

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
- (columns1 datatype(size),
- columns2 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;

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.

Ex. Select *from student;

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.

Ex. Select id,name from student;

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.

Ex. Select *from student where id=111;

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.

Ex. Select id,name from student where name in ('syam');

CREATING A NEW TABLE FROM ANOTHER TABLE

- ▶ Syntax :
- ▶ Create table <new table name>
- ▶ 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.
- ▶ Ex. create table vijay as select roll,name from paras

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

- ▶ **alter command** is used for altering the table structure, such as, to add a column to existing table. to rename any existing column. to change datatype of any column or to modify its size.
- ▶ **Syntax :**
 - Alter table table_name ADD(column_name);
 - Ex. [1] alter table student add (mobile number(10))
 - [2] alter table student DROP COLUMN subject;

Modify (update)

- ▶ “**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.

update

- ▶ Syntax :
 - Update table_name SET columnsname where columename.

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

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.

Delete

► Syntax[1] :

Delete from <table name>;

Ex.: delete from student ;

Syntax[2]:

- Delete from table_name where record_name

Ex.: delete from student where roll=2

truncate

- ▶ Delete/Remove table all 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 describe**
 - 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

Operator's

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(અને):
 - Ex. select *from student where name='aaa' AND salary=2000
- ▶ OR(અથવા):
 - Ex. select *from student where id=2 OR salary=2000
- ▶ NOT:
 - Ex. select *from student where id NOT LIKE 2

In/not 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)`

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')

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

Bitween operator

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

Like operator

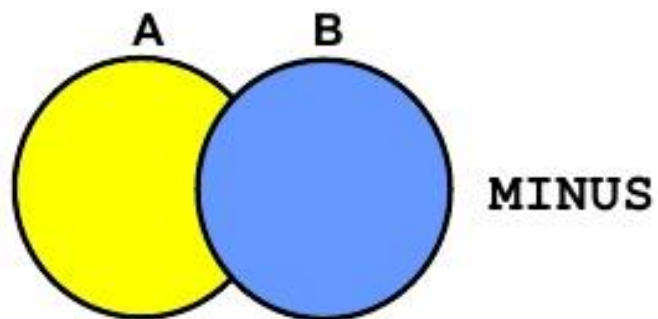
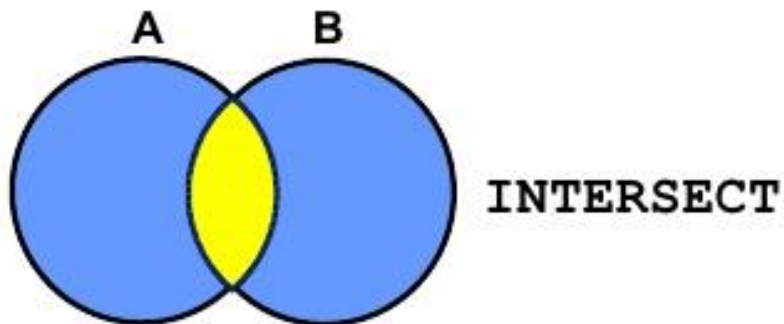
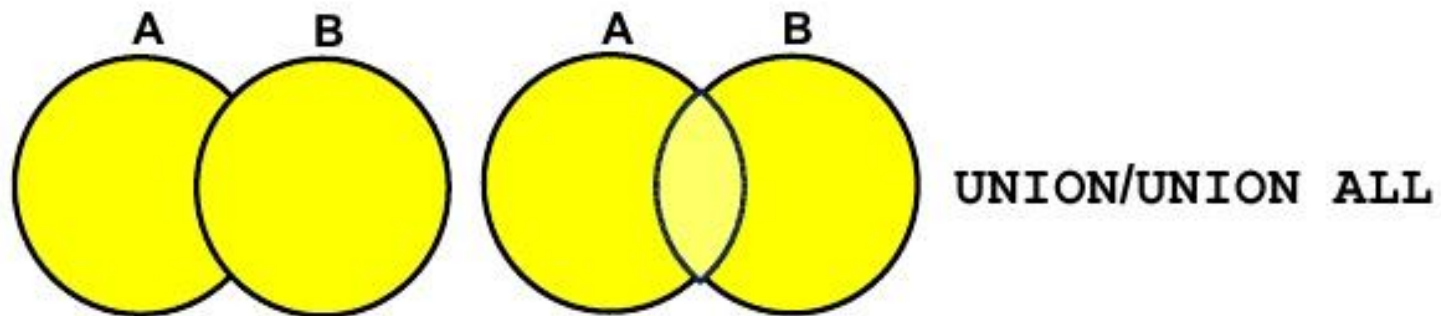
- ▶ In query 'pattern matching' can be done using % along with LIKE operator.
- ▶ Ex.
 - `select *from emp where job LIKE '%salesman'`
 - `select *from emp where deptno LIKE '30'`

SET operators

SQL supports three types of operators

- ▶ [1] UNION
- ▶ [2] MINUS
- ▶ [3] INTERSECT

Set Operators



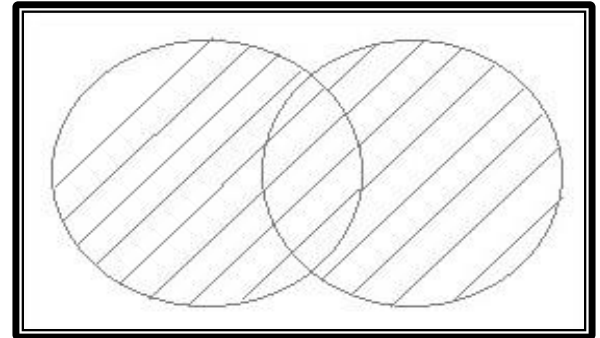
- ▶ create table emp1 (emp_id number (3),emp_name varchar2(10),dept_id number(3))
- ▶ insert into emp1 values(1,'ram',10)
- ▶ insert into emp1 values(2,'syam',20)
- ▶ insert into emp1 values(3,'kam',30)
- ▶ insert into emp1 values(4,'man',10)
- ▶ select *from emp1

