DBMS & RDBMS USING ORACLE

PREPARED BY :
P.V.Thummar

CHAPTER – 1
DBMS OVERVIEW, SQL & SQL * PLUS

INTRODUCTION OF DBMS

- DATA : “The name of any object is called data.”
- Database : “ A database is a collection of related & meaningful data.
- Example :

Database

Data

Address book
House name
Area
City
Pin code

Introduction to DBMS

- DBMS : DBMS is a system (software) which is used to manage the database."
- For creating & using database in our computer effectively, we need a software which can insert, update, delete and process the data.
- To manage all the above tasks, we need a software which is called database management system.
- Examples : oracle, ms access, FoxPro, FoxBASE, Sybase, dbase 3+ etc.

BENEFITS OF DBMS

- WE can reduce duplication of data.
- Stored data can be shared to multiple users.
- We can enter only accurate data without error.
- Security of data can be maintain.

INTRODUCTION OF RDBMS

- Definition:
- RDBMS is a DBMS, which is used to set relationship between related tables.
- OR
- RDBMS is a DBMS, that is based on relational model, created by dr. E.F.Codd.

RDBMS

- RDBMS stores the data in the form of related tables. It means, if we have multiple tables, which are related to each other, we can get collected data from them.
- Opposite to it, we can spread a single table across several tables.
- Examples of RDBMS: oracle, MS SQL Server,
- Sybase SQL server, and IBM's DB2.

DBMS v/s RDBMS

	DBMS	RDBMS
1	Relationship between 2 tables are maintained programmatically.	Relationship between 2 tables can be specified at the time of table creation.
2	DBMS does not support client/server architecture.	RDBMS supports client/server architecture.
3	It does not support distributed database.	It supports distributed database.
4	In DBMS, there is no security of data.	It provides multiple levels of security 1. OS level 2. Command level 3. Object level.
5	Each table is given an extension.	Many tables are grouped into one database.
6	It satisfies less than 7 to 8 rules of dr. E.F.Codd.	It satisfies more than 7 to 8 rules of dr. E.F. Codd.

12 rules by Dr. E.F.Codd for RDBMS

Rule 1	Information rule
	All data should be in tabular format. All information(including metadata)
Rule 2	Guaranteed access rule
	All data should be accessible easily without any problem. The system should provide a guarantee, that the data stored in it will be accessible in the future. Table Name + Primary Key(Row) + Attribute (column).
Rule 3	Systematic treatment of null values.
	A field of a table should be allowed to remain empty, which is called null value. It means, if user don't want to enter any value in a field, it should allowed. Primary key must not be null

12 rules by Dr. E.F.Codd for RDBMS

Rule 4	Dynamic online catalog based on relational model
	The structure of a table can be accessed by the same tools (commands) that are used to access the table data. description of the complete Database
Rule 5	Comprehensive data sub language rule.
	The database must support one language that includes all functions for data creation, manipulation, data integrity and transaction commands. Most of RDBMS uses SQL (structure query language) as their supportive language. Example: SQL, etc.
Rule 6	View updating rule.
	If multiple users at different places work on the same database, and any user make changes in database, that changes should be displayed to all other users. updatable by the system as well.

12 rules by Dr. E.F.Codd for RDBMS

Rule 7	High – level insert, update and delete
	In relational database, many tables are related to each other. So, any operations like insert, update or delete applied in one table, should be automatically applied to all related tables. Set operation like Union, Intersection and minus should also be supported.
Rule 8	Physical data independence
	The user should be un aware about physical storage of database. There is no need for user to know that where the database is stored in hard disk. some file supporting table is renamed or moved from one disk to another,
Rule 9	Logical data independence
	When the structure of data is changed, the view of data should not be changed. join of the two tables. This rule is most difficult to satisfy.

12 rules by Dr. E.F.Codd for RDBMS

Rule 10	Integrity independence
	When user input the data, it must be checked that the inputted data are valid or not. Hence, only accurate data are allowed to store. Key and Check constraints, trigger etc, should be stored in Data Dictionary. This also make **RDBMS** independent of front-end.
Rule 11	Distribution independence
	The user should be un aware about database is distributed or not. A database should work properly regardless of its distribution across a network.
Rule 12	Non sub version rule
	As we know that, every dbms or rdbms support any one language. So, data & structure should be modified by same language. There is no requirement of subversion or any other language. system it should not be able to subvert or bypass integrity rules to change the data.

E-R Diagram

- This is one of the data model that was developed by P.P.chen in 1976.
- Entity relationship approach on the entity relationship model and is one of the best known approaches to the problem of DB design.
- It represents

- 1.ENTITIES.
- 2.ATTRIBUTES.
- 3.RELATIONS.

