

Dr.M.Lovelin Ponn Felciah
Associate Professor
Bishop Heber College

Introduction to oracle:

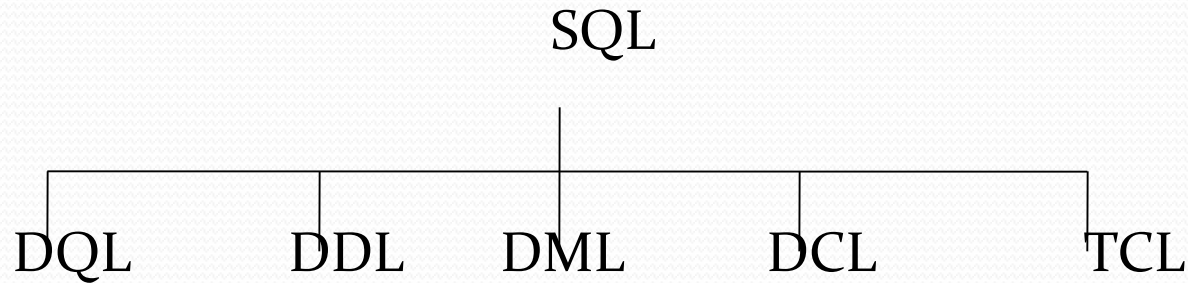
1. Oracle database server(DB)- storage, retrieval, manipulation of data.
2. Oracle developer suite-Used for developing GUI screens and reports.
3. Oracle Application server(AS)-All screens and reports developed using Developer suite are compiled and stored on AS.

Oracle naming conventions:

There are certain rules which are to be followed while giving name to tables or columns names.

- Names cannot be more than 30 characters in length.
- Names can contain letters(alphabets), digits and underscore(_).
- A name must start with a letter. It cannot start with any other character even underscore.
- Although Oracle allows \$ and # in names but it is advisable not to use them.

Classification of SQL commands:



DQL-Data Query Language

DDL- Data Definition Language

DML-Data Manipulation Language

DCL- Data Control Language

TCL- Transaction Control Language

Examples:

DQL-SELECT

DDL-CREATE,DROP,ALTER

DML-INSERT, UPDATE,DELETE,TRUNCATE

DCL-GRANT,REVOKE

TCL-COMMIT,ROLLBACK,SAVEPOINT

Data Query Language:

These commands are used to view records from tables onto computer monitor.

The actual data in the form of tables is stored onto the hard disk of Oracle Database Server.

DQL commands enables to view the information/records stored in database onto the monitor.

for example: SELECT.

Data Definition Language:

DDL statements are used to create tables, remove tables and make changes to the table structure.

Table structure refers to the number of columns and data types of columns and constraints which can be applied or revoked as desired.

for example: CREATE, DROP, ALTER.

Data Manipulation Language:

For adding new records, removing existing records or changing values of existing records DML statements is used, for example: INSERT, UPDATE, DELETE.

Data Control Language:

Oracle is a multi user database management system.

Each user has his own assigned area where the data is stored.

By default one user cannot view, modify or delete the information stored in another user's area.

Oracle has provision for allowing the owner to grant permissions on data owned by him to another user.

These commands control how and to what extent one user can view, modify and delete information in another user's area, for examples, GRANT, REVOKE.

Transaction Control Language:

Each operation which is done on database is called a Transaction.

Transaction can be addition of new record, modification in values of existing records or deletion of existing records.

Each transaction can be stored permanently or can be undone by using the transaction control statements.

for examples, COMMIT, ROLLBACK, SAVEPOINT.

Oracle Data Types:

DATATYPE-A DEFINITION:

Data type implies the set of numbers, characters, special character which can be input as a value for a particular column of the table. A column whose type has been defined as NUMBER will not accept any character other than numerals(0-9), a decimal and +/- sign.

CHAR:

valid characters : 0-9, a-z, A-Z, special characters like \$, %, &, etc.

Default size:1,min.len:1,max.len:1

CHAR(N):

valid characters : 0-9, a-z, A-Z, special characters like \$, %, &, etc.

Default size: N, min.len:1, max.len:2000

VARCHAR2(N):

valid characters : 0-9, a-z, A-Z, special characters like \$, %, &, etc.

Default size: N, min.len:1,max.len:4000

LONG:

similar to VARCHAR2(N) with maximum size of 2GB.

valid characters : 0-9, a-z, A-Z, special characters like \$, %, &, etc.

Default size : 2GB , min.len:1,max.len:2 GB

NUMBER(N):

Integer values with maximum N digits, e.g. NUMBER(7)

valid characters: 0-9, +,-. Default size:N,min.value:-9999999,max.val:+9999999

NUMBER(p, s):

This data type is used to store decimal numbers, for example, NUMBER(7,2).

p: Precision : Total number of digits including digits before decimal point.

s: Scale : Total number of digits after decimal point.