- ▶ create table dept (dept_id number (3),dept_name varchar2(10),emp_id number(3))
- ▶ insert into dept values(1,'ram',10)
- ▶ insert into dept values(2,'syam',20)
- ▶ insert into dept values(3,'kam',30)
- ▶ insert into dept values(4,'man',10)
- ▶ select *from dept

[1]

union

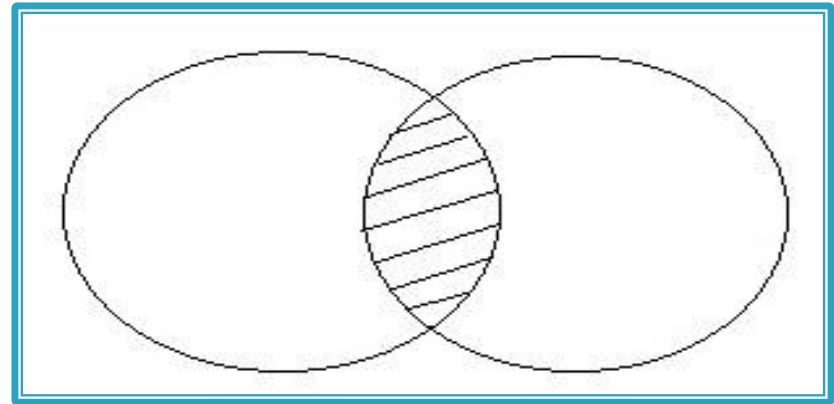
- ▶ **UNION** is used to combine the results of two or more **SELECT** statements.
- ▶ However it will eliminate duplicate rows from its result set.
- ▶ In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.



- ▶ **Ex.**
 - `Select dept_id from emp1 UNION select dept_id from dept`

[2] intersect

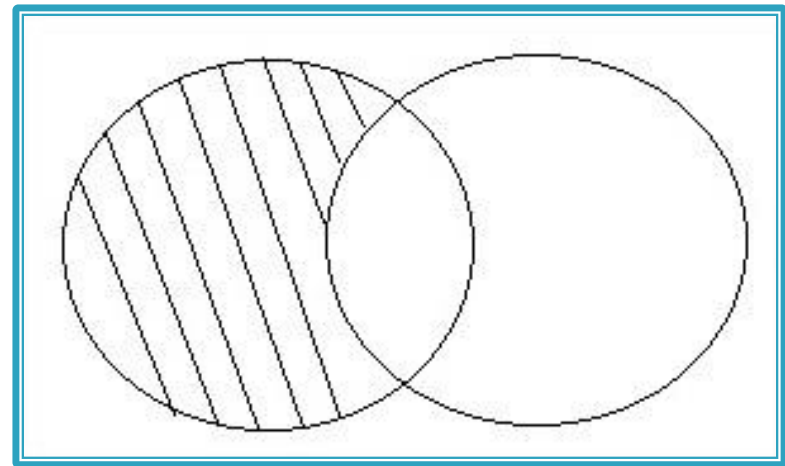
- ▶ Intersect operation is used to combine two SELECT statements,
- ▶ but it only returns the records which are common from both SELECT statements.
- ▶ In case of **Intersect** the number of columns and datatype must be same.