Entities

- A person, thing concept about which you want to store data.
- Ex. student , courses.
- symbol :Entity that represent rectangle.

relation

- Used to represent a connection among related entities like a link.
- Ex. student –to– college , Client –to– server
- Symbol:

attributes

- Data values of an entities.
- Characteristics of person ,thing or concept.
- Ex. stud_id, stud_name, stud_class.
- Symbol:

First normal form (1NF)

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Second normal form (2NF)

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Third Normal form (3NF)

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

Boyce Codd normal form (BCNF)

Example: Suppose there is a company wherein employees work in **more than one department.** They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

INTRODUCTION OF ORACLE

- In June 1970, Dr E.F.Codd worked for IBM, introduced a model sponsored by IBM, called "relational model".
- By using this relational model, it became possible to make relationship between different tables.
- In 1979, they establish a company called "Relational Software Incorporation", and released the first commercial version of this relational model called "SQL".
- After some time, the name of "Relational Software Incorporation" was changed to "ORACLE CORPORATION".

INTRODUCTION TO SQL

- The language developed by IBM to work on data of relational model, was originally called "**Structure English Query Language(ESQUEL)**".
- Later on, the word "English" was being removed and language was known as "**Structure Query Language(SQL)**".

INTRODUCTION OF SQL

- The SQL is generated by combining different small database languages like DDL,DML, DQL and DCL. These small languages contains the commands which are used to work on data.
- Development of SQL was governed by standards.
- In 1986, SQL was approved by ANSI standards.
- In 1987, SQL was approved by ISO standards.
- In 1992, new version is called "SQL-92".
- In 1999, new version is called "SQL-99" or SQL-3.

INTRODUCTION OF SQL

YEAR	EVENT
1970	Relational model (SQL) introduced by Dr. E.F.Codd in IBM.
1979	The company “Relational Software Incorporation” was established, and commercially released first version of SQL.
1986	Approved by ANSI
1987	Approved by ISO
1992	Standard revised and called SQL-92
1999	Standard revised and called SQL-99 or SQL-3.

Introduction to SQL * plus

- Sql *plus is an oracle tool (product) and submits the sql statements to the oracle server for execution.
- It provides command window where sql and sql * plus commands can be written by user.
- When we start sql *plus, it gives you a prompt window for entering user name, password and connection string (host name).
- After logging into oracle using sql*plus, sql commands can be executed.

SQL V/S SQL*PLUS

NO	SQL	SQL * PLUS
1	SQL is a language for communication with oracle to access the data	Sql *plus recognizes the sql statements and sends them to the server.
2	Sql is approved by ANSI	Sql *plus is a tool of oracle. So that, no standards is given.
3	Sql manipulates the data & structure of table.	Sql *plus does not manipulate the values of table.
4	Sql is entered one or more lines into sql buffer.	Sql *plus passes one line at a time, but not stored in sql buffer.
5	Sql does not have continuous character.	Sql *plus uses a dash(-) sign as a continuous character.
6	It can not be abbreviated.	It can be abbreviated.
7	Sql uses termination character	It does not use termination character.
8	Sql uses functions to format data.	It uses commands to format data.

Components of SQL or Commands of SQL

- SQL commands are divided into following 4 categories.
- (1) DDL (Data Definition language)
- (2) DML (Data manipulation language)
- (3) DCL (Data control language)
- (4) DQL (Data Query Language)

Components of SQL

- (1) DDL (Data definition language)
- It is a set of SQL commands used to create, modify and delete the database structure but not data.
- Ex. : create, alter, drop, truncate, rename.
- (2)DML (Data manipulation language)
- DML commands are used to work and manage the data of database.
- Ex. : insert, update, delete.

Components of SQL

- (3) DCL (Data control language)
- These commands are used to control the data access from user. The user can access the data according to the rights given to user.
- Ex. : grant, revoke, commit, rollback, savepoint.
- Commit, rollback & save point commands are also called TCL(Transaction control commands).

Components of SQL

(4) DQL (Data Query Language)

It is used to get the data from database and display on screen. It contains only one command “SELECT”, and it is the heart of SQL.

Operators in SQL

- An operator is a symbol which is used to perform operation on data and give result".
- There are 2 types of operators.
- (1) unary operators
- (2) binary operators
- (1)unary operators :
- The unary operators has only 1 operand.
- Ex. : +2
- (2) binary operators :
- The binary operators has 2 operands.

Operators in SQL

Arithmetic operators
+
-
*
/
Concatenation Operators
||
Relational operators
<
>
<=
>=
=
!= , <> , ^=

SQL basic data types.

