Types of Data
==========
We have two types of data.

1) Unstructured Data

2) Structured Data

1) Unstructured Data

A data which is not in readable format is called unstructured data.

In general, meaningless data is called unstructured data.

ex:

201 Lakemba SYD NSW AUS

2) Structured Data

A data which is in readable format is called structured data.

In general, meaning full data is called structured data.

ex:

Unit	Locality	City	State	Country
201	Lakemba	SYD	NSW	AUS

Management system
==========
Management system is a software which is used to manage the database.

Using management system we can perform following activities very easily.

1) Adding the new data

2) Modifying the existing data

3) Dropping the unnecessary data

4) Selecting the required data

Q) What is the difference between DBMS and RDBMS?

DBMS	RDBMS
It stands for Database Management System.	It stands for Relational Database Management System.
It stores the data in files.	It stores the data in tables.
It is not designed to store large amount of data.	It is designed to store large amount of data.
It provides support for single user at a time.	It provides support for multiple users at a time.
There is no data security.	There is high data security.
It does not support normalization.	It supports normalization. (It reduce code redundency)

Oracle
=======
It is one of the database which is used to store structured data.

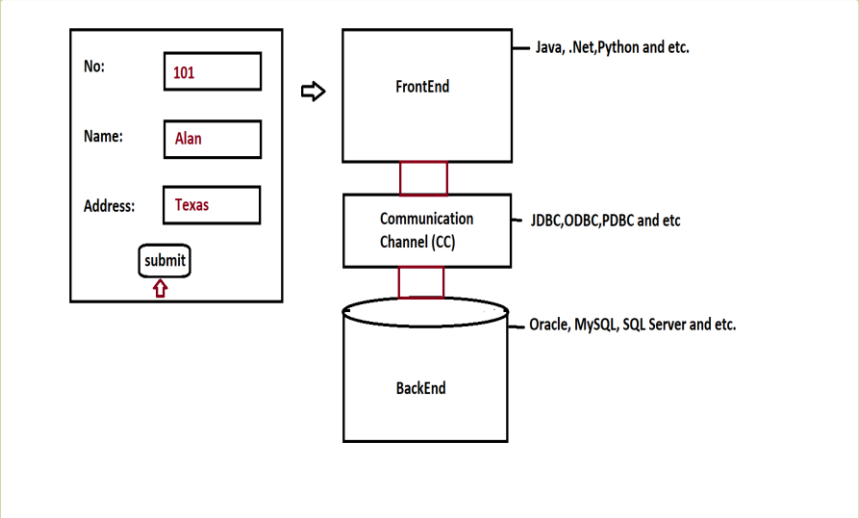
It is a RDBMS database.

It is product of Oracle Corporation.

It is classified into two types.

Oracle
|
|----------|
SQL — (Structured Query Language)
PL/SQL — (Procedural / Structured Query Language)

Client-Server Architecture
==========
Diagram: oracle1.1



Above architecture describes how our frontend data goes to backend.

FrontEnd

The one which is visible to the enduser to perform some operations is called frontend.
ex:
Java,.Net,Python,Perl,D2K and etc.

Communication channel

It acts like a bridge between frontend and backend.
ex:
JDBC- Java Database Connectivity
ODBC- Open Database Connectivity
PDBC- Python Database Connectivity

BackEnd

The one which is not visible to the enduser but it performs operations based on the instructions given by frontend is called backend.
ex:
Oracle, MySQL, SQL Server, MongoDB , NoSQL and etc.

SQL
====
SQL stands for Structured Query Language which is pronounce as SEQUEL.

This language is used to interact with oracle database.

It is a command based language.

It is a case insensitive language.

Every command must starts with verb.

Every command must ends with semicolon.

It is developed in the year 1972 by Mr.Codd (by IBM).

Sub languages of SQL
==========
We have five sub languages of SQL.