- ▶ **Ex.**
 - `Select emp_name from emp1 INTERSECT select dept_name from dept`

[3] minus

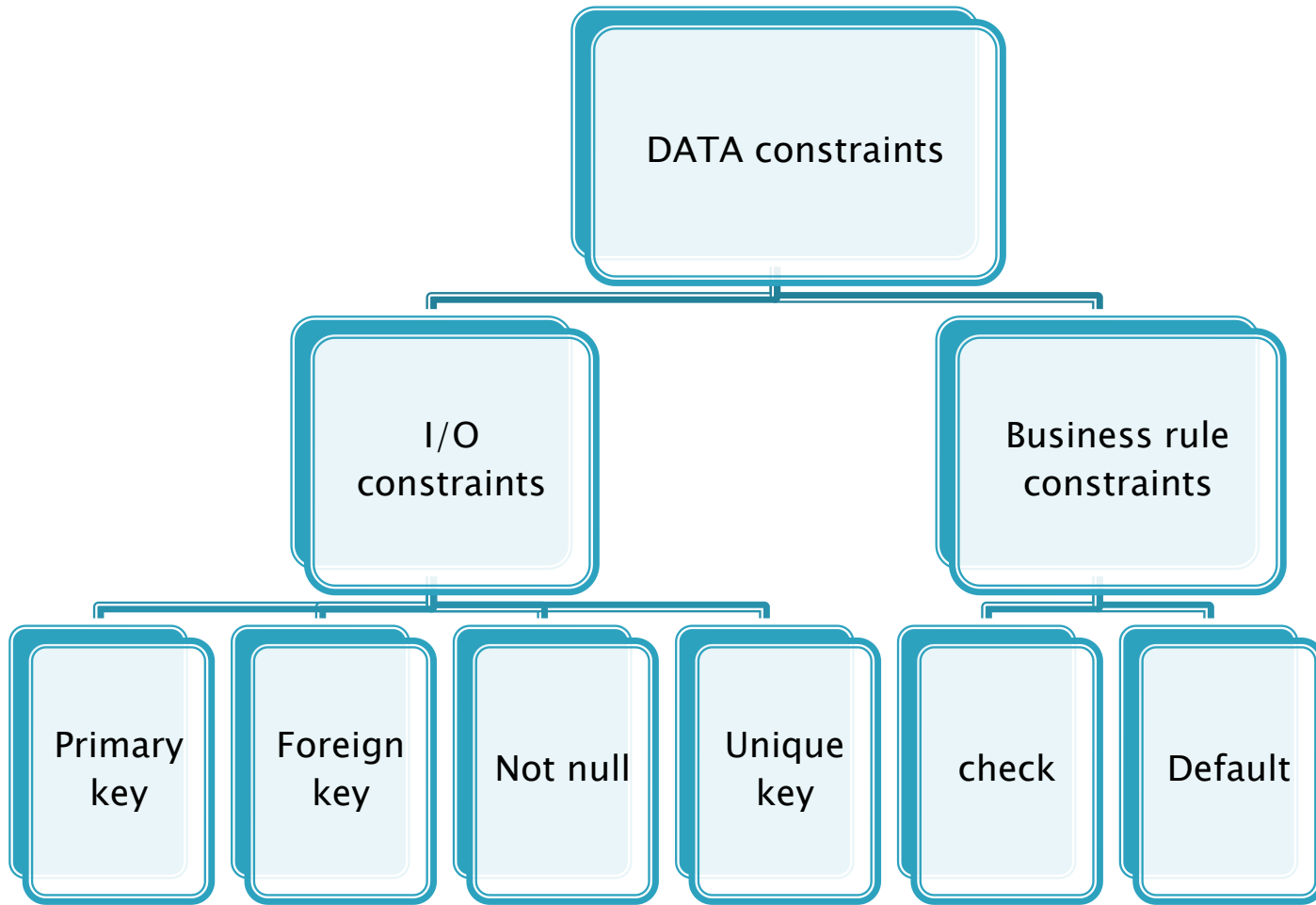
- ▶ The Minus operation combines results of two SELECT statements and
- ▶ return only those in the final result, which belongs to the first set of the result.



- ▶ Ex.
 - Select emp_id from emp1 MINUS select emp_id from dept

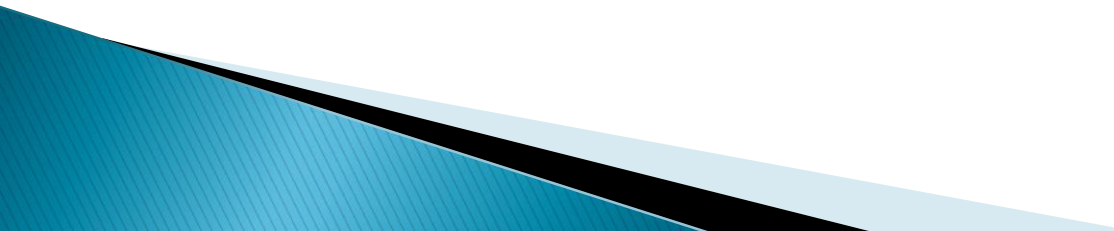
DATA constraints

- ▶ 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.



DATA CONSTRAINTS

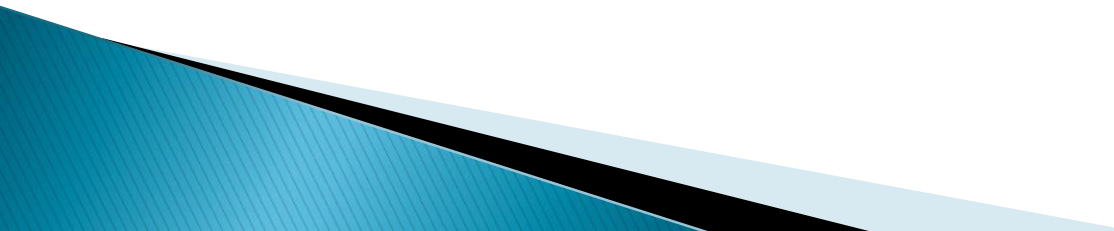
▶ Following are types of constraints:

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

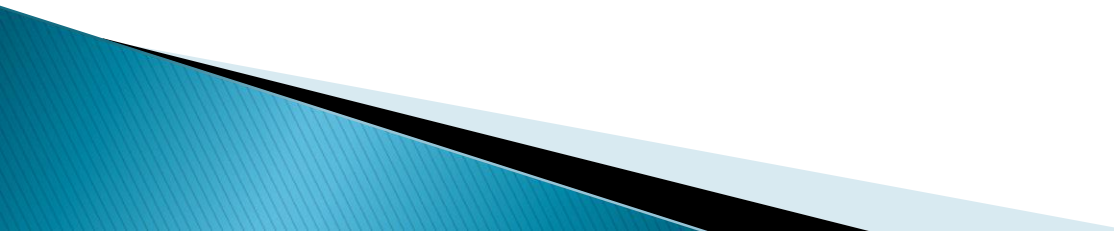
DATA CONSTRAINTS

- ▶ (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));

