# Introduction to relational model:-

**Relation** nothing but a table. In relational model the data will be stored in the form of tables or relations.

# Ex:-

employee table or instance of relation

Fields or attributes or columns

|  |  |  |
| --- | --- | --- |
| **Eid** | **Ename** | **Salary** |
| **1001** | Eone | 1000 |
| **1002** | Etwo | 2000 |
| **1003** | Ethr | 3000 |
| **1004** | Efour | 4000 |
| **1005** | Efive | 5000 |

**Domain:-** it is defined as the set of values or value set for a column or an attribute.

**Ex:-** domain of Eid in the above relation is {1001,1002,1003,1004,1005} **Degree:-** it is defined as the total number of columns of a given relation. **Ex:-** degree of Employee table is ‘3’.

**Cardinality:-** it is defined as number of rows or tuples of a given relation.

**Ex:-** cardinality of Employee table is ‘5’

**Instance:-** it is defined as collection of data (rows) that is going into database relation at a particular point of time.

**Tuple:-** it is nothing but a row of a table **Attribute:-** it is a property to represent an entity. **Importance of null values:-**

* A ‘NULL’ is a term used to represent a missing value.
* Null is undefined, unknown, unavailable and it is not equal to zero or a space.
* The regular operators like +, -, \*, %, =, <, >, <=, >=, will be fail with null values.

# Why should we avoid placing of null values into DB:-

* All arithmetic and comparison operators will fail with null values i.e. if we add a column to the null value column then the result will become null only.
* A null will occupy large space in a databases.
* We use two operators with the null values.
  1. is null
  2. is not null

**Ex:-** select ename from emp where comm is null

We have the following functions to handle with the null values.

1. nvl()
2. nvl2()
3. coalesce()

**nvl(expr/column, default value):-**takes two arguments

this function returns first argument value if the first argument is not null, if it is null then it return the 2nd argument value.

**Ex:-** select nvl(comm,0) from emp;

In the output of above query if comm is null then the default value (2nd argument) i.e. 0 will be displayed, if comm is not null then that value is displayed as it is.

# Output:-

nvl(comm,0)

The actual comm column from emp is

Comm

----------

300

500

1400

-

null values

-

-

-

-

-

-

14 rows selected

------------------

0

300

500

0

1400

0

0

0

0

0

0

0

0

0

14 rows selected.

# nvl2(expr1/column1, expr2/column2, expr3/column2):-

if the first argument value is not null then this function returns the value of 2nd argument, if the first argument value is null thoutpien this function returns the value of 3rd argument

**ex:-** select nvl2(comm, sal+com,sal) from emp;

# output:-

nvl2(comm, sal+com, sal)

800

1900

1750

2975

2650

2850

2450

3000

5000

10000

1100

950

3000

1300

14 rows selected.

# Coalesce(expr1/column1, expr2/column2, expr n/column n):-

It takes ‘n’ arguments.

This function accepts two or more arguments and returns the first not null value in the list. If all the arguments contain null values then this function returns a null value.

**Ex:-** select coalesce(20,30,null) from dual;

# Output:-

20first not null value.

select coalesce(null,null,30) from dual;

# Output:-

30 first not null value in the argument list.

# Simple database schema:-

* database schema is the skeleton structure that represents logical view of the entire database.
* It defines how the data is organized and how the relations among them are associated.
* It formulates all the constraints that are to be applied on the data.
* A database schema defines it entities and the relationship among them. It contains a descriptive detail of the database.

# Key constraints:-

Constraint is a condition or rule on a column that restricts value in the database. Constraints are used to full fill data integrity.

# Types of constraints:-

1. Not null
2. Unique
3. Primary key
4. Foreign key/ references
5. Check

Another classification of constraints

1. **Column level:-** If we define a constraint on a column immediately after the column definition, then it is called as column level or inline specification of constraint.
2. **Table level:-**If a constraint is defined after all the columns defined or as a part of table definition then that is called table level or out of line specification of constraint.
3. **not null:-**This constraint should be declared in column level or inline only. If we define this constraint with a column, then the column will not allow null values into it.

**Ex:-** create table student(sid number(6) not null, sname varchar(10) not null);

while inserting data into sid and sname columns of student we should not leave these columns empty (i.e. we need a supply a value)

**Ex:-** insert into student values(1001,’’);

if we execute the above command in the sqlprompt then it will give the following error. ‘can not insert null into STUDENT.SNAME’ because we tried to place a null value into not null column i.e. sname

1. **unique:-**This constraint can be defined with column level and table level also. This is used to allow only unique values into a column. If unique key is defined on number of columns then it is called as composite unique key. Composite unique key should be defined in table level only. A unique key column can contain any number of null values. Unique key is also known as candidate key.

