DATABASE MANAGEMENT SYSTEMS LAB MANUAL(R15)

A.Y.:2019 – 2020

Department of Information Technology

Name of the Faculty: Mr. R Prasadu
Dr. M Rekha Sundari
Dr. K Sharada

Overview of SQL DDL, DML and DCL Commands.

DDL is Data Definition Language statements. Some examples:
CREATE - to create objects in the database
ALTER - alters the structure of the database
DROP - delete objects from the database
TRUNCATE - remove all records from a table, including all spaces allocated for the records ar
removed
COMMENT - add comments to the data dictionary
GRANT - gives user's access privileges to database
REVOKE - withdraw access privileges given with the GRANT command
DML is Data Manipulation Language statements. Some examples:
SELECT - retrieve data from the a database
INSERT - insert data into a table
UPDATE - updates existing data within a table
DELETE - deletes all records from a table, the space for the records remain
CALL - call a PL/SQL or Java subprogram
EXPLAIN PLAN - explain access path to data
LOCK TABLE - control concurrency
DCL is Data Control Language statements. Some examples:
COMMIT - save work done
SAVEPOINT - identify a point in a transaction to which you can later roll back
ROLLBACK - restore database to original since the last COMMIT
SET TRANSACTION - Change transaction options like what rollback segment to use

Basic SQL DDL Commands.

To practice basic SQL DDL Commands such as CREATE, DROP, etc.

1. SQL - CREATE TABLE

Syntax: CREATE TABLE tablename (column_name data_ type constraints, …)

Example:

INPUT:

SQL> CREATE TABLE Emp (EmpNo short CONSTRAINT PKey PRIMARY KEY,

EName VarChar(15), Job Char(10) CONSTRAINT Unik1 UNIQUE,

Mgr short CONSTRAINT FKey1 REFERENCES EMP (EmpNo),

Hiredate Date, DeptNo short CONSTRAINT FKey2 REFERENCES DEPT(DeptNo));

RESULT: Table created.

SQL>Create table prog20 (pname varchar2(20) not null), doj date not null,dob date not null, sex varchar(1) not null, prof1 varchar(20),prof2 varchar(20),salary number(7,2) not null);

RESULT:

Table created.

SQL>desc prog20;

Name	Null?	Type
PNAME	NOT NULL	VARCHAR2(20)
DOJ	NOT NULL	DATE
DOB	NOT NULL	DATE
SEX	NOT NULL	VARCHAR2(1)
PROF1		VARCHAR2(20)
PROF2		VARCHAR2(20)
SALARY		NOT NULL NUMBER(7,2)

2. SQL - ALTER TABLE

INPUT:

SQL>ALTER TABLE EMP ADD CONSTRAINT Pkey1 PRIMARY KEY (EmpNo);

RESULT: Table Altered.

Similarly, ALTER TABLE EMP DROP CONSTRAINT Pkey1;

3. SQL - DROP TABLE

– Deletes table structure – Cannot be recovered – Use with caution

INPUT:

SQL> DROP TABLE EMP; Here EMP is table name

RESULT: Table Dropped.

4**. TRUNCATE** TRUNCATE TABLE <TABLE NAME>;

Basic SQL DML Commands.

To practice basic SQL DML Commands such as INSERT, DELETE, etc.

1. SQL - INSERT INTO

Syntax: INSERT INTO tablename VALUES (value list)

- Single-row insert

INSERT INTO S VALUES('S3','SUP3','BLORE',10)

- Inserting one row, many columns at a time

INSERT INTO S (SNO, SNAME) VALUES ('S1', 'Smith');S1' Smith'

- Inserting many rows, all/some columns at a time.

INSERT INTO NEW_SUPPLIER (SNO, SNAME)

SELECT SNO, SNAME FROM S

WHERE CITY IN ('BLORE','MADRAS')

Other Examples:

INPUT:

SQL>Insert into prog values (‘kkk’,’05-may-56’);

RESULT: 1 row created.

INPUT:

SQL>Insert into prog20 values(‘Hema’,’25-sept-01’28-jan-85’,’f’,’c’,’c++’,’25000’);

RESULT: 1 row created.