1) DDL (Data Definition Language)

2) DML (Data Manipulation Language)

3) DRL/DQL (Data Retrieve/Query Language)

4) TCL (Transaction Control Language)

5) DCL (Data Control Language)

1) DDL (Data Definition Language)

This language is used to maintain the objects in database.

It is a collection of five commands.

ex:

create, alter, drop , truncate and rename.

2) DML (Data Manipulation Language)

This language is used to manipulate the data which is present in database.

It is a collection of four commands.

ex:

insert, update, delete and merge

3) DRL/DQL (Data Retrieve/Query Language)

This language is used to retrieve the data from database.

It is a collection of one command.

ex:

select

4) TCL (Transaction Control Language)

This language is used to maintain the transaction of database.

It is a collection of three commands.

ex:

commit, rollback and savepoint

5) DCL (Data Control Language)

This language is used to control the access of data to the user.

It is a collection of two commands.

ex:

grant and revoke

Table

=====

Table is an object which is used to represent the data.

Table is a collection of rows and columns.

Data which is present in a table is a case sensitive.

ex:

NO	NAME	ADD
101	Alan	Florida
102	Jose	Texas
103	Lisa	Chicago

Here table contains 3 rows and 3 columns.

Oracle

=======

Version : 10g

Vendor : Oracle Corporation

Website : www.oracle.com/in/products

Software : Expression Edition

Port No : 1521

Username : system (default)

Password : admin

Download link :

https://drive.google.com/file/d/0B9rC21sL6v0td1NDZXpkUy1oMm8/view?usp=drive_link&resourcekey=0-aKooR3NmAh_eLo_qGw_inA

deEstablish the connection with database

==========

To execute any comment in database or to perform any operation on database we need to establish the connection with database software.

Once work with database is completed we need to close the connection with database.

ex:

SQL> connect

username : system

password : admin

SQL> disconnect

ex:

SQL> conn

username : system

password : admin

SQL> disc

ex:

SQL> conn system/admin

SQL> disc

create command
==========
It is used to create a table in a database.

syntax:
---
create table <table_name>(col1 datatype(size),col2 datatype(size),.....,
colN datatype(size));

ex:
create table student(sno number(3),sname varchar2(10),sadd varchar2(12));

create table dept(deptno number(3),dname varchar2(10),dloc varchar2(10));

create table emp(eid number(3),ename varchar2(10),esal number(10,2),
deptno number(3),job varchar2(10),comm number(8));

Describe command
==========
It is used to display the structure of a table.

syntax:
desc <table_name>;

ex:
desc emp;
desc dept;
desc student;

Insert command
==========
Insert command is used to insert row/record in a table.

syntax:
---
insert into <table_name> values (value1,value2,....,valueN);

ex:
insert into student values(101,'raja','hyd');

insert into student values('ravi',102,'delhi'); //invalid

insert into student values(102,'ravi'); // invalid

```
insert into student values(102,'ravi',null);
```

Note:
---
null represent undefined or unavailable.

approach2

```
insert into student(sno,sname,sadd) values(103,'ramana','vizag');

insert into student(sno,sname) values(104,'ramulu');
```

approach3

Using '&' symbol we can insert dynamic inputs.

ex:
```
insert into student values(&sno,'&sname','&sadd');
```

commit command
==========
It is used to make the changes permanent to database.

syntax:
```
commit;
```

dept table
==========
```
create table dept(deptno number(3),dname varchar2(10),dloc varchar2(10));

insert into dept values(10,'CSE','HYD');
insert into dept values(20,'ECE','PUNE');
insert into dept values(30,'MEC','VIZAG');
insert into dept values(40,'EEE','DELHI');
commit;
```

emp table
==========

```
create table emp(eid number(3),ename varchar2(10),esal number(10,2),
                                    deptno number(3),job varchar2(10),comm number(8));

insert into emp values(201,'Alan',9000,10,'Clerk',null);
insert into emp values(202,'Jose',19000,10,'Clerk',500);

insert into emp values(203,'Kelvin',45000,20,'HR',300);
insert into emp values(204,'Nelson',23000,20,'HR',900);

insert into emp values(205,'Lisa',21000,30,'Manager',500);
insert into emp values(206,'Jesicca',37000,30,'Manager',800);

commit;
```

select command
==========
It is used to retrieve the records from database table.