# Ex:-

1. create table promotions1(promo\_id number(6) unique, promoname varchar(10));
2. create table promotions2(promo\_id number(6) unique, promoname varchar(10), unique(promo\_id));
3. create table promotions3(promo\_id number(6) unique, promoname varchar2(10), unique(promo\_id,promoname));

**3 primary key:-**this constraint can be defined with table level or column level. Primary key= notnull+unique

It is also used to allow only unique values and it will not allow any null values. If this constraint defined on more than one column then it is called composite primary key. Maximum number of columns in a composite key are 32.

# Ex:-

1. create table loation1(location\_id number(6) primary key,address varchar2(10)not null);
2. create table loation2(location\_id number(6) primary key,address varchar2(10)(location\_id);
3. create table loation3(location\_id number(6) primary key,address varchar2(10)(location\_id,address);

consider the following data is there is the locations1 table location\_id address

1. hyd
2. pune
3. chirala

* if we want to insert the following row into locatiions1 insert into locations1 values(2,’delhi’);

it will give the following error

‘unique constraint violated LOCATIONS1.LOCATION\_ID’ , because

we are trying to insert ‘2’ which is already there in location\_id column of locations1

* insert into locations1 values(‘’,’xyz’); it will give the following error

‘cannot insert NULL into LOCATIONS1.LOCATION\_ID’ because we are trying to place null into lovations\_id

1. **foreign key:-**this can be defined in column level and table level also. It is also called referential integrity constraint. It is used to establish relationship between two or more tables using primary key and foreign key relationship. If number of columns associated with foreign key then is called composite foreign key.

* the table contain the foreign key is called child or detailed table (master, details)
* the table contain the primary key (referenced by) is called parent or master table.
* To maintain this relationship between tables first we should create master table and then the master and then the master\_details or details table.
* ‘References’ clause should be used when the foreign key constraint is inline or column level. When the foreign key constraint is out of line or table level then we have to use ‘foreign key’ clause.
* The column on which the foreign key dependent is called referenced key column.the referenced key column may be in the same table or in the other table.

# Ex:-

create table college(cid number(4)primary key, cname varchar2(10),cplace varchar2(10));

create table college(sid number(4)primary key, sname varchar2(10), cid number(4) references college(cid));

college

|  |  |  |
| --- | --- | --- |
| cid | Cname | Cplace |
| 101 | Cone | Cp1 |
| 102 | Ctwo | Cp2 |
| 103 | cthr | Cp3 |

student

|  |  |  |
| --- | --- | --- |
| **Sid** | **Sname** | **cid** |
| 1001 | **Sone** | **101** |
| 1002 | Stwo | 102 |
| 1003 | Sthr | 101 |
| 1004 | Sfour | 103 |
| 1005 | Sfive | 102 |
| 1006 | ssix | 103 |

if I want to insert a row into student shown below.

Insert into student values(1007,’sseven’,105); it will give the following error ‘parent key not found’ because in the parent table college 105 for cid is not there.

1. **check:-**This can be defined in column level or table level. It is used to define a condition on a column while creating the table. One column can be associated with any number of check constraints.

# Ex:-

create table dept(deptno number(3)primary key check(deptno between 10 and 99));

create table dept(deptno number(3)primary key, dname varchar2(10), check(deptno between 10 and 99));

if we want to insert a value ‘120’ into deptno of dept then we get

‘check constraint voilated’ because deptno column will take the values b/w 10 and 99

# Data integrity:-

It is a state in which all the data values are stored in the database. This will increase thw quality of data in database.

1. **Entity integrity:-** it defines row as a unique entity for a particular table. It enforces through the primary key of a table.

**Ex:-** student

|  |  |  |
| --- | --- | --- |
| Sid | Sname | Marks |
| 1001 | Sone | 60 |
| 1002 | Sone | 60 |
| 1003 | Stwo | 70 |
| 1004 | sthr | 80 |

* + Sid in the above table is taken as primary key by using this only we can identify entities or rows of student table uniquely. Each and every row is different from other row.
  + If we try to insert a row which takes sid column value 1002, we can not because primary key enforces us not to insert duplicate value.

1. **Domain integrity:-** it validates the entities for a given column. It enforces through data types, check constraint, foreign key, default, notnull.