Decimal values with maximum p-s digits to the left of decimal point and s number of digits to the right of decimal point.

valid characters: 0-9, +,-.

NUMBER Data Type:

Just by specifying NUMBER, it can store large numbers.

Positive numbers from 1×10^{-130} to $9.99...9 \times 10^{125}$ with up to 38 significant digits.

Negative numbers from -1×10^{-130} to $9.99...99 \times 10^{125}$ with up to 38 significant digits.

By 38 significant digits we mean the number can have 38 9's before the decimal point.

RAW(SIZE) Data Type:

This is used to store binary data like images, sound, videos, graphs, etc.

Its variable length data type with maximum size of 2000 bytes.

LONG RAW:

It is also used to store binary data like images, sound, videos, graphs, etc.

Its variable length data type with maximum size of 2GB.

ORACLE OPERATOR

TYPES OF OPERSTOR

OPERATORS

Mathematical Operators

+, -, *, /

comparison/Relational Operators

=, <, >, >=, <=,

<> or !=

Logical/Boolean Operators

AND, OR,

NOT

Special Operators

IN, IS, LIKE,

BETWEEN

Mathematical Operators:

The mathematical operators are used for performing mathematical operation like addition, subtraction, multiplication, division, etc.

Assume an “emp” table which contains information about employees.

SQL> select * from emp;

EMPNO	ENAME	JOB	MGR	HIREDAT E	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-81	800	250	20
7499	ALLEN	SALESM AN	7698	20-FEB-81	1600	300	30
7369	WARD	SALESM AN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGE R	7839	02-APR-81	2975	250	20
7654	MARTIN	SALESM AN	7698	28-SEP-81	2000	250	30
7698	BLAKE	MANAGE R	7839	01-MAY-81	2850	250	30

- Display the sum of sal and commission as **TOTAL SALARY** for all employees

SQL> select empno, ename, *sal+comm* from emp;

EMPNO	ENAME	SAL+COMM
7369	SMITH	1050
7499	ALLEN	1900
7521	WARD	1750
7566	JONES	3225
7654	MARTIN	2250
7698	BLAKE	3100
9999	XEON	2300

- Display the column heading as **TOTAL** instead of **SAL+COMM**

SQL>select empno, ename, *sal+comm as total* from emp;

EMPNO	ENAME	TOTAL
7369	SMITH	1050
7499	ALLEN	1900
7521	WARD	1750
7566	JONES	3225
7654	MARTIN	2250
7698	BLAKE	3100
9999	XEON	2300

Find out what percent of salary is **commission** for all employees.

SQL>select empno, ename, sal, comm, **comm/sal*100 as percent** from emp;

EMPNO	ENAME	SAL	COMM	PERCENT
7369	SMITH	800	250	31.25
7499	ALLEN	1600	300	18.75
7521	WARD	1250	500	40
7566	JONES	2975	250	8.40336134
7654	MARTIN	2000	250	12.5
7698	BLAKE	2850	250	8.77192982
9999	XEON	2000	300	15

- Display the percent up to **2 places of decimal**.

SQL>select empno, ename, sal, comm, **round(comm/sal*100)** as percent
from emp;

EMPNO	ENAME	SAL	COMM	PERCENT
7369	SMITH	800	250	31.25
7499	ALLEN	1600	300	18.75
7521	WARD	1250	500	40
7566	JONES	2975	250	8.4
7654	MARTIN	2000	250	12.5
7698	BLAKE	2850	250	8.77
9999	XEON	2000	300	15

• Calculate **BONUS** as 10 percent of (salary + commission).

SQL>select empno, ename, sal, comm, **10/100*(sal + comm) as bonus** from emp;

EMPNO	ENAME	SAL	COMM	BONUS
7369	SMITH	800	250	105
7499	ALLEN	1600	300	190
7521	WARD	1250	500	175
7566	JONES	2975	250	322.5
7654	MARTIN	2000	250	225
7698	BLAKE	2850	250	310
9999	XEON	2000	300	230

COMPARISON OR RELATIONAL OPERATOR

- List employees of **DEPARTMENT 20** only.

SQL>select empno, ename, sal, comm, deptno, from emp **where deptno =20;**

EMPNO	ENAME	SAL	COMM	DEPTNO
7369	SMITH	800	250	20
7566	JONES	2975	250	20

- List only those employees whose salary is **MORE THAN 1000**.

SQL>select empno, ename, sal, comm, from emp **where sal>1000**;

EMPNO	ENAME	SAL	COMM
7499	ALLEN	1600	300
7521	WARD	1250	500
7566	JONES	2975	250
7654	MARTIN	2000	250
7698	BLAKE	2850	250
9999	XEON	2000	300

- List all employees who are not in department 20.

