# Relational Database Management System

**Relational Database:** It is a collection of logically related tables.

Table: Employee

| Eno | Name | Desig |
|---|---|---|
| 34 | Ankur Singh | Mgr |
| 45 | Jatin Dua | Dir |
| 32 | Ravina | Mgr |
| 12 | Harshit | Acc |
| 01 | Raj | Recp |
| 09 | Kirti | Mgr |

Table: Pay

| Desig | Designation | Salary |
|---|---|---|
| Mgr | Manager | 78000 |
| Dir | Director | 90000 |
| Acc | Accountant | 25000 |
| Recp | Receptionist | 10000 |

**Relational Database Management System:** The software required to handle/manipulate these tables/relations is known as Relational Database Management System (RDBMS) – Oracle, Sybase, DB2, MS SQL Server, MYSQL, etc.

**Table/Relation:** Table is the collection of related data entries which means that the table should consists of columns and rows. The horizontal subset of the Table is known as a Row/Tuple. The vertical subset of the Table is known as a Column/an Attribute. A relation in a database has the following characteristics:

1. Every value in a relation is atomic - i.e. it cannot be further divided
2. Names of columns are distinct and order of columns is immaterial
3. The rows in the relation are not ordered

"Since relation is a set, and sets are not ordered hence no ordering defined on tuples of relation"

Table:Employee

| Eno | Name | Desig |
|---|---|---|
| 34 | Ankur Singh | Mgr |
| 45 | Jatin Dua | Dir |
| 32 | Ravina | Mgr |
| 12 | Harshit | Acc |
| 01 | Raj | Recp |
| 09 | Kirti | Mgr |

Table:Pay

| Desig | Salary |
|---|---|
| Mgr | 78000 |
| Dir | 90000 |
| Acc | 25000 |
| Recp | 10000 |

Column/Attribute
Degree: No. of columns

Rows/Tuples
Cardinality: No. of Rows

**Key:** An attribute/group of attributes in a table that identifies a tuple uniquely is known as a Key. A table may have more than one such attribute/group of attribute that identifies a tuple uniquely, all such attribute(s) are known as **Candidate Keys**. Out of the Candidate keys, one is selected as **Primary Key**, and others become **Alternate Keys.**

Table:Stock

| Ino | Item | Qty |
|---|---|---|
| I01 | Pen | 560 |
| I02 | Pencil | 780 |
| I04 | CD | 450 |
| I09 | Floppy | 700 |
| I05 | Eraser | 300 |
| I03 | Duster | 200 |

Candidate Keys

Primary Key Alternate Key

Table:Department

| DCode | Dname |
|---|---|
| D01 | Acc |
| D05 | Recp |
| D02 | HR |
| D03 | Admin |

Candidate Keys

Primary Key Alternate Key

**Relational algebra:** Following set of operations can be carried out on a relation:

1. **Selection (unaryoperator)**: To select a horizontal subset of a relation.

2. **Projection(unaryoperator):** To select vertical subset of a relation

3. **Cartesian Product (binary operator):** It operates on two relations and is denoted by X. For example Cartesian product of two relations R1 and R2 is represented by R = R1 X R2. The degree of R is equal to sum of degrees of R1 and R2. The cardinality of R is product of cardinality of R1 and cardinality of R2.
Example:
Relation: R1

| Roll No | Student Name | Class |
|---|---|---|
| 1 | Akash | XII |
| 4 | Debakar | X |
| 10 | Rishi | XI |

Relation: R2

| Teacher Code | Teacher Name |
|---|---|
| 102 | Ms. Rinee |
| 309 | Mr. Tanmoy |

Resultant : R = R1 X R2

| Col1 | Col2 | Col3 | Col4 | Col5 |
|---|---|---|---|---|
| 1 | Akash | XII | 102 | Ms Rinee. |
| 1 | Akash | XII | 309 | Mr. Tanmoy |
| 4 | Debakar | X | 102 | Ms. Rinee |
| 4 | Debakar | X | 309 | Mr. Tanmoy |
| 10 | Rishi | XI | 102 | Ms. Rinee |
| 10 | Rishi | XI | 309 | Mr. Tanmoy |

4. **Union (binary operator):** It operates on two relations and is indicated by U. For example, R=R1 U R2 represents union operation between two relations R1 and R2. The degree of R is equal to degree of R1. The cardinality of R is sum of cardinality of R1 and cardinality of R2. Following have to be considered for the operation R1 U R2.