**Ex:-** student

|  |  |  |
| --- | --- | --- |
| Sid | Sname | Marks |
| 1001 | Sone | 60 |
| 1002 | Sone | 70 |
| 1003 | Stwo | 80 |
| 1004 | sthr |  |

* + The database of sid column is number but if we want to insert into a string value into sid, the system will not accept it because while creating student table we have given the sid column as number datatype.

1. **Referential integrity:-** it preserves the defined relationship between the tables when the rows are inserted or deleted.
   * Referential integrity enforces inserting rows to a related table if there is no associated row is primary or master table.

**Ex:-** college

|  |  |
| --- | --- |
| cid | Cname |
| 101 | Cone |
| 102 | Ctwo |
| 103 | cthr |

student

|  |  |  |
| --- | --- | --- |
| sid | sname | cid |
| 1001 | Sone | 101 |
| 1002 | Sone | 101 |
| 1003 | Sthr | 103 |
| 1004 | Sfive | 102 |
| 1005 | Ssix | 103 |
| 1006 | Ssven | 102 |

* + If we want to insert into a row into student table by giving 105 for cid column we are unable to insert it because in the parent table college 105 is not there for cid.

# Data types:-

* + Each value in oracle is maintained by a data type.
  + The value of one data type if different from other data type.
  + The data type defines the domain of values that each column can caontain

# Character data types:-

These store character data Different character data types are

1. char
2. varchar2
3. **char data type:-** it specifies fixed length character string. Size should be specified. If the data is less then original specific size, blank spaces are applied. The default length is 1byte and maximum length is 200 bytes.

Ex:- char(10);

1. **varchar2 data types:-** it specifies the variable length character string. It occupies only that space for which the data is supplied. The maximum size is 1byte and the maximum size is 400 bytes.

**Ex:-** varchar2(10); **Number data types:-**

1. number(p,s)

p precision, range is 1 to 38 s scale, range is -84 to 127 **ex:-** number(8,3);

1. **float:-** it is used to specify floating point values. It specifies decimal precision 38.
2. **Long data types:**- these are used to store very large text strings. A single table can have only one long column.

# Data and time data type:-

* 1. **Date:-** it is used to store date information. The default date format in oracle is DD-MM-YYY
  2. **Time:**- this is used to store time information. It has atleast 8 positions embedded in single quotes. ‘HH:MM:SS’

**Ex:-** :- 11:07:05

* 1. **Time stamp:-** it includes both time and date along with minimum 6digits representing decimal fraction of seconds.

The format is ‘DD-MM-YYYY HH:MM:SS’

**Ex:-** ’31-05-1950 01:02:05 123456’

1. **Large object data types:-** these can store large and un structural data like text, image, video and special data.

the max size is upto 4 GB the types are

* 1. BLOB(binary large object)
  2. CLOB(character large object) Maximum size is 4 GB

**Note:-** in place of along data type which is deprecated, now a days we are using LOB data datatype.

1. **Raw and long raw data types:-** these are used to store binary data or byte strings. These are variable length data types. They are mostly used to store graphics, sound documents etc.

# Types of SQL commands:-

1. **Data definition language(DDL):-** it is used to define the database schema. The commands used under this languages are:-
   1. create
   2. after
   3. drop

# syntaxes and examples:- syntax:-

1. create table <table-name>(col1 datatype[size] constraints list, col2 datatype[size] constraints list, );

**ex:-** create table student (sid number(4)primary key, sname varchar2(10)not null);

1. **after:-** used to alter the table definition

# alter with add option:- syntax:-

alter table <table-name> add <col-name> datatype[size]

**ex:-** alter table dept add loc1 varchar2(10);

# alter with drop option:-

**syntax:-** alter table <table-name>drop column <col-name>;

**ex:-** alter table dept drop column loc1;

# after with modify option:-

syntax:- alter table <table-name>modify <col-name>datatype[size];

**ex:-** alter table dept modify loc varchar2(10);

# alter with rename option:-

**syntax:-** alter table<table-name> rename column<old col name> to <new column name>

**ex:**- alter table rename column loc to location

1. **drop:-** used to drop a database table permanently.