SQL>select empno, ename, sal, comm, deptno from emp **where**
deptno<>20;



(Not equal to

operator)

EMPNO	ENAME	SAL	COMM	DEPTNO
7499	ALLEN	1600	300	30
7521	WARD	1250	500	30
7654	MARTIN	2000	250	30
7698	BLAKE	2850	250	30
9999	XEON	2000	300	50

LOGICAL OPERATORS/BOOLEAN OPERATORS

- List of Boolean operators:

BOOLEAN OPERATOR	CONDITION 1	CONDITION 2	RESULT
AND	TRUE	TRUE	TRUE
	TRUE	FALSE	FALSE
	FALSE	TRUE	FALSE
	FALSE	FALSE	FALSE
OR	TRUE	TRUE	TRUE
	TRUE	FALSE	TRUE
	FALSE	TRUE	TRUE
	FALSE	FALSE	FALSE
NOT	TRUE	-	FALSE
	FALSE	-	TRUE

AND OPERATOR:

- List employees of department 30 who have salary more than 1500.

**SQL>select empno, ename, sal, comm, deptno from emp where deptno=30
and sal>1500;**

EMPNO	ENAME	SAL	COMM	DEPTNO
7499	ALLEN	1600	300	30
7654	MARTIN	2000	250	30
7698	BLAKE	2850	250	30

- List employees whose salary ranges between 1000 and 2000.

SQL>select empno, ename, sal, comm, deptno from emp **where sal>=1000**
and sal<=2000;

EMPNO	ENAME	SAL	COMM	DEPTNO
7499	ALLEN	1600	300	30
7521	WARD	1250	500	30
7654	MARTIN	2000	250	30
9999	XEON	2000	300	50

OR OPERATOR:

**SQL>select empno, ename, sal, comm, deptno from emp
where sal>=1000 OR COMM>300;**

EMPNO	ENAME	SAL	COMM	DEPTNO
7521	WARD	1250	500	30
7566	JONES	2975	250	20
7698	BLAKE	2850	250	30

AND ,OR OPERATORS:

- List employees who are in dept 20 and salary>1000 or any other department with salary>2000.

SQL>select empno, ename, sal, comm, deptno from emp **where deptno=20 and sal>2000 or deptno!=20 and sal>2000;**

EMPNO	ENAME	SAL	COMM	DEPTNO
7566	JONES	2975	250	20
7698	BLAKE	2850	250	30

Not Operator:

- List employees who are not in dept 10.

SQL>select * from emp **where not deptno=10;**

EMPNO	ENAME	JOB	MGR	HIREDAT E	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-81	800	-	20
7499	ALLEN	SALESM AN	7698	20-FEB-81	1600	300	30
7566	JONES	MANAGE R	7839	02-APR-81	2975	300	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20

SPECIAL OPERATORS:

IN:

- List employees of dept 10 and 30 only.

SQL>select * from emp *where deptno=10 or deptno;*

SQL>select * from emp *where deptno IN (10,30);*

EMPNO	ENAME	JOB	MGR	HIREDAT E	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7839	KING	PRESIDENT	-	17-NOV-81	5000	-	10
7934	MILLER	CLERK	7782	23-JAN-82	1300	400	10

NOT Boolean Operator:

SQL>select * from emp *where deptno not in(10,30);*

EMPNO	ENAME	JOB	MGR	HIREDAT E	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	-	20
7566	JONES	MANAGE R	7839	02-APR-81	2975	300	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20

BETWEEN:

- List of employees whose salary is between 2000 and 5000.

SQL>select * from emp *where sal>=2000 and sal<=5000;*

SQL>select * from emp *where sal BETWEEN 2000 and 5000;*

LIKE OPERATOR:

- List employees whose name starts with 'A'.

SQL>select * from emp *where ename like 'A%';*

EMPNO	ENAME	JOB	MGR	HIREDAT E	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	12-JAN-83	1100	-	20
7499	ALLEN	SALESM AN	7698	20-FEB-81	1600	300	30

SQL>select * from emp *where ename like 'D%';*

EMPNO	ENAME	JOB	MGR	HIREDAT E	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	12-JAN-83	1100	-	20

IS Operator:

Before understanding the IS operator we need to understand a concept called NULL value.

SQL>select empno,ename,sal from emp where sal IS null;

NULL value:UNKNOWN values are called NULL value in Oracle. NULL is not blank spaces or zero.