- (1)CHAR : the CHAR data type is a fixes-length alphanumumeric character string.
  - Size can range from a minimum of 1 to a maximum of 2000 characters.
  - If the data is shorter then the defined length, it is space padded on the right.
- For ex. CHAR data type of 5 size. CHAR(5)

s	y	a	m	

Data types of SQL

- Char (size):
- This data type is used to store string values.
- The size in bracket shows the number of characters the cell can hold. Maximum size of this data type is 255.
- It is space padded data type.
- [2] Varchar (size) / varchar2 (size):
- This data type is also used to store string values where the size longer than char size.
- Maximum size of this data type is 4000 char.
- It is non- padded data type.

Count…..

- **VARCHAR**: A VARCHAR datatype is currenty synonymous with the VARCHAR2 datatype.
- **VARCHAR2 :The** VARCHAR2 datatype is a variable-length alphanumeric string having a maximum lenth of size bytes.
  - Only require the amount of space needs to store the data. Its columns can store up to 4000 byte.
- Ex.VARCHAR2 datatype of 5 size .VARCHAR2(5).

G	O	p	i	

Data types of SQL

- Date:
- This data type is used to store data & time. The default format is DD-MON-YY.
- It stores the time in 24 hour format. Default date for date field is first day of current month.
- Number (p,s):
- It is used to store numeric data. Valid values are zero, negative & positive with decimal point(.).
- In bracket, p shows precision which is maximum length of numeric data. S shows scale, which is number of digits to the right side of decimal point.
- Maximum length of p (precision) is 38 digits.

Data types of SQL

- Long :
- It is used to store binary data in ascii format. Maximum length of long data type is 2gb.
- This type of data can not be used by any commands of sql, and only one long value can be stored in one table.
- Raw / long raw:
- It is also used to store binary data such as picture or image.
- Maximum length of raw is 255 bytes and long raw is 2gb.

ASSIGNMENT –1 (chapter –1)

Q. NO.	QUESTIONS
1	Explain DBMS & RDBMS
2	Differentiate : DBMS v/s RDBMS
3	List out and explain Dr. E.F.Codd's rules for RDBMS
4	Write a short not on oracle.
5	Write a short note on SQL
6	What is SQL * plus? Explain difference between SQL & SQL * plus.
7	Explain components of SQL.
8	Explain data types of SQL.

CHAPTER – 2

MANAGING TABLES AND DATA, DATA CONTROL AND TRANSACTION CONTROL COMMAND

CREATING A NEW TABLE

- “create table” command is used to create a new table.
- Syntax:
- Create table <table name>
- (<col1> data type (size), <col2> data type (size), …..);
- “create table” is a command to create a new table.
- <table name> is a name of table given by user.
- In brackets, we have to declare columns (fields) required in a table.
- <col> is a name of column, given by user specified by data type and (size).
- Example:
- Create table student(rollno number(2), name varchar(10));

Create

- Syntax :
  - Create table table_name
  - (colums1 datatype(size),
  - colums2 datatype (size),
  - … … …
  -)

  Ex.: create table student (roll number(5),name varchar2(10),salary number(10))

Displaying a structure of table

- “describe” command is used to display the structure of a table.
- Syntax :
- Describe <table name>;
- Example:
- Describe student;

INSERTING DATA INTO TABLES

- "INSERT INTO" command is used to insert data into created tables.
- It creates a new row in a table, and stores the values in columns, specified by user.
- Syntax:
- Insert into <table name>
- (col1, col2, col3, …..) values
- (val1, val2, val3, …..);
- The sequence, data type & size of values must match with columns specified in first bracket.
- Example:
- Insert into biodata (rollno, name) values
- (1, 'amit');

insert

- Syntax :
  - Insert into table_name values
  - (val_1,val_2,……)

Ex.: insert into student values(1,'paras')

alter

- Syntax :
  - Alter table table_name ADD(column_name);
  - Ex. alter table student add (mobile number(10))

update

- Syntax :
  - Update table_name SET columsname where columename.

Ex.: UPDATE emp set id=10 where city='jnd'

delete

- Syntax :
  - Delete from table_name where record_name

Ex.: delete from student where roll=2

truncate

- Remove table record
- Syntax :
  - Truncate table table_name

Ex.: truncate table emp

drop

- Remove fully table.
- Syntax :
  - Drop table table_name

Ex.: drop table student;

rename

- Newname of Tablename

- Syntax :
  - Rename old table_name to new table_name

Ex.: remane student to stud;

Sql query

- **To display all row and all column**
  - Ex. select *from student;
- **To display selected column and all row**
  - Ex. select id,name,city from student
- **To display all column and selected row**
  - Ex. select *from student where id >=5
  - Ex. select *from student where name like'M%'

Count…..

- **To display selected column and selected rows**
  - Ex.Select name,city,mobile from student where pincode=365440
- **To table discribe**
  - Ex. Desc student.
- **To specific row update**
  - Ex.update student set city=‘jnd’,pincode=‘365440’ where rollno=2

Count..

