

Lower Value

**IS Operator:**

When we have to check null condition ,we go for IS Operator.

Select \* from table\_name where column\_name is not null.

**Like Operator:**

\_ and % is two special character

\_ contains single value character

% 0 to N spaces

Select \* from table\_name where column\_name like ‘\_%’;

Select \* from employee where name is like ‘z%’;

**Function:**

Function means to perform some specific task.

* Single row function/Scalar Function
* Multi row function/Aggregate Function
* **Single row function/Scalar Function:**

**Input 1**

**Output 3**

**Output 2**

**Output 1**

**Input 3**

**Input 2**

**For n no of inputs for n no. of o/p.**

* **Multi row function/Aggregate function:**

**Input 1**

**Output**

**Input 3**

**Input 2**

**For n no of inputs only one output**

* **AVG()- Returns Average Value.**

Select sum(salary) from employee;

* **COUNT()- It counts the no of rows.**

Select count(\*) from employee;

* **MAX()- Returns maximum value of the selected column .**

Select max(salary) from employee;

* **MIN()- Returns maximum values of the saleceted column.**

Select min(salary) from employee;

* **SUM()- Returns sum of the all the values of the selected columns.**

Select sum(salary) from employee;

**Order By Clause:**

Select \* from table\_name where condition… order by coloum\_name asc/desc;

**\*\*Order of execuation**: From>where>select>order by

**Sub Queries:**

* Query inside a query is known as sub query.
* Inner query will execute first and it’ll be treated as input for the outer query.
* Final outcome will come help of the outer query.

Select \* from table\_name where column\_name operator (select \* from table\_name)

**Inner Query**

**Two Types Of Sub Query:**

* **Single Row Sub Query:**

Whenever we are dealing with relational operator/comparison operator,that means that is a single row sub query.

Select \* from employee where id **=** 01;

Select \* from table\_name where column\_name operator (select \* from table\_name);

select max(salary) from employee where salary<(select max(salary) from employee);

* **Multi Row Sub Query:**

Whenever multirow row fuction will execute ,inner query will give multiple result set and that is called multi row sub query.

Select \* from Table\_Name where column\_name operator (select \* from table);

For ANY & ALL we have to pass multiple value;

**Group By Caluse:**

Group by clause execute row by row.

Form>Group By >count>select

|  |  |  |
| --- | --- | --- |
| Eid | Name | DeptNo |
| 1 | A | 10 |
| 2 | B | 20 |
| 3 | C | 10 |
| 4 | D | 20 |
| 5 | E | 30 |

|  |  |  |
| --- | --- | --- |
| 1 | A | 10 |

**Grouping &**

**count(2)**

|  |  |  |
| --- | --- | --- |
| 3 | C | 10 |

|  |  |  |
| --- | --- | --- |
| 2 | B | 20 |

|  |  |  |
| --- | --- | --- |
| 4 | D | 20 |

**Grouping &**

**count(2)**

**Grouping &**

**count(1)**

|  |  |  |
| --- | --- | --- |
| 5 | E | 30 |

**Having Clause:**

After Groupby if we have any filteration logic then we have to use Have Clause.

Select \* from table\_name where condition groupby column\_name having condition;

Form>Group By >Having Clause>count>select>order by

\*\***Where Clause** is use to filter the records & **Having Clause** is use to filter the group by clause.

**TCL:**

Rollback

Commit

Savepoint

Savepoint only works with rollback.

**JOINS:**

* **Cross Join:**

Record form first table will be marge with all the record of second table.

|  |  |
| --- | --- |
| **A1** | **B1** |
| **a** | **10** |
| **b** | **20** |
| **c** | **30** |

|  |  |
| --- | --- |
| **A2** | **B2** |
| **d** | **40** |
| **e** | **50** |
| **f** | **60** |

|  |  |  |  |
| --- | --- | --- | --- |
| **A1** | **B1** | **A2** | **B2** |
| a | 10 | d | 40 |
| a | 10 | e | 50 |
| a | 10 | f | 60 |
| b | 20 | d | 40 |
| b | 20 | e | 50 |
| b | 20 | f | 60 |
| c | 30 | d | 40 |
| c | 30 | e | 50 |
| c | 30 | f | 60 |