SQL>insert into emp values (8000, 'RAHUL', 'MANAGER', 7839,'11-sep-97',null,null,20);

SQL>select * from emp;

EMPNO	ENAME	JOB	MGR	HIREDAT E	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-81	800	250	20
8000	RAHUL	MANAGE R	7839	11-SEP-97			20
7369	WARD	SALESM AN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGE R	7839	02-APR-81	2975	250	20
7654	MARTIN	SALESM	7608	28-SEP-81	2000	250	30

BUILT IN FUNCTION:

GROUPING BY HAVING clause:

GROUP BY clause forms groups on the specified columns. Filter of these group depends on some conditions which can be achieved by using GROUP BY HAVING clause.

- Display department wise total salary such that only those department are shown to which company is paying total salary of more than 9000.

SQL>select deptno, sum(sal) from emp group by deptno *having*
sum(sal)>9000 order by deptno;

DEPTNO	SUM(SAL)
20	10875
30	9400

- List the departments with more than three employees.

SQL>select deptno, count(empno) empcount from emp group by deptno
having count(empno)>3 order by deptno;

DEPTNO	EMPCOUNT
20	5
30	6

GROUPING on multiple columns:

- Display the amount of salary being paid jobwise for each department.

SQL>select deptno, job from emp *group by deptno*, job order by deptno,
job;

DEPTNO	JOB
20	CLERK
30	SALESMAN
30	SALESMAN
20	MANAGER
30	SALESMAN
30	MANAGER

SORTING

Introduction :

sorting refers to arranging records in a specified sequence may be alphabetically. This sorting can be achieved by the “ORDER BY” clause of select statement.

Sorting on multiple columns:

SQL>select * from stu *order by college asc, marks desc;*

COLLEGE	ROLLNO	NAME	MARKS
ADC	5	ASHOK	50
BHC	4	BALA	40
CEC	3	CALAB	30
DHC	2	DEEPAK	20
ESC	1	EDGE	10

SQL>select * from stu *order by marks;*

SQL>select * from stu *order by marks desc;*

SORTING ON A COLUMN WITH NULL VALUES:

NULL indicates an unknown value and therefore records with NULL values are listed at the end while sorting in ascending order.

SQL> insert into stu values('DHC', 5, NULL,50);

COLLEGE	ROLLNO	NAME	MARKS
DHC	5	-	50
CEC	4	BALA	40
ESC	3	CALAB	30
BHC	2	DEEPAK	20
ADC	1	EDGE	10

COMBINING **WHERE** AND ORDER BY CLAUSES:

SQL is a highly –structured language and requires the WHERE clause to appear before the ORDER BY clause.

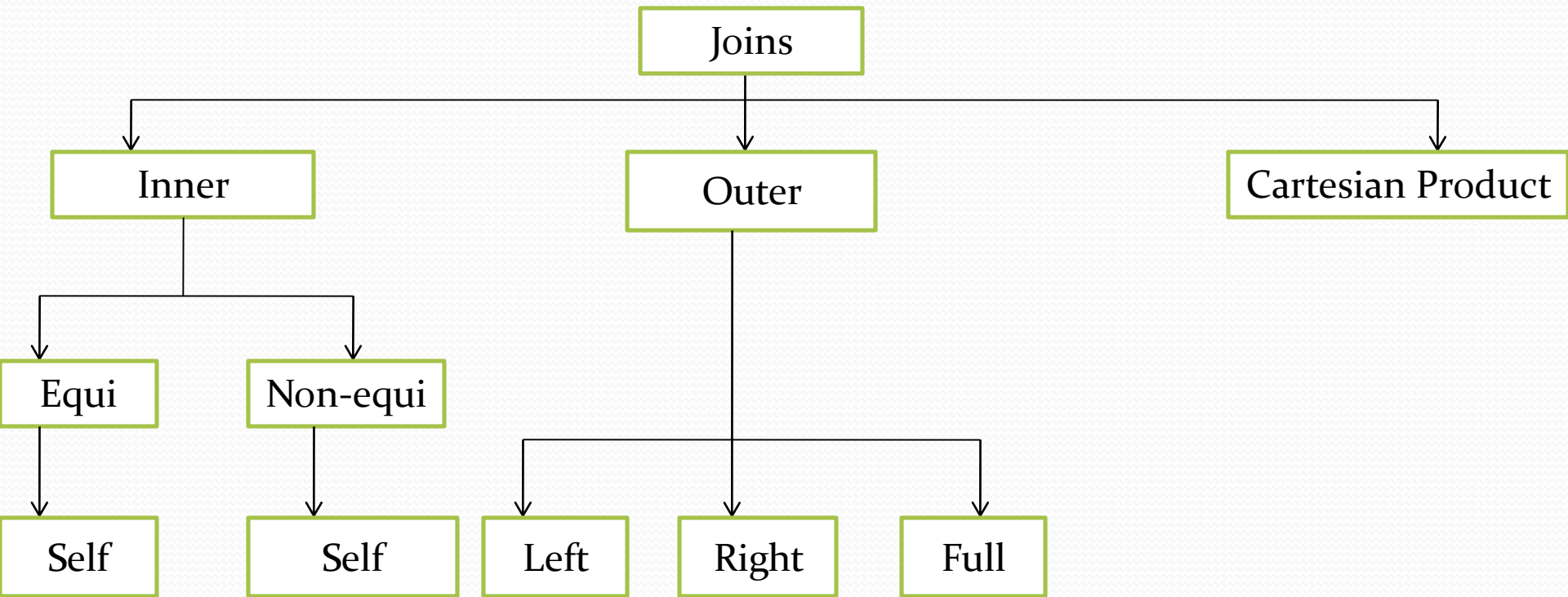
SQL>select * from stu *where college='DHC' order by marks desc;*