- **To adding new column**
  - Ex.alter table student add(mobile number(15))
- **To modify existing column**
  - Ex.alter table student modify(mobile varchar2(15))
- **To deleting a column**
  - Ex. alter table student drop column salary

Arithmetic operator

- They are arithmetic operator are +, -, and *.
- Addition:
  - Ex. select name,salary +5000 from student where id=1
- Substraction:
  - Ex. select name,salary -2000 from student where id=1
- Multiplication:
  - Ex. select name,salary *500 from student where id=2

Logical operators

- They are logical operator are AND,OR,NOT.
- AND:
  - Ex. select *from student where name='aaa' AND salary=2000
- OR:
  - Ex. select *from student where id=2 OR salary=2000
- NOT:
  - Ex. select *from student where id NOT LIKE 2

In (માં હોવું) / not in operator

- IN :
  - Ex. select *from student where id IN(1,2,6,4)
- NOT IN :
  - Ex. select *from student where name NOT IN('aaa','bbb','ccc')

Bitween operator

- Ex. select *from student where name between 'aaa' and 'fff‘
- Ex. select *from student where id between 1 and 5

DATA constraints

- The constraints are used in oracle to implement integrity rules of a relational database and to implement data integrity at the individual-column level.

- Types of data constraints:
- (1)input constraints.
- (2)output constraints.

Operator

In Operator

- The operator IN is used to check if given values matches with any values inside the list or not.
  - Ex.
  - select *from emp where job IN('analyst','manager')
  - select *from emp where deptno IN(30,10)

Between operator

- Between operator may be used to specify lower and upper values.
- If the condition values falls between specified upper and lower values(inclusive of both),then row will be display.
- Ex.
  - select ename,deptno,job,sal from emp where sal BETWEEN 1000 and 2000
  - select ename,deptno,job,sal from emp where sal BETWEEN 1000 and 2000 AND deptno=30

Like operator

- In query ‘pattern matching’can be done using % along with LIKE operator.

- Ex.
  - select *from emp where job LIKE '%salesman‘
  - select *from emp where deptno LIKE '30'

SET operators

SQL supports three types of operators

- [1]UNION
  - Select id from emp UNION select id from dept
- [2]MINUS
  - Select id from emp MINUS select id from dept
- [3]INTERSECT
  - Select id from emp INTERSECT select id from dept

- create table emp (emp_id number (3),emp_name varchar2(10),dept_id number(3))
- insert into emp values(1,'ram',10)
- insert into emp values(2,'syam',20)
- insert into emp values(3,'kam',30)
- insert into emp values(4,'man',10)

[1]union

- The UNION operators combines the results of two SELECT statement and weeds out the duplicate rows at the same time.

- Ex. Select dept_id from emp UNION select dept_id from dep

constraints
I/O constraints
Business rule constraints
Primary key
Foreign key
Not null
Unique key
check

-: Features Primary key :-

- 1)primary key is a column or a set of columns that uniquely identifies a row.
  - its main purpose is the 'record uniqueness'.
- 2)primary key will not allow duplicate values.
- 3)primary key will also not allow null values.
- 4)primary key is not compulsory but it is recommended.
- 5)primary key cannot be LONG or LONG RAW data type.
- 6)only one primary key is allowed per table.
- 7)one table can combine up to 16 column in a composite primary key.

- ex.
- create table cust (roll number(3) primary key ,name varchar2(10),city varchar2(10));

-: Features Foreign key :-

- 1)Foreign key is a column that references a column of a table and it can be the same table also.
- 2)parent that is being references has to be unique or primary key.
- 3)child may have duplicates and nulls but unless it is specified.
- 4)Foreign key constraint can be delete provided no child record exist.
- 5)master table can not be updated if child record exist.

- ex.
- create table cust_master (cust_no number(3) references cust(roll) ,name varchar2(10),city varchar2(10));

-:Unique key constraint :-

- 1)Unique key will not allow duplicate values.
- 2)Unique key index is created automatically.
- 3)a table can have more than one Unique key which is not possible in primary key.
- 4)Unique key can combine up to 16 columns in a composite Unique key.
- 5)Unique key cannot be LONG or LONG RAW data type.

- ex.
- create table college (roll_no number(5) UNIQUE,stud_name varchar2(10),serial number(3));

joins

joins

- To combine the data from two or more tables ,it is called a JOIN.
- Table are joined on columns that have the same data type and data width in the tables.
- In joins Rows in one table can be joined to rows in another table according to common values (that are relational values)existing in corresponding columns, that is usually primary and foreign key columns.