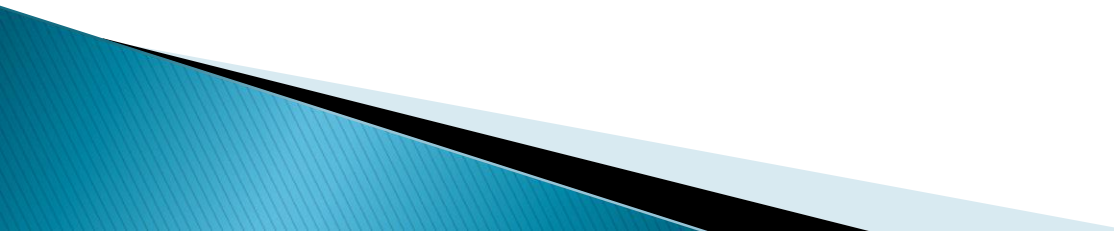
DATA CONSTRAINTS

- ▶ 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.
- 

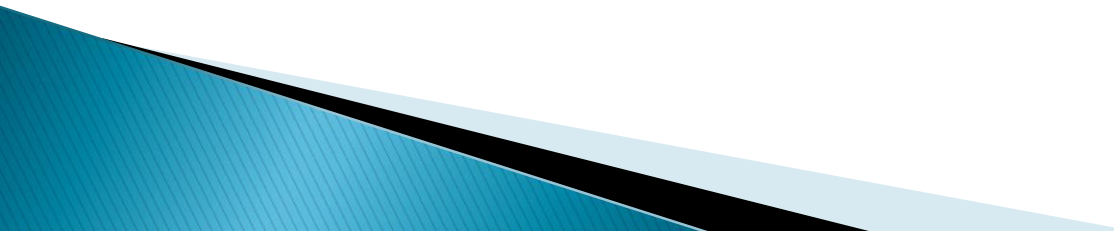
DATA CONSTRAINTS

- ▶ (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.
- 

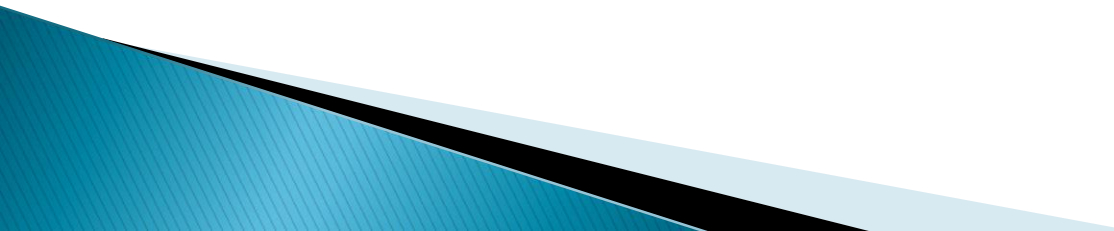
DATA CONSTRAINTS

- ▶ 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.
- 

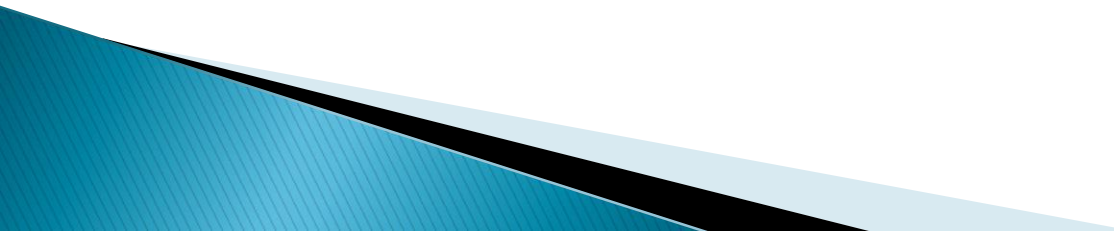
DATA CONSTRAINTS

- ▶ 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 corresponding 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 we enter 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';`

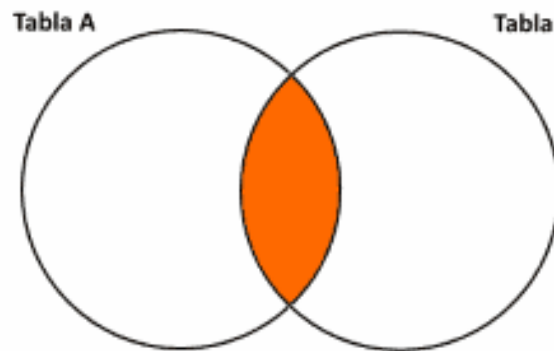
joins

joins

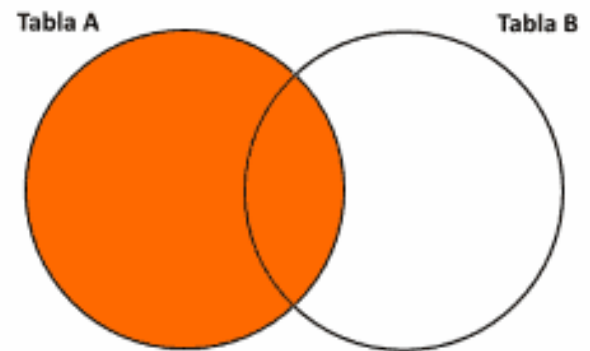
- ▶ To combine the data from two or more tables ,it is called a JOIN.
- ▶ Table are joined on columns that have the same data type and data width in the tables.
- ▶ In joins Rows in *one table can be joined to rows in another table* according to common values (that are relational values)existing in corresponding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

Types of joins

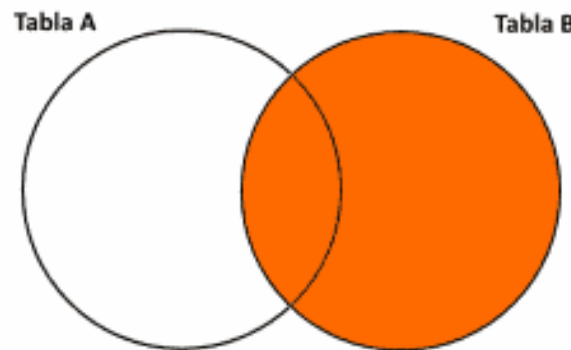
- ▶ [1] Inner join
- ▶ [2] Outer (Left , Right)
- ▶ [3] cross
- ▶ [4] full outer
- ▶ [5] Self