COLLEGE	ROLLNO	NAME	MARKS
DHC	5	ASHOK	-
DHC	4	BALA	-
DHC	3	CALAB	30
DHC	2	-	20
DHC	1	EDGE	10

JOINS:

The concept called “join” is used in order to achieve the desired result.

CLASSIFICATION OF JOINS:



INNER JOINS:


Joins which are based on the selection of matching records between tables called an *Inner Join*. Inner joins can be further categorized as Equi and Non-Equi.

(i). Inner equi join

when the two tables are joined using EQUAL TO operator then that join is called *Inner Equi Join*.

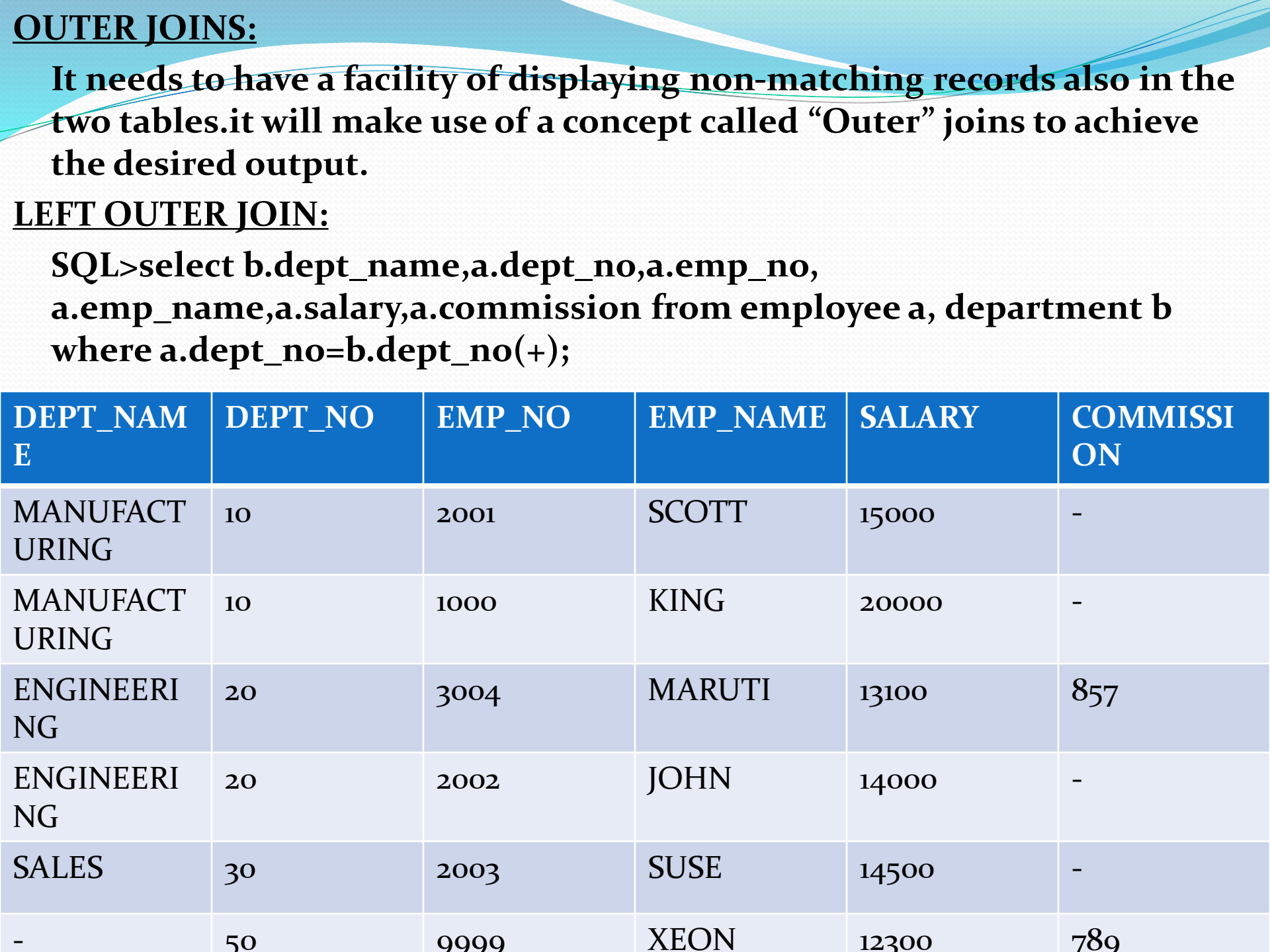
(ii). Inner Non-equi join

when the two tables are joined using any comparison operator (<, >, <=, >=, !=) other than EQUAL TO operator then that join is called *Inner Non-Equi Join*.



```
SQL>select department.dept_name, employee.emp_no,  
employee.emp_name,  
employee.emp_salary, employee.commission from  
employee,department where employee.dept_no=department.dept_no;
```

DEPT_NAME	EMP_NO	EMP_NAME	SALARY	COMMISSION
MANUFACTURING	1000	KING	20000	-
MANUFACTURING	2001	SCOTT	15000	-