INPUT:

SQL>Insert into prog values(‘&pname’,’&doj’);

SQL> Insert into prog values('&pname','&doj');

Enter value for pname: ravi

Enter value for doj: 15-june-81

RESULT:

old 1: Insert into prog values('&pname','&doj')

new 1: Insert into prog values('ravi','15-june-81')

1 row created.

2. SQL - UPDATE

Syntax: UPDATE tablename SET column_name =value [WHERE condition]

Examples:

UPDATE S SET CITY = ‘KANPUR’ WHERE SNO=‘S1’

UPDATE EMP SET SAL = 1.10 * SAL

SQL> update emp set sal=20000 where empno=7369;

1 row updated.

3. SQL - DELETE FROM

Syntax: DELETE FROM tablename WHERE condition

Examples:

DELETE FROM SP WHERE PNO= 'P1'

DELETE FROM SP

INPUT:

SQL>Delete from emp where empno=7369;

RESULT: 1 row deleted.

Basic SQL DCL Commands.

To practice basic SQL DCL Commands such as COMMIT, ROLLBACK etc.

1. COMMIT

Save changes (transactional).

Syntax:

COMMIT [WORK] [COMMENT '*comment_text*']

COMMIT [WORK] [FORCE '*force_text*' [,*int*]]

FORCE - will manually commit an in-doubt *distributed* transaction

force_text - transaction identifier (see the DBA_2PC_PENDING view*)*

int - sets a specific SCN.

If a network or machine failure prevents a distributed transaction from committing properly, Oracle will store any commit comment in the data dictionary along with the transaction ID.

INPUT:

SQL>commit;

RESULT: Commit complete.

2. ROLLBACK

Undo work done (transactional).
Syntax:
ROLLBACK [WORK] [TO [SAVEPOINT]'*savepoint_text_identifier*'];

ROLLBACK [WORK] [FORCE '*force_text*'];

FORCE - will manually rollback an in-doubt *distributed* transaction

INPUT:

SQL>rollback;

RESULT: Rollback complete.

3. SAVEPOINT
Save changes to a point (transactional).

Syntax:

SAVEPOINT *text_identifier*

Example:
```
UPDATE employees
SET salary = 95000
WHERE last_name = 'Smith';

SAVEPOINT justsmith;

UPDATE employees
SET salary = 1000000;

SAVEPOINT everyone;

SELECT SUM(salary) FROM employees;

ROLLBACK TO SAVEPOINT justsmith;

COMMIT;
```

Writing and Practice of Simple Queries.

To write simple queries and practice them.

1**. Get the description of EMP table.**

SQL>desc emp;

RESULT:

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(3)
AGE		NUMBER(3)
ESAL		NUMBER(10)

2. Get the description DEPT table.

SQL>desc dept;

RESULT:

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

3.List all employee details.

SQL>select * from emp;

RESULT:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	AGE	ESAL
7369	SMITH	CLERK	7902	17-DEC-80	800	0	20	25	0
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	25	0
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	25	0
7566	JONES	MANAGER	7839	02-APR-81	2975	500	20	25	0
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	1400	30	25	0

4.List all employee names and their salaries, whose salary lies between 1500/- and 3500/- both inclusive.

INPUT
SQL>select ename from emp where sal between 1500 and 3500;
RESULT

ENAME

ALLEN
JONES
BLAKE
CLARK
SCOTT
TURNER
FORD
russel
greg

9 rows selected.

5. List all employee names and their and their manager whose manager is 7902 or 7566 0r 7789.

INPUT SQL>select ename from emp where mgr in(7602,7566,7789);
RESULT

ENAME

SCOTT
FORD

6. List all employees which starts with either J or T.

INPUT SQL>select ename from emp where ename like 'J%' or ename like 'T%';

RESULT:

ENAME

JONES
TURNER
JAMES

7. List all employee names and jobs, whose job title includes M or P.

INPUT SQL>select ename,job from emp where job like 'M%' or job like 'P%';

RESULT:

ENAME	*JOB*
JONES	MANAGER
BLAKE	MANAGER
CLARK	MANAGER
KING	PRESIDENT

8. List all jobs available in employee table.

INPUT SQL>select distinct job from emp;

RESULT:

JOB
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
assistant
clerk

7 rows selected.