syntax:-

```
select * from <table_name>;
```

Here '*' means all rows and columns.

ex:

```
select * from student;
select * from dept;
select * from emp;
```

Projection

Selecting specific columns from database table is called projection.
ex:

```
select sno,sname,sadd from student;
select sno,sname from student;
select sname from student;
```

Arithmetic operations

In select command we can perform arithmetic operations.
ex:

```
select sno,sname,sadd from student;
select sno-100,sname,sadd from student;
select sno+100,sname,sadd from student;
```

Column alias

A userdefined heading given to a column is called column alias.

Column alias is temperory.

We can create column alias for any column.

ex:

```
select sno-100,sname,sadd from student;

select sno-100 as SNO,
        sname,sadd from student;

select sno as roll_no,
                sname as Name,
                        sadd as City from student;
```

Interview Queries

Q) Write a query to display all employees information from employee table?

```
select * from emp;
```

Q) Write a query to display employee id, employee name and employee salary from emp table?

```
select eid,ename,esal from emp;
```

Q) Write a query to display list of tables present in database?

```
select * from tab;
```

Q) Write a query to display logical database name?

```
select * from global_name;
```

Q) Write a query to display employee id,employee name, employee salary and annual salary from employee table?

```
select eid,ename,esal,esal*12 from emp;
```

Q) Write a query to display employee id,employee name, employee salary and annual salary as ANNUAL_SAL from employee table?

```
select eid,ename,esal,esal*12 as ANNUAL_SAL from emp;
```

where clause
============
It is used to select specific records from database table.

syntax:
----
select * from <table_name> where condition;

ex:
select * from student where sno=101;

select * from student where sname='raja';

select * from student where sadd='pune';

Interview Queries

Q) Write a query to display student information whose is living in hyderabad?

select * from student where sadd='hyd';

Q) Write a query to display employees information those who are working in 10 department?

select * from emp where deptno=10;

Q) Write a query to display employee information whose commission is null?

select * from emp where comm is null;

update command
==========
update command is used to update the values in a row.

syntax:
------
update <table_name> set <col_name>=value where condition;

ex:
update student set sname='rani' where sno=101;

```
update student set sname='Alan',sadd='USA' where sno=103;

commit;

rollback;
```

Note:

If we are not using any where clause then all rows will be udpated.

ex:

```
update student set sname='raja';
update student set sno=101;
update student set sadd='hyd';
```

delete command
==========

A delete command is used to delete the rows from database table.

syntax:

```
delete from <table_name> where condition;
```

ex:

```
delete from student where sno=101;
delete from student where sname='ravi';
delete from student where sadd='pune';
commit;
```

Note:

If we won't use where clause then all rows will be deleted.

ex:

```
delete from student;
delete from emp;
delete from dept;
```

Interview Questions
==========

Q) Write a query to terminate all the employees those who are working as a Clerk?

```
delete from emp where job='Clerk';
```

Q) Write a query to display employees information whose commision is null?

```
select * from emp where comm is null;
```

Q) Write a query to increment salary by 1000 whose employee id is 201?

```
update emp set esal=esal+1000 where eid=201;
```

Logical Operators
==========
Logical operators are used to declare multiple conditions in a query.

We have three logical opreators.

1) AND

2) OR

3) NOT

1) AND

It returns the records only if our condition is true.

All conditions must be from same row only.

ex:

```
select * from emp where eid=201 AND ename='Alan';

select * from emp where eid=201 AND ename='Ana'; //no rows selected

select * from emp where eid=201 AND ename='Jose';//no rows selected
```

2) OR

It returns the records only if one condition is true.