   - Degree of R1 = Degree of R2
   - jth attribute of R1 and jth attribute of R2 must have a common domain.

   Example:
   Relation: R1

| Student_ID | Name |
|---|---|
| R490 | Fatima |
| R876 | Faraz |
| R342 | Gauri |

Relation: R2

| Student_Code | Student_Name |
|---|---|
| S123 | Rekha |
| S456 | Tushi |

Resultant Relation : R = R1 U R2

| Column1 | Column2 |
|---|---|
| R490 | Fatima |
| R876 | Faraz |
| R342 | Gauri |
| S123 | Rekha |
| S456 | Tushi |

# SQL - Structured Query Language

## DDL–Data Definition Language

The SQL-DDL contains set of commands, which sets up, changes or removes data structures from the database. These data structures can be tables or other database objects.

## DML – Data Manipulation Language

The SQL-DML includes those commands, which are based on both the relational algebra and the tuple relational calculus. DML is a language that enables users to access or manipulate data. By data manipulation, we mean:

- The retrieval of information stored in the table
- The insertion of new row with information into the table
- The deletion of information from the table (not deleting the column)
- The modification of information stored in the table (not modifying the data type of column)

DCL – Data Control Language
TCL – Transaction Control Language

## Data Types (as supported by SQL in Oracle)

| | |
|---|---|
| CHAR, VARCHAR2 | To store textual data |
| NUMBER | To store numeric data |
| DATE | To store date |

### Creating a new table in the database (DDL)

```
Syntax:
CREATE TABLE <Table Name>
            (<Column Name1> <Data Type>,<Column Name2> <Data Type>,
                  ... <Column Name n> <Data Type>);
Example:
CREATE TABLE student
    (Rno NUMBER(5),Name VARCHAR2(25),Fees NUMBER(8,2), DOB Date);
```

### Inserting a new row at the bottom of the table (DML)

```
Syntax:
INSERT INTO <Table Name> [(<Col1>,<Col2>,... <Col N>)]
            VALUES ((<Col1 Value>,<Col2 Value>,... <Col N Value>);

Examples:
INSERT INTO student VALUES (34,'Manish',7800,'07-Sep-03');

INSERT INTO student VALUES (35,'Dhruv',8800,'29-Feb-00');

INSERT INTO student VALUES (38,'Gagan',9800,'10-Sep-01');
```

### Adding a new column(s) in the table (DDL)

```
Syntax:
ALTER TABLE <Table Name>
            ADD (<Column Name1> <Data Type>,<Column Name2> <Data Type>,
                  ... <Column Name n> <Data Type>);

Examples:
ALTER TABLE  student ADD (ClassSec VARCHAR2(3));

INSERT INTO student (Rno,Name,Fees,ClassSec)
VALUES (23,'Rimal',6700,'12C');
```

## Displaying the content from a table - SELECT

**Syntax:**

```
SELECT */<Col1>,<Col2>, ... <Col N>
FROM <Table Name>
WHERE <Condition>;
```

**Example:**

```
SELECT * FROM student;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 34 | Manish | 7800 | 07-Sep-03 | |
| 35 | Dhruv | 8800 | 29-Feb-00 | |
| 38 | Gagan | 9800 | 10-Sep-01 | |
| 23 | Rimal | 6700 | | 12C |