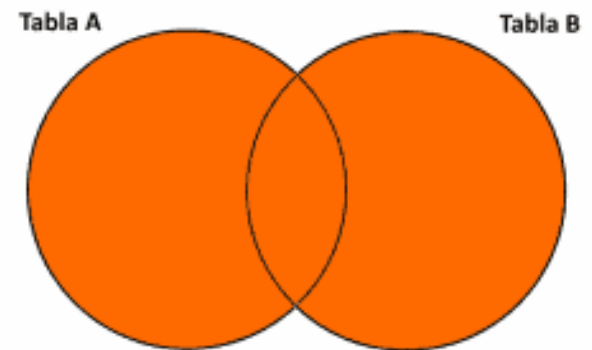
INNER JOIN



LEFT OUTER JOIN



RIGHT OUTER JOIN



FULL OUTER JOIN

- ▶ 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

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

[1] Inner join

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

EMPNO	ENAME	JOB	DNAME	LOC	DEPTNO
111	ram	analyst	mrk	hyd	10
222	syam	clark	acco	bgsr	20
333	jigar	manager	mrk	hyd	10

▶ 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>;

[2] Outer join

- ▶ An outer join allows you to join two tables and results even when the second table doesn't have any records corresponding with the first.
- ▶ **LEFT-OUTER Join**
 - The left outer join can be used to return all the rows from the first table even if there are no matches in the second table.
 - **Ex.** Select e.empno,e.ename,e.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	-	-

▶ (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>;

► 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 e.empno,e.ename,e.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


▶ (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>;

[3] Cross join

- ▶ This will gives the cross product.
- ▶ Ex.
 - [Ansi] Select empno,ename,job,dname,loc from emp cross join dept;
 - [Theta] Select empno,ename,job,dname,loc from emp ,dept;

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

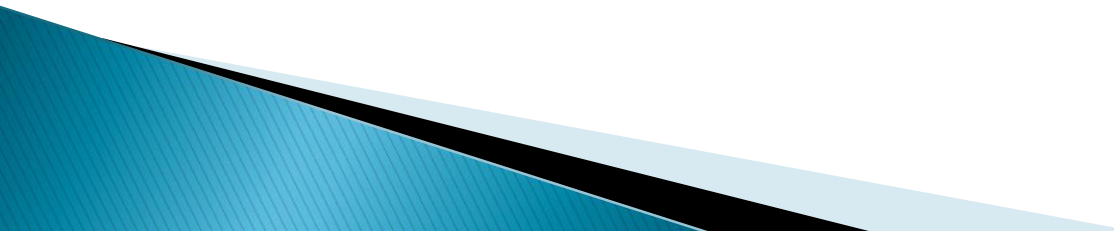
- ▶ 3. CROSS JOIN :
 - ▶ Cross join combines every row from left table with every row in right table.
 - ▶ Cross join creates complexity for user to understand output.
 - ▶ This join is not usually used because it takes very long time to run and produces huge result set which may not be useful.
 - ▶ Syntax (ANSI style):
 - ▶ `Select col1,col2,col3,... From <table1> cross join <table2> ;`
 - ▶ Syntax (THETA style):
 - ▶ `Select col1,col2,col3,... From <table1>,<table2>;`
- 

[4] Full outer join

- ▶ This will display the all matching records and the non-matching record from both tables.
- ▶ **Ex.**
 - Select e.empno,e.ename,e.job,d.dname,d.deptno from emp e,dept d where e.deptno(+) = d.deptno UNION Select e.empno,e.ename,e.job,d.dname,d.deptno from emp e,dept d where e.deptno = d.deptno (+)

EMPNO	ENAME	JOB	DNAME	DEPTNO
111	ram	analyst	mrk	10
222	syam	clark	acco	20
333	jigar	manager	mrk	10
444	kam	engineer	-	-
-	-	-	hr	30

5. 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.
- 

Function

BUILT-IN-FUNCTION

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

- ▶ `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;`

-:: 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) "square 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

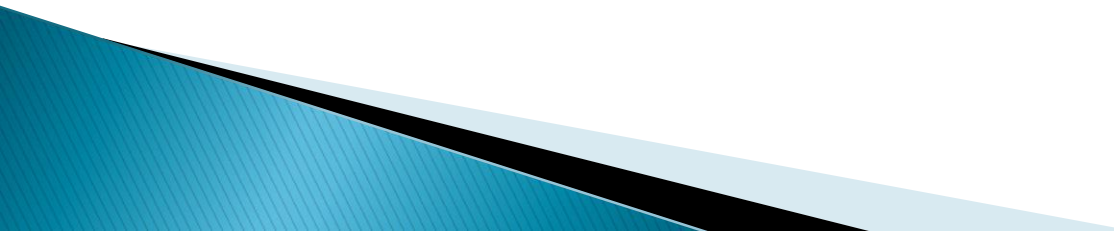
- ▶ **(9)GREATEST()**

- ▶ **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()

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

▶ (3)MAX()

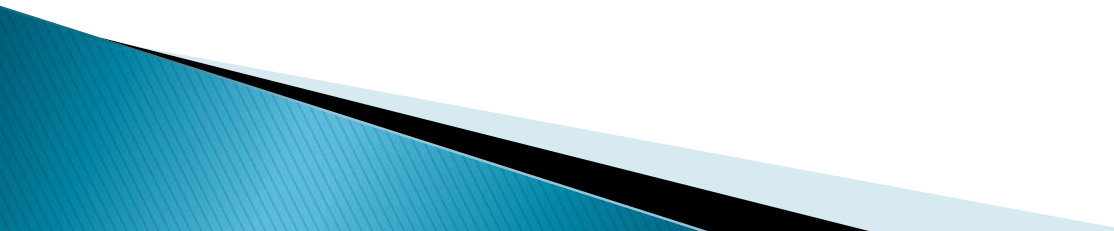
- ▶ select MAX(salary) from employee;
- ▶ o/p:999