Here conditions can be from any row.

ex:
--

```
select * from emp where eid=201 OR ename='Alan';

select * from emp where eid=201 OR ename='Ana';

select * from emp where eid=201 OR ename='Jose';
```

3) NOT

It will return the records except the condition.

A <> symbol denoted as not operator.

ex:

```
select * from emp where NOT eid=202;

select * from emp where eid<>202;

select * from emp where job<>'Manager';
```

Interview Queries

Q) Write a query to display employees information whose salary is greater then 20000 and less then 50000?

```
select * from emp where esal>20000 AND esal<50000;
```

Q) Write a query to display employees information those who are not working in 10 department?

```
select * from emp where deptno<>10;
```

Q) Write a query to display employee information whose employee id is 201,202 and 203?

```
select * from emp where eid=201 OR eid=202 OR eid=203;
```

Between operator

==========

Between operator returns the records those who are in the range of values.

In between operator we will take lower limit then higher limit.

ex:

```
select * from student where sno between 101 and 105;
```

```
select * from emp where esal between 5000 AND 20000;

select * from emp where deptno between 10 AND 30;
```

IN operator
==========
IN operator is a replacement of OR operator.

In operator returns the records those who are matching in the list of values.

ex:

```
select * from student where sno IN(101,102,103);

select * from student where sname IN('raja','ravi','Alan');
```

Interview Queries

Q) Write a query to delete employees information whose employee id 201,202 and 203?

```
delete from emp where eid IN (201,202,203);
```

Q) Write a query to display employees information whose deptno number between 10 to 30?

```
select * from emp where deptno between 10 and 30;
```

Pattern Matching operators
==========
Pattern matching operators are used to select the letters from table.

Pattern matching operators take the support of like keyword.

We have two type of pattern matching operators.

1) Percentage (%)

2) Underscore (_)

1) Percentage (%)

Q) Write a query to display employees information whose emplyoee name starts with 'A' letter?

```
select * from emp where ename like 'A%';
```

Q) Write a query to display employees information whose employee name ends with 'n' letter?

```
select * from emp where ename like '%n';
```

Q) Write a query to display employees information whose employee name having middle letter as 'l' letter?

```
select * from emp where ename like '%l%';
```

2) Underscore (_)

Q) Write a query to display employee information whose employee name having second letter as 'l'?

ex:

```
select * from emp where ename like '_l%';
```

Q) Write a query to display employee information whose employee name having second last letter as 's'?

```
select * from emp where ename like '%s_';
```

Q) WRite a query to display employee information whose employee name having third letter as 'l' ?

```
select * from emp where ename like '__l%';
```

DDL commands

==========

1) create (tables)
2) alter (columns)
3) drop   (tables)
4) truncate (records)
5) rename (tables)

drop command

==========

It is used to drop the table from database.

syntax:

drop table <table_name>;

ex:

drop table student;

drop table dept;

drop table emp;

rename command

==========

It is used to rename the table name.

syntax:

rename <old_name> to <new_name>;

ex:

rename student to students;

rename emp to employees;

rename dept to departments;

truncate command

==========

It is used to delete the records permanently.

syntax:

truncate table <table_name>;

ex:

truncate table student;

truncate table emp;

truncate table dept;

Q) What is the difference between delete and truncate command?

delete	truncate
It deletes the rocords temperory.	It delete the records permanently.
We can rollback the data.	We can't rollback the data.
Where clause can be used.	Where clause can't be used.

Alter command

==========

Using alter command we can perform following activities very easily.

i) Adding new columns

ii) Modifying the columns

iii) Renaming the columns

iv) Dropping the columns

i) Adding new columns

Using alter command we can add new columns in a existing table.

syntax:

---

alter table <table_name> add (col_name datatype(size));

ex:

alter table student ADD (state varchar2(10));
alter table student ADD (pincode number(8));
alter table student ADD (state varchar2(10),pincode number(8));

update student set state='Telangana' where sno=101;

ii) Modifying the columns

Using alter command we can modify the columns.