Types of joins

- [1]Inner
- [2]Outer (Left,Right)
- [3]Self

[1] Inner join

- Inner joins are also knows as equl joins ,because here the where statement generally compares two columns from the two table with equivalence operator(=).
- Ex.
  - select empno,ename,job,dname,loc from emp e,dept d where e.deptno = d.deptno;
  - Select empno,ename,job,dname,loc from emp inner join dept using(deptno)

Table

- Emp

EMPNO	ENAME	JOB	MGR	DEPTNO
111	ram	analyst	444	10
222	syam	clark	333	20
333	jigar	manager	111	10
444	kam	engineer	222	40

- dept

DEPTNO	DNAME	LOC
20	acco	bgsr
10	mrk	hyd
30	hr	mumbai

- Create table dept (deptno number(3),dname varchar2(10),loc varchar2(10))
- Insert into dept values (10,'mrk','hyd')
- insert into dept values (20,'acco','bgsr')
- Insert into dept values (30,'hr','mumbai')
- select *from dept

- Create table emp (empno number(3),ename varchar2(10),job varchar2(10),mgr number(3),deptno number(3));
- Insert into emp values (111,'ram','analyst',444,10)
- Insert into emp values (222,'syam','clark',333,20)
- Insert into emp values (333,'jigar','manager',111,10)
- Insert into emp values (444,'kam','engineer',222,40)
- Select *from emp

[2] Outer join

- An outer join allows you to join two tables and results even when the second table doesn't have ant records corresponding with the first.
- **LEFT-OUTER Join**
  - The left outer join can be used to returns all the rows from the first table even if there are no matches in the second table.
  - **Ex.** Select empno,e.ename,job,d.dname,d.deptno from emp e,dept d where e.deptno = d.deptno(+)

EMPNO	ENAME	JOB	DNAME	LOC
222	syam	clark	acco	bgsr
333	jigar	manager	mrk	hyd
111	ram	analyst	mrk	hyd
444	kam	engineer	-	-

RIGHT-OUTER Join

- The right outer join can be used to returns all the rows from the second table even if there are no matches in the first table.

- Ex. Select empno,e.ename,job,d.dname,d.deptno from emp e,dept d where e.deptno (+) = d.deptno

EMPNO	ENAME	JOB	DNAME	LOC
111	ram	analyst	mrk	hyd
222	syam	clark	acco	bgsr
333	jigar	manager	mrk	hyd
-	-	-	hr	mumbai

Full outer joiin

- This will display the all matching records and the non-matching record from both tables.

- Ex.
  - Select empno,ename,job,dname,loc from emp e full outer join dept d

Cross join

- This will gives the cross product.

- Ex.
  - Select empno,ename,job,dname,loc from emp cross join dept;

Function

- select lower ('DHANAK COLLEGE')"lowercase" from dual;
- select upper ('dhanak college')"uppercase" from dual;
- select INITCAP ('oracle database') "capital" from dual;
- select SUBSTR ('SubString',3,5) "sibstromg" from dual;
- select LPAD ('Program',10,'*') "LPAD" from dual;
- select RPAD ('Program',11,'*') "RPAD" from dual;
- select LTRIM ('NISHA','N') "LTRIM" from dual;
- select RTRIM ('PROGRAM','M') "RTRIM" from dual;
- select chr(67) || chr(65) || chr (84) from dual;
- select length ('Dhanak College')"length" from dual;

- select INSTR ('This a Program' , 'a') from dual;

BUILT-IN-FUNCTON

- **Function use:**
- -performing arithmetic and general calculation on data,
- -converting or transformation data,
- -modifying individual data,
- -manipulating a group of rows, and
- -formatting columns.

-:: STRING FUNCTION ::-

- (1) lower()
- select lower ('DHANAK COLLEGE')"lowercase" from dual;
- o/p:dhanak college

- (2) upper()
- select upper ('dhanak college')"uppercase" from dual;
- o/p:DHANAK COLLEGE

- (3)INITCAP()
- select INITCAP ('oracle database') "capital" from dual;
- o/p:Oracle Database

- (4)SUBSTR()
- select SUBSTR ('SubString',3,5) "sibstromg" from dual;
- o/p:bStri

- (5)LPAD()
- select LPAD ('Program',10,'*') "LPAD" from dual;
- o/p:***Program

- 6)RPAD()
- select RPAD ('Program',11,'*') "RPAD" from dual;
- o/p:Program****

- (7)LTRIM()
- select LTRIM ('NISHA','N') "LTRIM" from dual;
- o/p:ISHA

- (8)RTRIM()
- select RTRIM ('PROGRAM','M') "RTRIM" from dual;
- o/p:PROGRA

- (9)Ascii()
- select chr(67) || chr(65) || chr (84) from dual; o/p=CAT
- select ascii('D') from dual; o/p=68
- select ascii('h') from dual; o/p=104