**ASCII Syntax:**

Select \* from table\_name1 cross join table\_name2;

**Oracle Syntax:**

Select \* from table\_name 1,table\_name2

**Inner Join:**

Inner Join is use to opted only matched record form the table.

|  |  |
| --- | --- |
| **A1** | **B1** |
| **a** | **10** |
| **b** | **20** |
| **c** | **30** |

|  |  |
| --- | --- |
| **A2** | **B1** |
| **d** | **20** |
| **e** | **50** |
| **f** | **60** |

**ANSI Syntax:**

Select \* from table \_name 1 Inner Join table\_name 2 on condition

Select \* from employee e inner join department d on e.deptno=d.deptno

**Oracle Syntax:**

Select \* from Table\_name 1,Table\_name2 Where<join.condition>

**Left Outer Join:**

It’s a combination of inner join and unmatched record present in the left record.

**ANSI Syntax:**

Select \* from Table\_name1 leftouterjoin Table\_name2 ob condition.

**Oracle Syntax:**

Select \* from Table1,Table2 where T1.Column\_name=T2.Column\_name

|  |  |
| --- | --- |
| **A1** | **B1** |
| **a** | **10** |
| **b** | **20** |
| **c** | **30** |

|  |  |
| --- | --- |
| **A2** | **B1** |
| **d** | **20** |
| **e** | **50** |
| **f** | **60** |

|  |  |  |  |
| --- | --- | --- | --- |
| **A1** | **B1** | **A2** | **B1** |
| **b** | **20** | **d** | **20** |
| **a** | **10** | **null** | **null** |
| **c** | **30** | **null** | **null** |

**Full Outer Join:**

It opted the unmatched record from both the table.

SELECT \* FROM table1 FULL OUTER JOIN table2 ON table1.column\_name=table2.column\_name;

**Natural Join:**

Natural join is similar to inner join where we don’t specify the condition.

Natural Joins provides inner query output if there are common column present else it returns Cartesian Join output.

SELECT \*FROM table1 NATURAL JOIN table2;

**Self Join:**

Joining of two same table is known as self join.We use this join whenever data to be selected within same table but it should present in two different table.

**\*\*We have to use Alias Name Here.**

**Syntax:**

SELECT column\_name(s) FROM table1 T1, table1 T2 WHERE condition;

Composite Key: When we have more than one primary key,we can call it as foreign key.

Super Key Attribute: The set of all the Key attribute is known as super key attribute.

**Functional Dependency:**

A relation which exist such that an attributes determined uniquely that is known as functional dependency.

**R --> { x, y }**

**x->Determinent**

**y->dependent**

* Total Functional Dependency
* Partial Functional Dependency
* Transitive Dependency

If all the attributes of a relation determined by a key attribute that is known as Total Functional Dependency.

R -->{A,B,C,D}

A->B,C,D

A relation is set to have partial functional dependency

It considered aa Composite Key Attribute

Their exist dependency such that an attribute can be determined by other attribute which is part of composite ki.

R->{A,B,C,D}

{A,B}->{C,D}

{B}->{C}

{A,B}->{D}

A relation is set to have a transitional functional dependency if they exist a relation such that as a attribute is determined by a non key attribute which entire determined by key attribute.

**NORMALISATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Name** | **Provider** | **S-ID** |
| **01** | **Soumen** | **Docomo** | **7501613519** |
| **02** | **Soumen** | **Airtel** | **8436984559** |
| **03** | **Soumen** | **Vodafone** | **8768145367** |

The process of decomposing the table into smaller table in order to remove redundecy and anomaly by identifying dependency is knows normalisation

A state of table without redudency and anomaly is known as Normal Form.

The problem which are occurred during the dml operation that is known as Anamoly.

* **1St Normal Form:**

A table is called in 1st Normal Form if it satisfied following condition,

1. Record Shouldn’t be repeated.
2. Every Cell in the table should be a single value(automic value).

* **2nd NF:**

1. It should follow the 1NF.
2. The table shouldn’t contain partial functional dependency**.**

* **3rd NF:**

1. It should follow the 2NF.
2. It shouldn’t have transitional functional dependency.

**Relation between more than two table is called as mapping.**

**One to One.**

**One to Many.**

**Many to One.**

**Many to Many.**