▶ (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;
- ▶ O/p:07-AUG-15 03.01.27.541000 PM +05:30

General functions

- ▶ **(1) Translate ()**
- ▶ **Syntax:** Translate (st1, st2, st3)
- ▶ **Purpose:** Returns the main string st1 by replacing the chars specified in st2 with st3. It replaces single char at a time.
- ▶ **Example:** select translate ('12345','24','bb')
from dual;
- ▶ **Output:** Translate
- ▶ -----
- ▶ 1b3b5

- ▶ **(2) Replace ()**
- ▶ **Syntax:** Replace (st1, st2, st3)
- ▶ **Purpose:** Returns the main string st1 by replacing char or chars in a string with one or more chars.
- ▶ **Example:** select replace ('jack and jue', 'j', 'bl') from dual;
- ▶ **Output:** Replace
- ▶ -----
- ▶ black and blue

- ▶ **(4) Chr ()**
- ▶ **Syntax: Chr (number)**
- ▶ **Purpose:** Returns the main string st1 by replacing char or chars in a string with one or more chars.
- ▶ **Example:** select chr(65) || chr(66) || chr(67)
"character" from dual
- ▶ **Output:** cha
- ▶ -----
- ▶ ABC

- ▶ **(5) COALESCE()**

- ▶ **Syntax:** COALESCE(*val1*, *val2*, ..., *val_n*)

- ▶ **Purpose:** The COALESCE() function returns the first non-null value in a list.

- ▶ **Example:**

SELECT COALESCE(NULL, NULL, NULL, 'W3Schools.com', NULL, 'Example.com') from dual;

- ▶ **Output:**

- ▶ -----

- ▶ W3Schools.com

▶ (6) DECODE()

- ▶ **Syntax:** DECODE(col|expression, search1, result1 [, search2, result2,...,][, default])
- ▶ **Purpose:** Facilitates conditional inquiries by doing the work of a CASE or IF-THEN-ELSE statement.

▶ **Example:** SELECT DECODE(1,2, 'Equal', 'Not Equal')
FROM DUAL

▶ **Output:**

▶ -----
▶ Not Equal

▶ **It works like the following IF-THEN-ELSE statement:**

▶ IF 1 = 2 THEN
▶ RETURN 'Equal';
▶ ELSE
▶ RETURN 'Not Equal';
▶ END IF;

▶ (7) CASE WHEN()

▶ Syntax: CASE

```
WHEN condition1 THEN result1  
WHEN condition2 THEN result2  
WHEN conditionN THEN resultN  
ELSE result
```

END;

- ▶ **Purpose:** the CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.
- ▶ If there is no ELSE part and no conditions are true, it returns NULL.

Select statement with
group by, having, order by
clause

- ▶ create table emp (empno number(5),ename varchar2(10),job varchar2(10),mgr number(5),sal number(10),deptno number(3))
- ▶ insert into emp values(7369,'smith','clerk',7902,800,20)
- ▶ insert into emp values(7499,'allen','salesman',7898,1600,30)
- ▶ insert into emp values(7521,'ward','salesman',7898,1250,30)
- ▶ insert into emp values(7566,'jones','manager',7839,2975,30)
- ▶ insert into emp values(7654,'martin','salesman',7898,1250,30)
- ▶ insert into emp values(7698,'black','manager',7839,2850,30)
- ▶ insert into emp values(7788,'scott','analyst',7566,3000,20)
- ▶ insert into emp values(7839,'king','president',7902,5000,10)
- ▶ insert into emp values(7844,'turner','salesmen',7698,1500,30)
- ▶ insert into emp values(7876,'adams','clerk',7788,1100,20)
- ▶ select *from emp

Select statement

- ▶ This statement may be used to retrieve(modify) information already stored in the database.
- ▶ It is a retrieve data you can either select all the column values or name specific column in the select clause(graft) to retrieve data.
- ▶ Ex.
 - `select ename,sal*10 "income"from emp`
 - o/p:all record display only ename and sal*10

Group by clause

- ▶ The rows in a table can be divided into different groups to treat each group separately.
- ▶ The GROUP BY clause is used for grouping the data.
- ▶ Ex.
 - select deptno,max(sal) from emp group by deptno