```
SELECT name FROM student;
```

| NAME |
|---|
| Manish |
| Dhruv |
| Gagan |
| Rimal |

| | |
|---|---|
| Relational Operators | =, <, >, <=, >=, <> |
| Logical Operators | AND, OR, NOT |

```
SELECT * FROM student WHERE Rno>35;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 38 | Gagan | 9800 | 10-Sep-01 | |

```
SELECT * FROM student WHERE Rno>23 AND Rno<38;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 34 | Manish | 7800 | 07-Sep-03 | |
| 35 | Dhruv | 8800 | 29-Feb-00 | |

```
SELECT * FROM student WHERE Rno>35 OR ClassSec='12C';
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 38 | Gagan | 9800 | 10-Sep-01 | |
| 23 | Rimal | 6700 | | 12C |

```
SELECT Name,Fees,12*Fees AFEES FROM student WHERE NOT (Fees=8800);
```

| NAME | FEES | AFEES |
|---|---|---|
| Manish | 7800 | 93600 |
| Gagan | 9800 | 117600 |
| Rimal | 6700 | 80400 |

Using an expression

Using Alias (i.e. an alternative name for a column)

**Fees<>8800;**

Use of IN (used for distinct set) and BETWEEN (used for a range) with all data types

**SELECT * FROM student WHERE Rno IN (23,34,38);**

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 34 | Manish | 7800 | 07-Sep-03 | |
| 38 | Gagan | 9800 | 10-Sep-01 | |
| 23 | Rimal | 6700 | | 12C |

The range is inclusive of 34 and 38

**SELECT * FROM student WHERE Rno BETWEEN 34 AND 38;**

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 34 | Manish | 7800 | 07-Sep-03 | |
| 35 | Dhruv | 8800 | 29-Feb-00 | |
| 38 | Gagan | 9800 | 10-Sep-01 | |

The range is inclusive of both names 'Dhruv' and 'Gagan'

**SELECT * FROM student WHERE Name BETWEEN 'Dhruv' AND 'Gagan';**

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 35 | Dhruv | 8800 | 29-Feb-00 | |
| 38 | Gagan | 9800 | 10-Sep-01 | |

Name Starting with 'D'

**SELECT Name FROM student WHERE Name LIKE 'D%';**

| NAME |
|---|
| Dhruv |

Name Ending with 'D'

**SELECT Name FROM student WHERE Name LIKE '%n';**

| NAME |
|---|
| Gagan |

Name having 'a' anywhere

**SELECT Name FROM student WHERE Name LIKE '%a%';**

| NAME |
|---|
| Manish |
| Gagan |
| Rimal |

## Modifying the existing content of the table (DML)

```
Syntax:
UPDATE <Table Name>
SET <Col1>=<Value1> [,<Col2>=<Value2>,... <Col N>=<Value N>]
[WHERE <Condition>];

Example:
UPDATE student SET ClassSec='12A' WHERE Rno<36;

SELECT Rno, Name, ClassSec FROM student;
```

| RNO | NAME | CLASSSEC |
|---|---|---|
| 34 | Manish | 12A |
| 35 | Dhruv | 12A |
| 38 | Gagan | |
| 23 | Rimal | 12C |

```
UPDATE student SET Name='Suryansh' WHERE Rno=38;
SELECT * FROM student;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 34 | Manish | 7800 | 07-Sep-03 | 12A |
| 35 | Dhruv | 8800 | 29-Feb-00 | 12A |
| 38 | Suryansh | 9800 | 10-Sep-01 | |
| 23 | Rimal | 6700 | | 12C |

```
UPDATE student SET ClassSec='12B' WHERE Rno=38;
SELECT * FROM student;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 34 | Manish | 7800 | 07-Sep-03 | 12A |
| 35 | Dhruv | 8800 | 29-Feb-00 | 12A |
| 38 | Suryansh | 9800 | 10-Sep-01 | 12B |
| 23 | Rimal | 6700 | | 12C |

```
UPDATE student SET DOB='01-Jul-01' WHERE Rno=23;

SELECT * FROM student;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 34 | Manish | 7800 | 07-Sep-03 | 12A |
| 35 | Dhruv | 8800 | 29-Feb-00 | 12A |
| 38 | Suryansh | 9800 | 10-Sep-01 | 12B |
| 23 | Rimal | 6700 | 01-Jul-01 | 12C |