We can increase or decrease the size of a column only when existing values are fit into new size.

syntax:

---

alter table <table_name> MODIFY (col datatype(size));

ex:

desc student;

alter table student MODIFY (state varchar2(15));

desc student;

We can change the datatype of a column only when that column is empty.

ex:

```
desc student;

alter table student MODIFY (pincode varchar2(8));

desc student;
```

iii) Renaming the columns

Using alter command we can rename the column name.

syntax:

```
alter table <table_name> rename column <old_name> to <new_name>;
```

ex:

```
alter table student rename column state to district;

alter table emp rename  column  esal to dailywages;

alter table emp rename column job to designation;
```

iv) Dropping the columns

Using alter command we can drop the columns.

syntax:

```
alter table <table_name> drop (col);
```

ex:

```
alter table student drop (state,pincode);
alter table student drop (district,pincode);
```

Copy of a table or duplicate table

==========

Using create and select command we can create copy of a table or duplicate table.

ex:

```
create table employees as select * from emp;
```

```
create table employees as select * from emp where deptno=10;
create table employee as select eid,ename,esal from emp;
create table employee as select * from emp where eid IN (201,202,203);
create table employee as select * from emp where comm is null;
create table employee as select * from emp where ename like 'A%';
create table employee as select * from emp where eid=201 and ename='Alan';
create table employee as select * fro emp where eid<>202;
```

cl scr
======
It is used to clear the output screen of SQL command prompt.

syntax:

```
cl   scr
```

Functions
==========
Functions are used to manipulate the data and give the result.

We have two types of functions.

1) Group Functions / Multiple row Functions

2) Scalar Functions / Single row Functions

1) Group Functions

Group functions are applicable for multiple rows.

We have following list of group functions.

ex:
sum(), avg(), max(), min(), count(*) and count(expression).

Q) Write a query to display sum of salary of each employee?

```
select sum(esal) from emp;
```

Q) Write a query to display average salary of each employee?

```
select avg(esal) from emp;
```

Q) Write a query to display highest salary from emp table?

```
select max(esal) from emp;
```

Q) Write a query to display least salary from emp table?

```
select min(esal) from emp;
```

Q) What is the difference between count(*) and count(expression)?

count(*)

It will return number of records present in a database table.

It will return null records.

ex:
--

```
select count(*) from emp;
```

count(expression)

It will return number of values present in a column.

It will not include null values.

ex:
--

```
select count(eid) from emp; // 6
select count(comm) from emp; // 5
```

userlist table
==========

```
drop table userlist;

create table userlist(uname varchar2(10),pwd varchar2(10));

insert into userlist values('raja','rani');

insert into userlist values('king','kingdom');

commit;
```

Q) Write a query to check given username and password valid or not?

```
select count(*) from userlist where uname='raja' AND pwd='rani';  //1

select count(*) from userlist where uname='raja' AND pwd='rani2';  //0
```

Dual table
==========
Dual table is a dummy table which is used to perform arithmetic operations and to see the current system date.

Dual table contains one row and one clumn.

ex:

```
select 10+20 from dual;

select 10*5 from dual;

select sysdate from dual;

select current_date from dual;
```

2)Scalar Functions
==========
We have following list of scalar functions.

i) Character functions

ii) Number functions

iii) Date functions

iv) Conversion functions

i) Character functions

We have following list of character functions.

upper()

It is used to convert the string to uppercase.
ex:

```
select upper('oracle training') from dual;
```

lower()

It is used to convert the string to lowercase.

ex:

```
select lower('ORACLE TRAINING') from dual;
```

initcap()

It is used to display the string with initial capital letter.

ex:

```
select initcap('oracle training') from dual;
```

lpad()

It is used to pad the characters to left side.