ENGINEERING	2002	JOHN	14000	-
SALES	2003	SUSE	14500	-
ENGINEERING	3004	MARUTI	13100	857



RIGHT OUTER JOIN:

```
SQL>select b.dept_name,a.dept_no,a.emp_no,  
a.emp_name,a.salary,a.commission from employee a, department b  
where a.dept_no(+)=b.dept_no;
```

DEPT_NAME	DEPT_NO	EMP_NO	EMP_NAME	SALARY	COMMISSION
MANUFACTURING	10	1000	KING	20000	-
MANUFACTURING	10	2001	SCOTT	15000	-
ENGINEERING	20	2002	JOHN	14000	857
SALES	20	2003	SUSE	14500	-
ENGINEERING	30	3004	MARUTI	13100	-
FINANCE	-	-	-	-	-

FULL OUTER JOIN:

```
SQL>select b.dept_name,a.dept_no,a.emp_no,  
a.emp_name,a.salary,a.commission from employee a, department b  
where a.dept_no(+)=b.dept_no(+);
```

 ORA-01468: a predicate may reference only one outer-joined table

CARTESIAN PRODUCT:-

A join in which each record of table 1 gets associated with each and every record of table 2 is called a *Cartesian Product*.

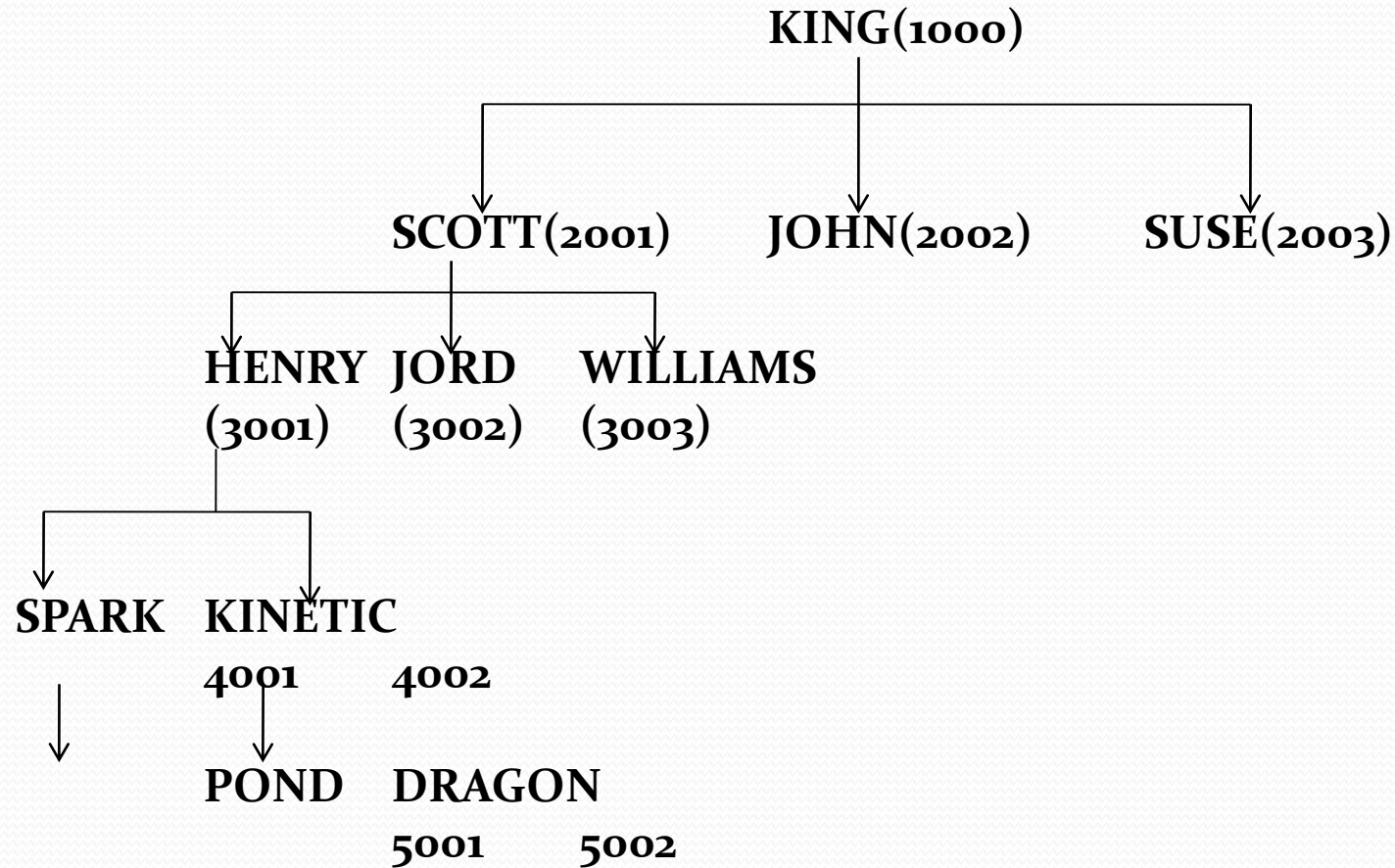
```
SQL>select b.dept_name,a.dept_no,a.emp_no,  
a.emp_name,a.salary,a.commission from employee a, department b  
order by a.emp_no;
```