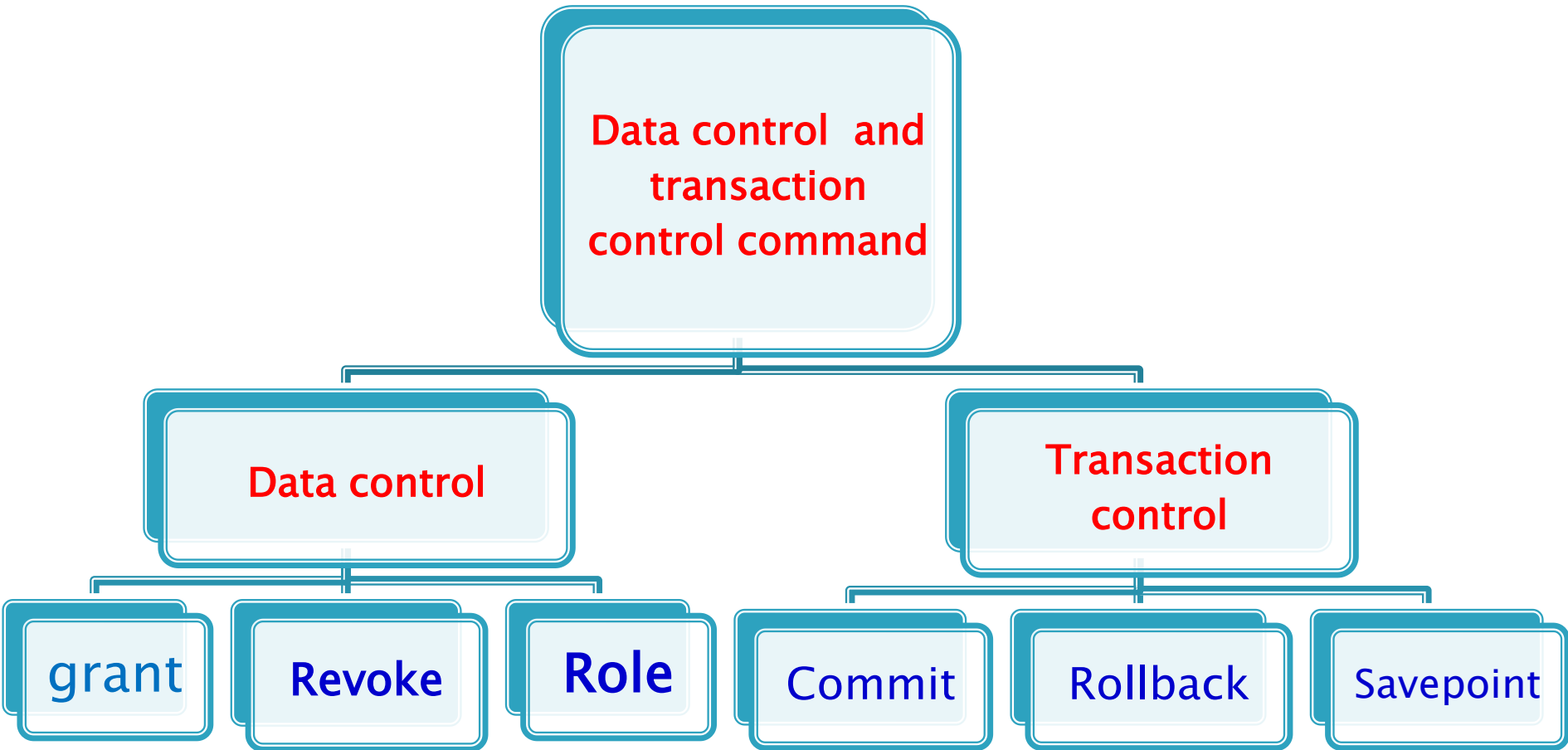
Order by clause

- ▶ Order by clause is useful to sort resulting rows in Ascending /Descending order at the time of displaying record.
- ▶ **Ex.1 select ename,sal,job from emp order by job**
- ▶ **Ex.2 select ename,sal,job from emp order by job desc**

Having clause

- ▶ The having clause can restrict group.
- ▶ The WHERE clause restrict rows, the group by clause group remaining rows, the group function works on each group and the having clause keeps the groups that match the group condition.
 - Ex. `select deptno,avg(sal) from emp group by deptno having avg(sal) >1000`

Data control and transaction control command



Privileges

Privileges := The “right” that allow the use of some or all of oracle’s resources on the server are called privileges.

- ▶ A **schema object privilege** is a privilege or right to perform a particular action on a specific schema object:
 - Table * View
 - Sequence * Procedure
 - Function * Package

Few CREATE system privileges are listed below:

System Privileges	Description
CREATE object	allows users to create the specified object in their own schema.
CREATE ANY object	allows users to create the specified object in any schema.

Few of the object privileges are listed below:

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
EXECUTE	allows user to execute a stored procedure or a function.

Create User

- ▶ The oracle system comes with two users already created, system and sys. You log onto the system user to create other users, since system has that privileges.
- ▶ Create user create a user account that lets you log onto the database with a certain set of privileges and strong settings. If you specify a password you must supply that password to logon.
- ▶ **Creating user & role:**
- ▶ **Syntax :**
 - create user username IDENTIFIED by password

Ex :

[1] CREATE USER ADMIN identified by KAMANI;

[2] Create role bca

**Data control
command**

grant

- ▶ **Grant Privileges on Table**
- ▶ You can grant users various privileges to tables. These privileges can be any combination of SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALTER, INDEX, or ALL.
- ▶ **The Syntax for the GRANT command is:**
- ▶ GRANT privilege_name
ON object_name
TO {user_name}
- ▶ ***privilege_name*** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- ▶ ***object_name*** is the name of an database object like TABLE, VIEW, and SEQUENCE.
- ▶ ***user_name*** is the name of the user to whom an access right is being granted.
- ▶ ***user_name*** is the name of the user to whom an access right is being granted.

▶ Ex.1 To give selected privileges(grant):

▶ GRANT SELECT, INSERT, UPDATE, DELETE ON system.emp TO admin

▶ **select** *from system.emp

▶ **insert** into system.emp values
(124,'aaa','professor',233,400,20)

▶ **update** system.emp set deptno=30 where
ename='kam'

▶ Ex.2 To give all privileges(grant):

▶ Grant all on emp to admin;

REVOKE

▶ Revoke Privileges

- The REVOKE command removes user access rights or privileges to the database objects.
 - Once you have granted EXECUTE privileges on a function or procedure, you may need to REVOKE these privileges from a user.
 - The owner of the object can take privileges once given 'back'. This is called revoking of privileges.
- ▶ It is command use of recover in data information to other user.