- (10)Length()
- select length ('Dhanak College') from dual;
- o/p=14

- (11)INSTR()
- select INSTR ('This a Program' , 'a') from dual;
- o/p:6

- select ABS(-15.11) "absolute" from dual;
- select power(15,2) "raised" from dual;
- select round(15.22222,3) "round" from dual;
- select sqrt(225) "squre root" from dual;
- select mod(15,7) "mod" from dual;
- select FLOOR(24.8) "floor" from dual;
- select ceil(24.8) from dual;
- select exp(1) from dual;
- select greatest(10,10,1,3,44,55,0)from dual;
- select least(10,10,1,3,44,55,0)from dual;

-:: NUMERIC FUNCTION ::-

- (1)ABS()
- select ABS(-15.11) "absolute" from dual;
- o/p:15.11

- (2)power()
- select power(15,2) "raised" from dual;
- o/p:225

- (3)round()
- select round(15.22222,3) "round" from dual;
- o/p:15.222

- (4)SQRT()
- select sqrt(225) "squre root" from dual;
- o/p:15

- (5)MOD()
- select mod(15,7) "mod" from dual;
- o/p:1

- (6)FLOOR()
- select FLOOR(24.8) "floor" from dual;
- o/p:24

- (7)CEIL()
- select ceil(24.8) from dual;
- o/p:25

- (8)exp()
- select exp(1) from dual;
- o/p:2.718

- (9)GREATEST()
- select greatest(10,10,1,3,44,55,0)from dual;
- o/p:55

- (10)LEAST()
- select least(10,10,1,3,44,55,0)from dual;
- o/p:0

GROUP Function

- select count (*) from employee;
- select MIN(salary) from employee;
- select MAX(salary) from employee;
- select sum(salary) from employee where empno=1;
- select sum(salary) from employee;
- select AVG(salary) from employee;

: GROUP Function OR AGGREGATE Function :

- (1)COUNT()
- select count (*) from employee;
- o/p:5

- (2)MIN(0
- select MIN(salary) from employee;
- o/p:000

- (3)MAX()
- select MAX(salary) from employee;
- o/p:999

- (4)SUM()
- select sum(salary) from employee where empno=1;
- select sum(salary) from employee;
- o/p: Total salary display

- 5)AVG()
- select AVG(salary) from employee;
- o/p: avg to the table

DATE FUNCTION

- select to_date(sysdate,'dd-mm-yy')from dual;
- select add_months(sysdate,1)from dual;
- select last_day(sysdate)from dual;
- select months_between('31-dec-2015','31-dec-2017')from dual;
- select next_day(sysdate,'sunday')from dual;
- select extract(year from sysdate)from dual;
- select extract(month from sysdate)from dual;
- select systimestamp from dual;

-:: DATE FUNCTION ::-

- (1)to_date()
- select to_date(sysdate,'dd-mm-yy')from dual;
- o/p:06-AUG-15

- (2)add_month()
- select add_months(sysdate,1)from dual;
- o/p:06-OCT-15

- (3)last_date()
- select last_day(sysdate)from dual;
- o/p:31-AUG-15

- (4)month_between()
- select months_between('31-dec-2015','31-dec-2017')from dual;
- o/p:-24

- (5)next_day()
- select next_day(sysdate,'sunday')from dual;
- o/p:09-AUG-15

- (6)round()
- select extract(year from sysdate)from dual;
o/p:2015
-
- select extract(month from sysdate)from dual; o/p:8

- (7)systimestamp()
- select systimestamp from dual;
- 0/p:07-AUG-15 03.01.27.541000 PM +05:30

Displaying records of table

- “select” command is used to display inserted records from table.
- There are 4 ways to use select command.

1. All rows & all columns:

Select * from <table name>;

This will display all rows & all columns of table. * shows all columns.

2. All rows & selected columns:

Select col1, col2, … from <table name>;

For displaying selected columns, we have to specify the names of columns which we want to display.

Displaying records of table

3. Selected rows & all columns:

Select * from <table name> where <condition>;

- For displaying selected rows from a table, “where” clause is used with condition to filter the data from table.
- <condition> with where clause is used with where clause is checked for all records one by one. Oracle will display only those records, that satisfy the specified condition.

4. Selected rows & selected columns:

Select col1, col2, … from <table name> where <condition>;

- This will display only specified columns. It will display only those records, which satisfy the condition.

Deleting (removing) records from table

- “delete” command is used to removes records from a table.
- We can delete records 2 ways.

1. Deleting all rows:

Delete from <table name>;

- This will remove all rows from a table.

2. Deleting selected rows:

Delete from <table name> where <condition>;

- This will remove only those records, which satisfy the condition.

Modify (update) data of table.

- “update” command is used to modify the values of a table. We can set new value in existing value.
- Syntax :

Update <table name>
Set <column>= <expression>
where <condition>;

- Set is a clause, which indicates that which column data you want to modify with new value.
- The condition is required to modify only specific records that match with condition.
- If we do not use condition, all records of a table will be modified.