ex:

```
select  lpad('oracle',10,'z') from dual; //zzzzoracle
```

rpad()

It is used to pad the characters to right side.

ex:

```
select  rpad('oracle',10,'z') from dual; //oraclezzzz
```

ltrim()

It is used to trim the characters from left side.

ex:

```
select ltrim('zzoraclezz','z') from dual;// oraclezz
```

rtrim()

It is used to trim the characters from right side.

ex:

```
select rtrim('zzoraclezz','z') from dual;// zzoracle
```

trim()

It is used to trim the characters from both the sides.

ex:

```
select trim('z' from 'zzoraclezz') from dual;
```

concat()

It is used to concatinate two strings.

ex:

```
select concat('mega','star') from dual;

select concat(concat('mega','star'),'chiru') from dual;
```

replace()

It is used to replace the characters.
ex:
```
select replace('gOOgle','O','oo') from dual;
```

ii) Number functions

We have following list of number functions.

abs()

It returns absolute value.
ex:
```
select abs(-45) from dual; //45
```

sqrt()

It returns square root value.
ex:
```
select sqrt(25) from dual; // 5
```

power(A,B)

It returns power value.
ex:
```
select power(5,3) from dual; //125
```

greatest()

It return greatest value.
ex:
```
select greatest(6,9,2,4) from dual; // 9
```

least()

It returns least value.
ex:

select least(6,9,2,4) from dual; //2

ceil()

It returns ceil value.
ex:
select ceil(10.9) from dual; // 11

select ceil(10.2) from dual; // 11

floor()

It returns floor value.
ex:
select floor(9.8) from dual; // 9

select floor(9.1) from dual; // 9

round()

It returns nearest value.
ex:
select round(10.5) from dual; //11

select round(10.4) from dual; // 10

trunc()

It removes decimal numbers.
ex:
select trunc(10.56) from dual; // 10

Working with Date values
==========
Every database software support date values.

Every database software support date values in different date patterns.

ex:
oracle - dd-MMM-yy
mysql - yyyy-MM-dd

emp1 table

---
drop table emp1;

create table emp1(eid number(3),ename varchar2(10),edoj date);

insert into emp1 values(101,'Alan','01-JAN-24');

insert into emp1 values(102,'Jose',sysdate);

insert into emp1 values(103,'Lisa',current_date);

commit;

Assignment
==========
Q) Write a java program to find out lucky number?

iii) Date functions

We have following list of date functions.

ADD_MONTHS()

It is used to add the months in a given date.
ex:
select ADD_MONTHS('07-OCT-24',4) from dual;

MONTHS_BETWEEN()

It returns number of months between two dates.
ex:
select MONTHS_BETWEEN('01-JAN-24','07-OCT-24') from dual;
select MONTHS_BETWEEN('01-JAN-24','01-NOV-24') from dual;
select ABS(MONTHS_BETWEEN('01-JAN-24','01-NOV-24')) from dual;

NEXT_DAY()

It will return next date of a given day in a week.
ex:
select NEXT_DAY(sysdate,'sunday') from dual;
select NEXT_DAY(sysdate,'monday') from dual;

LAST_DAY()

It will return last date of a month.

ex:

```
select LAST_DAY('13-FEB-24') from dual;

select LAST_DAY(sysdate) from dual;
```

iv) convertion functions

---

convertion function is used to convert from one datatype to another datatype.

ex:

TO_CHAR() function

We have two pseduo's for TO_CHAR

1) number TO_CHAR

It takes '9' in digits and dollar or euro's symbol.

ex:

```
select eid,ename,esal from emp;

select eid,ename,TO_CHAR(esal,'9,999') from emp;

select eid,ename,TO_CHAR(esal,'99,999') from emp;

select eid,ename,TO_CHAR(esal,'$99,999') from emp;

select eid,ename,TO_CHAR(esal,'$99,999') as ESAL  from emp;
```