▶ Syntax

- Revoke <Object privilege> ON <Object name>
FROM <User name>

▶ Ex.1 To give selected privileges(revoke):

- REVOKE select,insert on system.emp from admin

▶ Ex.2 To give all privileges(revoke):

- REVOKE all on emp from admin

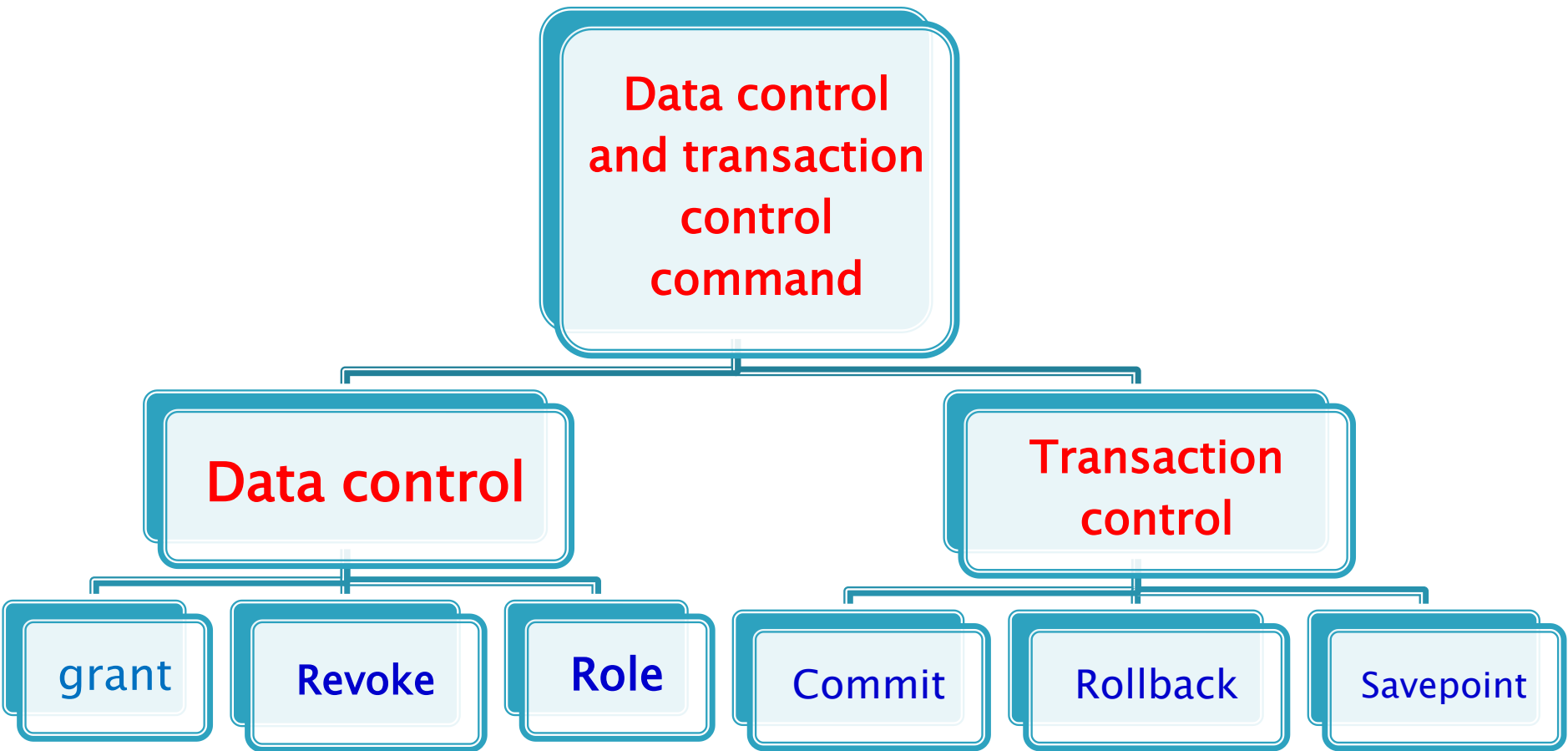
role

- ▶ A role is a set(group) of privileges that can be granted to users or to other roles.
- ▶ It is a used for privilege management.
- ▶ Roles are a collection of privileges or access rights.
- ▶ Any combination of system privileges ,object privileges ,and role privileges may be granted to a role.

Ex.

- Create role bca
- grant create table,create view to bca
- Grant bca to admin

transaction control command



What is a transaction ?

- ▶ A series of different operations performed on table data is known as TRANSACTION.
- ▶ All changes to data in a transaction are together or rollback together.
- ▶ A transaction will implicitly with an insert, update delete or select statement.
- ▶ Types of transaction command:
 - ▶ 1)commit
 - ▶ 2)rollback
 - ▶ 3)savepoint

Commit

- ▶ COMMIT command is used to permanently save any transaction into the database.
- ▶ A commit ends the current transaction and makes permanent any changes made during the transaction.
- ▶ Ex. :
 - Insert into student values(2,'aaa',10);
 - Commit;

Rollback

- ▶ A rollback exactly opposite of commit.
- ▶ Recover the data in database.
- ▶ Ex.
 - `Select *from student;`
 - `delete from student where id=2;`
 - `ROLLBACK;`
 - `select *from student;`

Savepoint

- ▶ Save point marks and saves the current point in the processing of a transaction.
- ▶ Always savepoint can be used with rollback command. [Note: don't work in express edition]
- ▶ **Ex.**
 - Insert into student values(1,'ramesh');
 - Savepoint first;
 - Insert into student values(2,'rakesh');
 - Savepoint second;
 - Insert into student values(3,'paresh');
 - Savepoint third;
 - Insert into student values(4,'mahesh');
 - Savepoint four;
 - Rollback to third
 - Select *from student; [up to return data before savepoint third]

Thank you
?