CREATING A NEW TABLE FROM ANOTHER TABLE

- Syntax :
- Create table <new table name>
- (<col1>,<col2>,…) as select
- <col1>,<col2>,… from <table name>;
- “new table name” is a new table which you want to create. “table name” is existing table.
- This will select all records from “table name” to “new table name”.
- If we do not want to store all records, but only selected records, use where clause.

INSERTING DATA INTO A TABLE FRO ANOTHER TABLE.

- SYNTAX:
- Insert into <table name 2> select
- <col1>,<col2>,… from <table name 1>
- Where <condition>;
- According to the condition, the specific records will be selected from <table1> and inserted into <table2>.

DATA CONSTRAINTS

- Data constraints:
- “Constraints are set of rules which are enforced on data being stored in a table.”
- To apply constraints to a table column, we have to apply condition on that table column. Only the data which satisfies the condition, should be stored into table. If the data fails to satisfies the condition, it must be rejected.

DATA CONSTRAINTS

- Following are types of constraints:

(1)Primary key
(2)Foreign key
(3)Unique key
(4)Not null
(5)Check constraint
(6)Default constraint

DATA CONSTRAINTS

- (1) PRIMARY KEY:
- “Primary key is a column in a table which is used to uniquely identify each row(record) in a table.”
- The column which we set as primary key, does not allow null value and duplicate value.
- It means, we can not leave that column blank. We have to enter value in that column compulsory, and that value must be unique.
- Syntax:
- <column name> data type (size) primary key
- Example :
- Create table biodata (rno number(2) primary key, name varchar(10));

DATA CONSTRAINTS

- Features of primary key:
- 1. its purpose is the record uniqueness.
- 2. it will not allow duplicate value.
- 3. it will not allow null value.
- 4. it is not compulsory, but it is recommended.
- It is used to set relationship between tables.
- Only one primary key should allow per table.
- More than one primary keys in a table are called composite key.
- On table can contain maximum up to 16 columns in a composite primary key.

DATA CONSTRAINTS

- (2) Foreign key:
- “Foreign key is a column whose values are derived from primary key of some other table.”
- Foreign key is a reference of primary key, and is used to represent relationship.
- The table which contains foreign key, is called foreign table or detail table. And the table which contains primary key is called primary table or master table.

DATA CONSTRAINTS

- Syntax :
- <column name> data type (size) references <table name> (column name)
- [on delete cascade / on delete set null] ;
- Example :
- Create table marks (rollno number(2) references biodata (rno));
- In above example, “rollno” is a foreign key and its reference is “rno” of biodata table.

DATA CONSTRAINTS

- Oracle will display an error message when we delete a record from master table, and same record exists in detail table.
- for this reason, we have to set ON DELETE CASCADE option. So that, delete operation in master table will remove all corosponding records from all detail tables.
- When ON DELETE SET NULL option is set, delete operation in master table will set “null” value in foreign key of detail table.

DATA CONSTRAINTS

- Features of foreign key:
- 1. foreign key is a reference of primary key of another table.
- 2. foreign key may contains duplicate & null values.
- It can be specified in child table, but not in parent table.
- We can not insert or update a record in detail table if corresponding record do not exists in master table.

DATA CONSTRAINTS

- (3) unique key:
- “unique key is a column, which contains only unique value, not duplicate value.”
- The difference between primary key and unique is that, unique key contains null value and primary key does not contains null value.
- Syntax :
- <column> data type (size) unique;

DATA CONSTRAINTS

- 4. NOT NULL:
- Null value means empty, which is different from blank or zero.
- “not null constraint ensures that the table column can not be left empty.”
- When a column defined as a not null, it shows that the value must be entered into that column. We can not remain it empty.
- It contains duplicate value, but not null value.
- Syntax :
- <column> data type (size) not null;

DATA CONSTRAINTS

- 5. CHECK CONSTRAINT
- Check constraint is used to set validation for table columns.
- It is used to apply user defined condition on column. When w enters a value in column, check constraint checks that the value is according to condition or not.
- It returns true or false. If value is proper, it is stored. Otherwise, whole record is rejected.
- Syntax:
- <column> data type (size) check (<condition>);
- Example:
- M1 number(3) check (m1>=0 and m1<=100);

DATA CONSTRAINTS

- 6. DEFAULT CONSTRAINT:
- Default constraint is used to set default value for a column.
- When enters the record, and left this column empty, oracle automatically load this column with specified default value.
- The data type of default value must match the data type of the column.
- Syntax:
- <column> data type size <value>;
- Example:
- City varchar(15) default ‘junagadh’;