```
SELECT Name,DOB FROM student;
```

| NAME | DOB |
|---|---|
| Manish | 07-Sep-03 |
| Dhruv | 29-Feb-00 |
| Suryansh | 10-Sep-01 |
| Rimal | 01-Jul-01 |

## Arranging the data in ascending or descending order of one/multiple columns (use of ORDER BY clause with SELECT)

**Syntax:**

```
SELECT */<Col1>,<Col2>, ... <Col N>
FROM <Table Name> ORDER BY <Col1> [ASC/DESC],<Col2> [ASC/DESC],... ;
```

**Example:**

```
SELECT * FROM student ORDER BY Rno;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 23 | Rimal | 6700 | 01-Jul-01 | 12C |
| 34 | Manish | 7800 | 07-Sep-03 | 12A |
| 35 | Dhruv | 8800 | 29-Feb-00 | 12A |
| 38 | Suryansh | 9800 | 10-Sep-01 | 12B |

```
SELECT * FROM student ORDER BY Name;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 35 | Dhruv | 8800 | 29-Feb-00 | 12A |
| 34 | Manish | 7800 | 07-Sep-03 | 12A |
| 23 | Rimal | 6700 | 01-Jul-01 | 12C |
| 38 | Suryansh | 9800 | 10-Sep-01 | 12B |

```
SELECT * FROM student ORDER BY Fees DESC;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 38 | Suryansh | 9800 | 10-Sep-01 | 12B |
| 35 | Dhruv | 8800 | 29-Feb-00 | 12A |
| 34 | Manish | 7800 | 07-Sep-03 | 12A |
| 23 | Rimal | 6700 | 01-Jul-01 | 12C |

```
SELECT ClassSec,Name,DOB,Fees
FROM student ORDER BY ClassSec, Name;
```

| CLASSSEC | NAME | DOB | FEES |
|---|---|---|---|
| 12A | Dhruv | 29-Feb-00 | 8800 |
| 12A | Manish | 07-Sep-03 | 7800 |
| 12B | Suryansh | 10-Sep-01 | 9800 |
| 12C | Rimal | 01-Jul-01 | 6700 |

```
SELECT ClassSec,Name,DOB,Fees
FROM student ORDER BY ClassSec, Name DESC;
```

| CLASSSEC | NAME | DOB | FEES |
|---|---|---|---|
| 12A | Manish | 07-Sep-03 | 7800 |
| 12A | Dhruv | 29-Feb-00 | 8800 |
| 12B | Suryansh | 10-Sep-01 | 9800 |
| 12C | Rimal | 01-Jul-01 | 6700 |

```
SELECT ClassSec,Name,DOB,Fees
FROM student ORDER BY ClassSec DESC, Name ;
```

| CLASSSEC | NAME | DOB | FEES |
|---|---|---|---|
| 12C | Rimal | 01-Jul-01 | 6700 |
| 12B | Suryansh | 10-Sep-01 | 9800 |
| 12A | Dhruv | 29-Feb-00 | 8800 |
| 12A | Manish | 07-Sep-03 | 7800 |