2) date TO_CHAR

```
select TO_CHAR(sysdate,'dd-MM-yyyy') from dual;

select TO_CHAR(sysdate,'yyyy-MM-dd') from dual;

select TO_CHAR(sysdate,'HH:MI:SS') from dual;

select TO_CHAR(sysdate,'dd-MM-yyyy  HH:MI:SS') from dual;

select TO_CHAR(sysdate,'year') from dual;

select TO_CHAR(sysdate,'month') from dual;
```

```
select TO_CHAR(sysdate,'day') from dual;
```

Integrity Constraints
==========
Constraints are the rules which are applied on the tables to achieve accuracy and quality of data.

We have five types of contraints.

1) NOT NULL

2) UNIQUE

3) PRIMARY KEY

4) FOREIGN KEY

5) CHECK

Constraints can be created at two levels.

i) column level

ii) table level

1) NOT NULL

NOT NULL constraint does not accept null values.

NOT NULL constraint can accept duplicate values.

NOT NULL constraint can be created only at column level.

column level

```
drop table student;

create table student(sno number(3) NOT NULL,sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(null,'ravi','delhi'); //invalid
```

```
insert into student values(101,'ravi','delhi');
```

Note:

NOT NULL constraint can be created for multiple columns.

ex:

```
drop table student;

create table student(sno number(3) NOT NULL,
                                    sname varchar2(10) NOT NULL,
                                            sadd varchar2(12) NOT NULL);

insert into student values(null,'raja','hyd'); //invalid

insert into student values(102,null,'delhi'); //invalid

insert into student values(103,'ravi',null); //invalid

insert into student values(104,'ramana','pune'); // valid
```

2) UNIQUE

UNQIUE constraint does not accept duplicates.

UNIQUE constraint can accept null values.

UNIQUE constraint can be created at column level and table level.

column level

```
drop table student;

create table student(sno number(3) UNIQUE,sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(null,'ravi','delhi');

insert into student values(101,'ravi','delhi'); //invalid
```

table level
---
drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12), UNIQUE(sno));

insert into student values(101,'raja','hyd');

insert into student values(null,'ravi','delhi');

insert into student values(101,'ravi','delhi'); //invalid

Note:

UNIQUE constraint can be applied to multiple columns.

ex:
---
drop table student;

create table student(sno number(3) UNIQUE,sname varchar2(10) UNIQUE,sadd varchar2(12) UNIQUE);

insert into student values(101,'raja','hyd');

insert into student values(101,'ravi','delhi'); //invalid

insert into student values(102,'raja','delhi'); //invalid

insert into student values(103,'ramana','hyd'); //invalid

3) PRIMARY KEY

Primary key is a combination of NOT NULL and UNIQUE constraint.

Primary key does not accept null values and duplicate values.

A table can have only one primary key.

Primary key constraint can be created at column level and table level.

column level

```
drop table student;

create table student(sno number(3) primary key,sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(null,'ravi','delhi'); //invalid

insert into student values(101,'ravi','delhi'); //invalid
```

table level

```
drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12), primary key(sno));

insert into student values(101,'raja','hyd');

insert into student values(null,'ravi','delhi'); //invalid

insert into student values(101,'ravi','delhi'); //invalid
```

4) FOREIGN KEY

Foreign key is used to establish the relationship between two tables.

This relationship is called parent and child relationship or master and details relationship.

To establish the relatinship between two tables a parent table must primary key or unique constraint and
child table must have foreign key.

Foreign key will accept only those values which are present in primary key.