**Syntax:-** drop table<table-name>

**Ex:**- drop table dept;

1. **Data manipulation language(DML):-** these are used to manipulate the data in the databases. The commands used in the languages are
   1. Insert
   2. Update
   3. Delete

# Syntaxes and examples:-

1. **Insert:-** used to insert rows into a table

**Syntax:**- insert into <table-name>(col1,col2,---,coln)values(val1,val2, ,valn);

**Ex:**- insert into dept(deptno,dname,loc)values(50,’xyz’,’hyd’);

1. **Update:-** used to update rows of table

**Syntax:**-update <table-name>set <column-name>=<value>where <col-name>=<value>

**Ex:**- update dept set dname=’pqr’ where deptno=50;

1. **Delete:-** used to delete rows from a table

**Syntax:**- delete from <table-name> where <col-name>=<value>;

**Ex:**- delete from dept where deptno=50;

1. **Data query language(DQL):-** it is used to extract data from database tables. The command comes under the language is
   1. Select

# Syntax:-

Select <col-list>,<group functions>from <table-name> where

<condition>groupby<column>having<group condition>orderby<column-name>

# Ex:-

Select deptno, sum(sal), max(sal), min(sal), avg(sal) from emp

Where job=’clerk’ group by deptno having avg(sal)>1000 order by deptno;

**4. Data control languages:-** these commands control the user access to the database. The commands comes under these languages are

1. Grant
2. Revoke

**Grant:-**used to grant the permissions to the user on the db tables. **Syntax:-** grant <priviliges-name>ON <object name>to<user-name> **Ex:-** grant select, insert, delete on emp to operators;

**Revoke:-** used to take back the permissions from the user.

**Syntax:-** revoke<priviliges-name>ON <object name>from<user-name>

**Ex:-** revoke insert, delete on emp from operators;

1. **Data administrative language(DAL):-**these commands are used for audit, the commands are
   1. Start audit;
   2. Sleep audit;
2. **Transaction control language(TCL):-** these commands are used to control the transactions
   1. Commit
   2. Rollback
   3. Savepoint

# Syntaxes:-

1. **commit;**
2. **rollback;**
3. **rollback to<save point name>;**

**Table definitions:-** we can define a table using create statement and we can after the definition of the table using alter command which will comes under DDL(data definition language commands) commands

* examples and syntaxes to create a new table and modifying the definition of created table with different options of alter command shown in pgno 9

**Different DML operations:-** these operators are used to modify a database table data. The commands under these operations are insert, delete, update , syntaxes and examples are in pgno 9.

# Basic SQL querying(select and project):-

* select statement is used to get the information or data of a database table i.e. columns and rows information.
* To get all the columns and all the rows of a table emp, we use Select \* from emp;
* To get the selected columns (projection) information we need to mention those columns in the selection list as shown below.

Select empno.ename,job from emp;

* As the emp table is containing 14rows the above two queries will display the 14 rows of emp, but the difference is first query will select and display all the columns of emp and second query display only 3 selected columns information.

**Using where clause:-** we can delimit or reduce rows returned from a table using select statement by using ‘where’ clause

**Ex:-** select \* from emp where sal>1200;

The above query will get all the columns information but it delimit the rows based on the where condition sal>1200 (i.e. we get <14 rows)

**Ex:-** select empno,ename,job from emp where sal>1200;

The above query will get three columns (not all columns) information depending upon the condition sal>1200

# Arithmetic and logical operators:-

1. **Arithmetic operators:-** these operators are used to do binary operations between two cols/exprs. These are +, -, \*, /

**Ex:**- 1. Select sal+500 from emp;

1. select 12\*sal from emp where deptno=20;
2. **logical operators:-** the operators are used in the conditions These are AND, OR NOT

# Ex:-

* 1. Select ename,job from emp where deptno=10 and sal>1000;
  2. Select empno,sal from emp where deptno=10 or deptno=20;
  3. Select ename,sal from emp where not(deptno=20);

# SQL functions:-

 Date and type functions Numeric functions

String conversion functions

1. **Date and type functions:- Sysdate:-** used to get current date select system for dual;

# output:-

SYSDATE

16-SEP-17

**Add-months():-** used to add months to a given date **Ex:-** select add\_months(’09-MAR-98’,2)from dual; **Output:-**

ADD\_MOTHS()

09=MAY-98

**Months\_between():-** used to find the month between the two dates given to it. **Ex:-** select round(months\_between(’09-MAR-98’,’24-JAN-96’))from dual; **Output:-**

MONTHS-BETW---

20

**Last\_day:-**used to return the last day of the month in the given date.

**Ex:-** select last\_day(’01-SEP-17’)from dual;

# Output:-

LAST\_DAY( )

30-SEP-17

**greatest():-** return the greatest value in the given list

**ex:-** select greaest(10,20,30)from dual;

# output:-

GREATEST( )