## Using Aggregate Functions with SELECT

| | |
|---|---|
| **COUNT()** | **To count the number of rows/values (non-null)** |
| **SUM()** | **To find the sum of values in the column (Numeric Data)** |
| **AVG()** | **To find the average of values in the column (Numeric Data)** |
| **MAX()** | **To find the maximum value in the column** |
| **MIN()** | **To find the minimum value in the column** |

```
SELECT COUNT(*) FROM student;
```

| COUNT(*) |
|---|
| 4 |

```
SELECT COUNT(Rno) FROM student;
```

| COUNT(RNO) |
|---|
| 4 |

```
SELECT SUM(Fees) FROM student;
```

| SUM(FEES) |
|---|
| 33100 |

```
SELECT AVG(Fees) FROM student;
```

| AVG(FEES) |
|---|
| 6620 |

```
SELECT MAX(Fees),MIN(Fees) FROM student;
```

| MAX(FEES) | MIN(FEES) |
|---|---|
| 9800 | 6700 |

```
SELECT DISINCT ClassSec FROM Student;
```

| CLASSSEC |
|---|
| 12A |
| 12B |
| 12C |

```
SELECT COUNT(DISTINCT ClassSec) FROM Student;
```

| COUNT(DISTINCT CLASSSEC) |
|---|
| 3 |

```
SELECT COUNT(ClassSec) FROM Student;
```

| COUNT(CLASSSEC) |
|---|
| 4 |

```
INSERT INTO Student VALUES (12, 'Jatin',6600, '09-Jan-01', '12B');
SELECT * FROM Student;
```

| RNO | NAME | FEES | DOB | CLASSSEC |
|---|---|---|---|---|
| 34 | Manish | 7800 | 07-Sep-03 | 12A |
| 35 | Dhruv | 8800 | 29-Feb-00 | 12A |
| 38 | Suryansh | 9800 | 10-Sep-01 | 12B |
| 23 | Rimal | 6700 | 01-Jul-01 | 12C |
| 12 | Jatin | 6600 | 09-Jan-01 | 12B |

```
SELECT SUM(Fees) FROM Student WHERE ClassSec='12A';
```

| SUM(FEES) |
|---|
| 16600 |

```
SELECT SUM(Fees) FROM Student WHERE ClassSec='12B';
```

| SUM(FEES) |
|---|
| 16400 |

```
SELECT SUM(Fees) FROM Student WHERE ClassSec='12C';
```

| SUM(FEES) |
|---|
| 6700 |

## Grouping data under given Column – (GROUP BY)

```
SELECT ClassSec,SUM(Fees) FROM Student GROUP BY ClassSec;
```

| CLASSSEC | SUM(FEES) |
|---|---|
| 12A | 16600 |
| 12B | 16400 |
| 12C | 6700 |

```
SELECT ClassSec,COUNT(*) FROM Student GROUP BY ClassSec;
```

| CLASSSEC | COUNT(*) |
|---|---|
| 12A | 2 |
| 12B | 2 |
| 12C | 1 |

```
SELECT ClassSec,MAX(Fees),MIN(Fees)
```

```
FROM Student GROUP BY ClassSec;
```

| CLASSSEC | MAX(FEES) | MIN(FEES) |
|---|---|---|
| 12A | 8800 | 7800 |
| 12B | 9800 | 6600 |
| 12C | 6700 | 6700 |

```
SELECT ClassSec,MAX(DOB)
FROM Student GROUP BY ClassSec HAVING COUNT(*)>1;
```

| CLASSSEC | MAX(DOB) |
|---|---|
| 12A | 07-Sep-03 |
| 12B | 10-Sep-01 |

## Order of precedence

```
SELECT     */<Col1>,<Col2>,..., <Col.N>/<Expression>/<Agg.Func.>
FROM      <Table Name>
[WHERE         <Condition>]
[GROUP BY <Grouping Col.>]
[HAVING   <Aggregate Condition>]
[ORDER BY <OrderingCol1>[ASC/DESC],<OrderingCol2> [ASC/DESC]...];
```