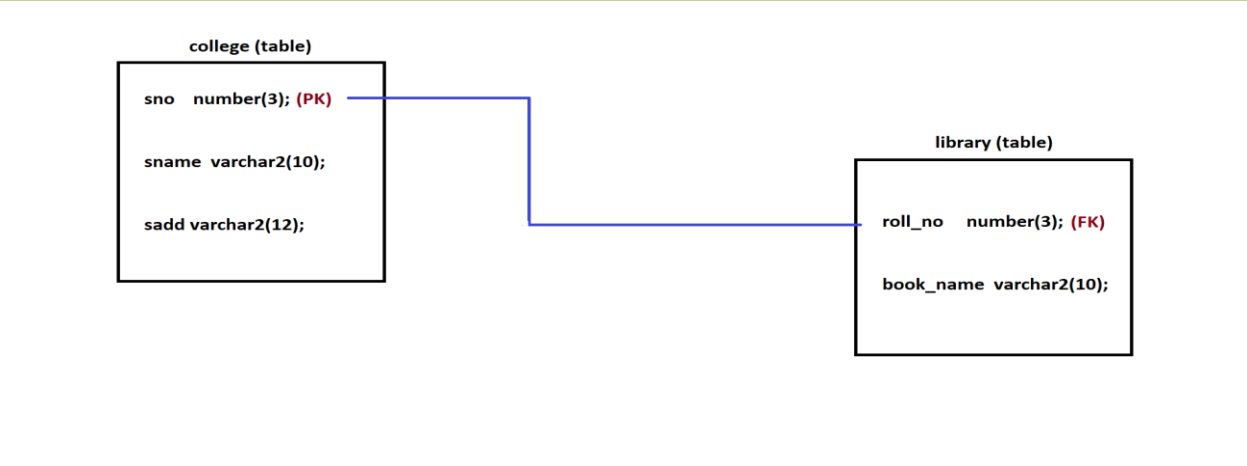
Foreign key name may or may not match with primary key but datatype must match.

Foreign key will accept duplicates and null values.

college table

```
drop table college;

create table college(sno number(3) PRIMARY KEY,sname varchar2(10),sadd varchar2(12));

insert into college values(101,'raja','hyd');

insert into college values(102,'ravi','delhi');

insert into college values(103,'ramana','vizag');

commit;
```

Diagram: oracle5.1



library table

```
drop table library;

create table library(roll_no  number(3) REFERENCES college(sno), book_name varchar2(10));

insert into library values(101,'java');

insert into library values(102,'oracle');

insert into library values(103,'html');

insert into library values(103,'CSS');

insert into library values(null,'Spring');

insert into library values(104,'hibernate'); //invalid
```

In order to drop the table, first we need to drop library table then college table.

ex:

```
drop table library;
drop table college;
```

5) CHECK

CHECK constraint describes domain of column.

Here domain means what type of value a column must accept.

Check contraint can be created at column level and table level.

column level

```
drop table student;

create table student(sno number(3),sname varchar2(10),smarks number(3) check(smarks<=100));

insert into student values(101,'raja',78);

insert into student values(102,'ravi',100);

insert into student values(103,'ramana',289);

commit;
```

column level

```
drop table student;

create table student(sno number(3),sname varchar2(10),smarks number(3) check(smarks between 0 and 100));

insert into student values(101,'raja',78);

insert into student values(102,'ravi',100);

insert into student values(103,'ramana',289); //invalid

commit;
```

column level

```
drop table student;

create table student(sno number(3),sname varchar2(10) check (sname=lower(sname)),
                                                    smarks number(3));

insert into student values(101,'raja',78);

insert into student values(102,'RAVI',100); //invalid

insert into student values(103,'RaMaNa',289); //invalid

commit;
```

column level

```
drop table student;

create table student(sno number(3),sname varchar2(10) check (sname=upper(sname)),
                                                    smarks number(3));

insert into student values(101,'raja',78); //invalid

insert into student values(102,'RAVI',100);

insert into student values(103,'RaMaNa',289); //invalid

commit;
```

column level

```
drop table student;

create table student(sno number(3),sname varchar2(10),smarks number(3),
                                                                    check (sname=upper(sname)));

insert into student values(101,'raja',78); //invalid

insert into student values(102,'RAVI',100);

insert into student values(103,'RaMaNa',289); //invalid
```

```
commit;
```