DEPT_NAME	DEPT_NO	EMP_NO	EMP_NAME	SALARY	COMMISSION
FINANCE	10	1000	KING	20000	-
SALES	10	1000	KING	20000	-
ENGINEERING	10	1000	KING	20000	-
MANUFACTURING	10	1000	KING	20000	-
FINANCE	10	2001	SCOTT	15000	-
SALES	10	2001	SCOTT	15000	-
MANUFACTURING	10	2001	SCOTT	15000	-
ENGINEERING	10	2001	SCOTT	15000	-
.					
.					
.					

SELF-JOINS:

When a table is joined with its own then it is called a “self-join”.

Hierarchy of employees:



SQL>select x.emp_no,x.emp_name
 “Emp.Name”,x.salary,x.commission,x.senior_emp_no,y.emp_name”Sr.Na
 me” from employee x, employee y where x.senior_emp_no=y.emp_no
 order by x.emp_no;

EMP_NO	Emp.Name	SALARY	COMMISSI ON	SENIOR_EM P_NO	Senior.Nam e
2001	SCOTT	15000	-	1000	KING
2001	JOHN	14000	-	1000	KING
2003	SUSE	14500	-	1000	KING
3001	HENRY	13600	1000	2001	SCOTT
3002	JORD	12000	950	2001	SCOTT
3003	WILLIAMS	12100	850	2001	SCOTT
3004	MARUTI	13100	857	2002	JOHN
3005	SANTRO	12200	800	2002	JOHN
3006	PIAGGIO	10000	700	2003	SUSE
3006	TANGO	11000	760	2003	SUSE
4001	SPARK	10000	600	3001	HENRY
4002	KINETIC	11000	700	3001	HENRY
4003	JONATHAN	8500	850	3002	JORD

SQL>select x.emp_no,x.emp_name
 “Emp.Name”,x.salary,x.commission,x.senior_emp_no,y.emp_name”Sr.Na
 me” from employee x, employee y where x.senior_emp_no=**y.emp_no (+)**
 order by x.emp_no;

EMP_NO	Emp.Name	SALARY	COMMISSI ON	SENIOR_EM P_NO	Senior.Nam e
1000	KING	20000	-	-	-
2001	SCOTT	15000	-	1000	KING
2002	JOHN	14000	-	1000	KING
2003	SUSE	14500	-	1000	KING
3001	HENRY	13600	1000	2001	SCOTT
3002	JORD	12000	950	2001	SCOTT
3003	WILLIAMS	12100	850	2001	SCOTT
3004	MARUTI	13100	857	2002	JOHN
3005	SANTRO	12200	800	2002	JOHN
3006	PIAGGIO	10000	700	2003	SUSE

ADVANCED QUERIES USING SPECIAL OPERATORS:

Nested sub-queries:

IN OPERATOR:

The IN operator checks for the equality of set of values.

- Display college, rollno and name of the students for which marks have been entered in details table.