## Deleting a row/rows from a table – (DML)

**Syntax:**
```
DELETE FROM <Table Name> [WHERE <Condition> ];
```

**Example:**
```
DELETE FROM Student WHERE Rno=13;
```

**To delete all rows of a table (Does not delete the structure of the table)**

```
DELETE FROM Student;
```

## Deleting a table – (DDL) To delete the data as well as the structure

**Syntax:**
```
DROP TABLE <Table Name>;
```

**Example:**
```
DROP TABLE Student;
```

## Modifying the data type of a column - (DDL)

**Syntax:**
```
ALTER TABLE <Table Name> MODIFY <Col1> <Data Type>;
```

**Example:**
```
ALTER TABLE Student MODIFY Name CHAR(30);
```

## Deleting a column from a table - (DDL)

**Syntax:**
```
ALTER TABLE <Table Name> DROP COLUMN <Column Name>;
```

**Example:**
```
ALTER TABLE Student DROP COLUMN Rno;
```

## Working with more than one table

### Cartesian Product (CROSS Product)

Table:Student

| Rno | Name |
|---|---|
| 1 | Rahat |
| 2 | Jaya |
| 3 | Tarun |

Table:Games

| Gcode | Gname |
|---|---|
| 101 | Football |
| 102 | Table Tennis |

**SELECT Name,Gname FROM Student,Games;**

| Name | Gname |
|---|---|
| **Rahat** | **Football** |
| **Jaya** | **Football** |
| **Tarun** | **Football** |
| **Rahat** | **Table Tennis** |
| **Jaya** | **Table Tennis** |
| **Tarun** | **Table Tennis** |

## Join

Table:ADMISSION

| RNO | NAME |
|---|---|
| 2 | Fardeen |
| 3 | Harish |
| 1 | ANIK |
| 4 | PRIYA |

Table: FEE

| RNO | FEES |
|---|---|
| 3 | 3500 |
| 1 | 2500 |
| 4 | 3000 |

**SELECT A.Rno,Name,Fees**
**FROM Admission A,Fee B WHERE A.Rno=B.Rno;**

| RNO | NAME | FEES |
|---|---|---|
| 3 | Harish | 3500 |
| 1 | ANIK | 2500 |
| 4 | PRIYA | 3000 |

**SELECT A.Rno,Name,Fees**
**FROM Admission A,Fee B WHERE A.Rno=B.Rno ORDER BY 1;**

| RNO | NAME | FEES |
|---|---|---|
| **1** | **ANIK** | **2500** |
| **3** | **Harish** | **3500** |
| **4** | **PRIYA** | **3000** |

**SELECT A.Rno,Name,Fees**
**FROM Admission A,Fees D WHERE A.Rno=B.Rno ORDER BY 3 DESC;**

| RNO | NAME | FEES |
|---|---|---|
| 3 | Harish | 3500 |
| 4 | PRIYA | 3000 |
| 1 | ANIK | 2500 |

## Union

- The number of columns selected from each table should be same
- The data types of corresponding columns selected from each table should be same

Table:Boys

| Rno | Name |
|---|---|
| 1 | Rahat |
| 2 | Harish |
| 13 | Tarun |

Table:Girls

| Rno | Name |
|---|---|
| 7 | Tara |
| 12 | Jaya |

```
SELECT Name FROM Boys WHERE Rno<13
UNION
SELECT Name FROM Girls WHERE Rno>7;
```

| Name |
|---|
| Rahat |
| Harish |
| Tara |

```
SELECT Rno,Name FROM Boys
UNION
SELECT Rno,Name FROM Girls;
```

| Rno | Name |
|---|---|
| 1 | Rahat |
| 2 | Harish |
| 13 | Tarun |
| 7 | Tara |
| 12 | Jaya |

```
SELECT Rno,Name FROM Boys
UNION
SELECT Rno,Name FROM Girls ORDER BY 2;
```

| Rno | Name |
|---|---|
| 2 | Harish |
| 12 | Jaya |
| 1 | Rahat |
| 7 | Tara |
| 13 | Tarun |

```
SELECT Rno,Name FROM Boys WHERE Rno<>1
UNION
SELECT Rno,Name FROM Girls ORDER BY 1 DESC;
```

| Rno | Name |
|---|---|
| 13 | Tarun |
| 12 | Jaya |
| 7 | Tara |
| 2 | Harish |