30

**least():-** this function returns the minimum value in the given list.

**Ex:-** select least(30,20,40)from dual;

# Output:-

LEAST( )

20

# Number functions:-

**abs():-**used to return the absolute value of the given number

**ex:-** select abs(-28)from dual;

# output:-

ABS(-28)

28

**ceil():-** used to return the highest inter of given number

**ex:-** select ceil(1.5)from dual;

# output:-

CEIL(1.5)

- 2

**floor():-** used to return the lowest integer of given number

**ex:-** select floor(1.4)from dual;

# output:-

FLOOR(1.4)

1

**mod():-** used return the reminder **ex:-** select mod(20,7)from dual; **output:-**

MOD(20,7)

6

**power():-** used to find the power of a number

**ex:-** select power(2,3) from dual;

# output:-

POWER(2,3)

8

**sign():-** used to find the sign of given number. If the number is +ve then this function returns 1, if it is –ve number then it returns -1

**ex:-** select sign(-20) from dual;

# output:-

SIGN(-20)

-1

**sqrt():-** used to find the square root of given number

**ex:-** selct sqrt(9) from dual;

# output:-

SQRT(81)

9

**trunc():-** used to truncate a value. **Ex:-** select trunc(2.56,1)from dual; **Output:-**

TRUNC(2.56,1)

2.5

**round():-** used to get rounded value from given value.

**Ex:-** select round(2.56,1)from dual;

# Output:-

ROUND(2.56,1)

2.6

**exp():-** used to find the exponential of a number

**ex:-** select exp(10)from dual;

# output:-

EXP(10)

22026.4658

# String functions:-

**chr():-** used to convert given number to a character

**ex:-** select chr(76)from dual;

# output:-

CHR(76)

L

**ascii():-** used to convert character to equivalent asci value

**ex:-** select asci(‘Z’)from dual;

# output:-

ASCII(‘Z’)

90

**concat():-** used to combine two given strings. It takes max two arguments only.

**Ex:-** select concat(‘abc’,’xyz’)from dual;

# Output:-

CONCAT(‘abc’, )

abcxyz

**initcap():-** used to display initial letter of a given string captal.

**Ex:-** select initcap(‘welcome’)from dual;

# Output:-

INITCAP( )

Welcome

**upper():-** used to convert upper case. **Ex:-** select upper(‘welcome’)from dual; **Output:**

UPPER( )

WELCOME

**lower():-** used to convert lower case.

**Ex:-** select lower(‘WELCOME’)from dual;

# Output:

LOWER( )

welcome

**lpad():-** used to pad left side with given character **ex:-** select lpad(‘abcde’,10,’\*’)from dual; **output:-**

LPAD(‘A )

\*\*\*\*\*abcde

**rpad():-** used to pad right side with given character

**ex:-** select rpad(‘abcde’,10,’\*’)from dual;

# output:-

RPAD(‘A )

abcde\*\*\*\*\*

**ltrim():-** used to trim left side characters of a given string with matching characters

**ex:-** select ltrim(‘aabbccaa’,’aa’)from duall;

# output:-

LTRIM(‘A )

bbccaa

**rtrim():-** used to trim right side characters of a given string with matching characters

**ex:-** select rtrim(‘aabbccaa’,’aa’)from duall;

# output:-

RTRIM(‘A )

aabbcc

**replace():-** used to replace the part of the given string with the given character or string

**ex:-** select replace(‘abacd’.’a’,’xy’)from dual;

# output:-

REPLACE(‘A )

xybxycd

**substr():-** used to display the part of the string **ex:-** select substr(‘abcdefg’,3,6)from dual; **output:-**

SUBSTR(‘A )

bcdefg

**instr():-** used to return an integer depending upon the matching string occurrence in the given string.

**Ex:-** select instr(‘aabbccaaccbbccaabbab’,’bb’,2,3)from dual;

# Output:-

INSTR(‘AA )

17

# String conversion functions:-

**to\_char():-** converts a given number into character **ex:-** select to\_char(1210.73,’9999.9’)from dual; **output:-**

TO\_CHAR( )

1210.7

**to\_date:-** converts given string into date

**ex:-** select to\_date(‘2003/03/09’,’YYYY/MM/DD’)from dual;

# output:-

TO\_DATE( )

09-03-03

**to\_number:-** converts given string into a number

**ex:-** select to\_number(‘6737.777’,’9999.999’)from dual;

# output:-

TO\_NUMBER(‘6 )

6737.777