SQL>select * from stud where *rollno IN(select rollno from stud1);*

COLLEGE	ROLLNO	NAME
GEC	1	SATISH
GEC	2	RASHMI
RKDF	3	RISHI

SQL>select * from stud where rollno(1,2,3);

NOT IN OPERATOR:

NOT is a negation operator which can be used in conjunction with the **IN** operator for reversing the desired condition.

- Display college, rollno and name of the student for which marks have not been entered into details table.

SQL>select * from stud where *rollno NOT IN(select rollno from stud1);*

COLLEGE	ROLLNO	NAME
RKDF	4	ANIL

EXISTS:

An EXISTS condition tests for the existence of rows in a sub-query.

- Display college, rollno and name of those students for which marks have been entered in details tables.

SQL>select * from stud a where **EXISTS** (select * from stud1 b where a.rollno = b.rollno);

COLLEGE	ROLLNO	NAME
GEC	1	SATISH
GEC	2	RASHMI
RKDF	3	RISHI

Differences Between IN and EXISTS Operators:

1. The working of EXISTS and IN operators is slightly different with the same result.
2. The EXISTS operator only checks for the existence of matching records in details table whereas the IN operator checks whether the returned rollnos in sub-query are in the master table.
3. secondly, while using the EXISTS operator, we need to equi-join tables which is not required in the IN operator.
4. Thirdly, in the EXISTS operator for each record found in the main query, Oracle searches for the matching records in details table(sub-query). The main query executes first and for each record found, the corresponding records are searched in details table.
5. As the EXISTS operator only checks for the existence of records in the sub-query for each record found in the master table, therefore, the sub-query can use any value in the SELECT clause.

```
SQL>select * from stud a where EXISTS (select * from stud1 b where a.rollno =  
b.rollno);
```

NOT EXISTS:

- Display college, rollno and name of those students for which marks have not been entered in details tables.

SQL>select * from stud a where **NOT EXISTS** (select * from stud1 b where a.rollno = b.rollno);

YEAR	COLLEGE	ROLLNO	NAME
2006-2007	GEC	1	SATISH
2006-2007	GEC	2	RASHMI
2007-2008	GEC	1	ANITHA
2007-2008	GEC	2	DEEPA
2007-2008	GEC	3	JAYA
2006-2007	RKDF	3	RISHI
2006-2007	RKDF	4	ANIL

INDEXING :

Index is an object which can be defined as the ordered list of values of a column or combination of columns used for faster searching and sorting of data.

CREATING INDEX:

- To speed up the retrieval of records for college columns we will create an index.

```
SQL> create index ind_student_college on  
student(college);  
Index created.
```

REMOVING INDEX:

- To remove the index, the DDL statement DROP INDEX can be used .

```
SQL> drop index  
ind_student_college;  
Index dropped.
```

CREATING INDEX ON MULTIPLE COLUMNS:

- Index object can be created on multiple columns.

```
SQL> create index ind_student_college on  
student(college,rollno);  
Index created.
```

VIEWS:

View – Creating And Accessing :

- A view is a pre-defined query on one or more tables. s.

COLLEGE	ROLLNO	NAME	MARKS
GEC	1	SATISH	25
GEC	2	NILESH	23
OIST	3	RASHMI	21
OIST	4	PIYUSH	13
GEC	5	RUCHI	30

SQL> create view vw_student as select rollno , marks from student;

View created.

SQL> desc vw_student;

NAME	NULL?	TYPE
ROLLNO	NOT NULL	NUMBER(4)
MARKS	NUMBER(3)	

SQL> select * from vw_student;

ROLLNO	MARKS
1	25
2	23
3	21
4	13
5	30

CLASSIFICATION OF VIEWS:

Views can be classified as updateable views and non-updateable views.

Updateable Views:

- **Updateable** means to say that one can insert ,update, and delete records from view. Actually all the DML operations are performed on the base table.
- **View with the following characteristics is called an updateable view.**
 - 1. It is created from a single table.**
 - 2. It includes all the PRIMARY KEYS and NOT NULL columns of base table.**
 - 3. Aggregate functions like SUM, AVG, MAX have not been used.**
 - 4. It should not have DISTINCT, GROUP BY, HAVING clauses.**
 - 5. It must not use constants, strings or value expressions like salary * 2.**
 - 6. It must not contain function calls(e.g. RPAD,SUBSTR, etc.).**
 - 7. If a view is defined from an other view then that view must also be updateable.**

Non – updateable Views:

- **Non – updateable means - cannot insert, update, delete records from that view.**
- **View with the following characteristics is called a Non – updateable view.**
 1. **It is created from more than one table.**
 2. **It has DISTINCT, GROUP BY, HAVING clause. Even if view is derived from single table but contains any of base clauses then it is not updateable.**
 3. **It does not include all the PRIMARY KEYS and NOT NULL columns of base tables.**