LOGICAL OPERATORS

- 1. AND
- 2. OR
- 3. NOT
- 1. AND:
- AND operator allows to use sql statements based on two or more conditions.
- Oracle will process all rows in a table and display result when all of the conditions specified using AND operator are satisfied.
- It can be used with select, update and delete commands.
- Example:
- Select * from biodata where city = ‘junagadh’ and course = ‘bca’;

LOGICAL OPERATORS

- 2.OR :
- OR operator allows to use sql statements based on any one of the conditions are met.
- Oracle will process all rows in a table and display the result only when any one of the conditions specified using OR operator is satisfied.
- Example :
- Select * from biodata where course = ‘bca’ or course = ‘bscit’;

LOGICAL OPERATORS

- 3. NOT :
- Oracle will process all rows in a table and display only those records that do not satisfy the specified condition.
- Example:
- Select * from biodata where not (city='junagadh');

SPECIAL OPERATORS

- 1. IN
- 2. NOT IN
- 3. BETWEEN AND
- 4. LIKE

SPECIAL OPERATORS

- 1. IN :
- The equal to (=) operator is used to compare a single value with another single value. But when we want to compare single value with list of value, then IN operator is used.
- IN operator is used instead of multiple OR conditions.
- Example :
- Select * from biodata where city in (‘junagadh’,’rajkot’,’morbi’);

SPECIAL OPERATORS

- 2. NOT IN :
- NOT IN operator is opposite of IN operator. This will select all rows where values do not match any values in the list.
- Example:
- Select * from studinfo where rollno not in (1,3,4,6,8);

SPECIAL OPERATORS

- 3. BETWEEN … AND :
- BETWEEN AND operator is used to select the data within the range of values. We have to specify lower and upper limit with “between” and “and” key words to select data in range.
- Only those rows are selected, which contains values in the range from lower limit to upper limit.
- Example:
- Select * from biodat where rollno between 3 and 8;

SPECIAL OPERATORS

- 4. LIKE :
- Like operator is used to compare one string value with another string value, which is not identical (fix).
- There are 2 wildcard characters (symbols) are used with like operator.
- (1) % : it allows to match any string of any length
- (2) _ : it allows to match on a single character.
- Example:
- Select * from biodata where name like ‘a%’;
- Select * from biodata where name like ‘%s_’

JOIN

- JOIN is used to bind data together across the table.
- Sometimes, it is require to get data from multiple tables. Join is used to achieve this by using a single sql statement.
- Tables are joined on column that have same data type and size in the tables.
- There are 4 types of join:
- 1. inner join
- 2. outer join (left, right)
- 3. cross join
- 4. self join

JOIN

- 1. inner join (equi join):
- Inner join is also called “equi join” because the where statement compares two columns from two tables with equal to (=) operator.
- Inner join returns common rows from both tables.

- Syntax (ANSI style):
- Select col1,col2,col3,…. From <table1> inner join <table2> on
- <table1>.col = <table2>.col where <condition>;

- Syntax (THETA style):
- Select col1,col2,col3,…. From <table1>,<table2> where
- <table1>.col = <table2>.col and <condition>;

JOIN

- 2. OUTER JOIN :
- Outer join is used to select all records from one table and only matching values from another table.
- There are 2 types of outer join
- (1) outer left join
- (2) outer right join

JOIN

- (1) OUTER LEFT JOIN :
- This will display all rows from the table, defined at left side of = operator and only matching rows from right side table.

- Syntax (ANSI style):
- Select col1,col2,col3,…. From <table1> left join <table2> on <table1>.col = <table2>.col
- where <condition>;

- Syntax (THETA style):
- Select col1,col2,col3,…. From <table1>,<table2> where <table1>.col = <table2>.col (+)
- and <condition>;

JOIN

- (2) OUTER RIGHT JOIN :
- This will display all rows from the table, defined at right side of = operator and only matching rows from left side table.

- Syntax (ANSI style):
- Select col1,col2,col3,…. From <table1> right join <table2> on <table1>.col = <table2>.col
- where <condition>;

- Syntax (THETA style):
- Select col1,col2,col3,…. From <table1>,<table2> where <table1>.col (+) = <table2>.col
- and <condition>;

JOIN

- 3. CROSS JOIN :
- Cross join combines every row fro left table with every row in right table.
- Cross join creates complexity for user to understand output.
- This join is not usually used because it takes very long time to run and produces huge result set which may not be useful.

- Syntax (ANSI style):
- Select col1,col2,col3,…. From <table1> cross join <table2> ;

- Syntax (THETA style):
- Select col1,col2,col3,…. From <table1>,<table2>;

JOIN

- 4. SELF JOIN :
- A self join joins a table to itself. This join compares values in a single column of one table is called self join.
- For this, we have to write same table name two times after from clause, with different alias.
- By doing this, two copies of the same table is opened in memory, and joined using one or more related columns.