9. List all employees who belongs to the department 10 or 20.

INPUT SQL>select ename from emp where deptno in (10,20);

RESULT:

ENAME
SMITH
JONES
CLARK
SCOTT
KING
ADAMS
FORD
MILLER

8 rows selected.

10. List all employee names , salary and 15% rise in salary.

INPUT SQL>select ename , sal , sal+0.15* sal from emp;

RESULT:

ENAME	SAL	SAL+0.15*SAL
SMITH	800	920
ALLEN	1600	1840
WARD	1250	1437.5
JONES	2975	3421.25
MARTIN	1250	1437.5
BLAKE	2850	3277.5
CLARK	2450	2817.5

7 rows selected.

11. List minimum , maximum , average salaries of employee.

INPUT SQL>select min(sal),max(sal),avg(sal) from emp;

RESULT:

MIN(SAL)	MAX(SAL)	AVG(SAL)
3	5000	1936.94118

12. Find how many job titles are available in employee table.

INPUT SQL>select count (distinct job) from emp;

RESULT:

COUNT(DISTINCTJOB)
7

13. What is the difference between maximum and minimum salaries of employees in the organization?

INPUT SQL>select max(sal)-min(sal) from emp;

RESULT:

MAX(SAL)-MIN(SAL)
4997

14. Display all employee names and salary whose salary is greater than minimum salary of the company and job title starts with 'M'.

INPUT SQL>select ename,sal from emp where job like 'M%' and sal > (select min (sal) from emp);

RESULT

ENAME	SAL
JONES	2975
BLAKE	2850
CLARK	2450

15. Find how much amount the company is spending towards salaries.

INPUT SQL>select sum (sal) from emp;

RESULT

SUM(SAL)
32928

16. Display name of the dept. with deptno 20.

INPUT SQL>select ename from emp where deptno = 20;

RESULT

ENAME
SMITH
JONES
SCOTT
ADAMS

17. List ename whose commission is NULL.

INPUT SQL>select ename from emp where comm is null;

RESULT

ENAME
CLARK
SCOTT
KING
ADAMS
JAMES
FORD

6 rows selected.

18. Find no.of dept in employee table.

INPUT SQL>select count (distinct ename) from emp;

RESULT COUNT(DISTINCTENAME

17

19. List ename whose manager is not NULL.

INPUT SQL>select ename from emp where mgr is not null;

RESULT ENAME

SMITH
ALLEN
WARD
JONES
MARTIN

5 rows selected.

Writing Queries using GROUP BY and other clauses.

To write queries using clauses such as GROUP BY, ORDER BY, etc. and retrieving information by joining tables.

Source tables: emp, dept, programmer, software, study.

Order by : The order by clause is used to display the results in sorted order.
Group by : The attribute or attributes given in the clauses are used to form groups. Tuples with the same value on all attributes in the group by clause are placed in one group.
Having: SQL applies predicates (conditions) in the having clause after groups have been formed, so aggregate function be used.

1. Display total salary spent for each job category.

INPUT SQL>select job,sum (sal) from emp group by job;

RESULT

JOB	SUM(SAL)
ANALYST	6000
CLERK	23050
MANAGER	8275
PRESIDENT	5000
SALESMAN	5600
assistant	2200
clerk	2003

7 rows selected.

2. Display lowest paid employee details under each manager.

INPUT SQL>select ename, sal from emp where sal in (select min(sal) from emp group by mgr);

RESULT

ENAME	***SAL***
chai	3
JAMES	950
MILLER	1000
ADAMS	1100
russel	2200

5 rows selected.

3. Display number of employees working in each department and their department name.

INPUT SQL> select dname, count (ename) from emp, dept where emp.deptno=dept.deptno group by dname;

RESULT

DNAME	COUNT(ENAME)
ACCOUNTING	3
RESEARCH	5
SALES	***9***

4. Display the sales cost of package developed by each programmer.

INPUT SQL>select pname, sum(scost) from software group by pname;

RESULT

PNAME	SUM(SCOST)
john	12000
kamala	12000
raju	12333

3 rows selected.

5. Display the number of packages sold by each programmer.

INPUT SQL>select pname, count(title) from software group by pname;

RESULT

PNAME	COUNT(TITLE)
john	1
kamala	1
raju	1
ramana	1
rani	1

5 rows selected.

6. Display the number of packages in each language for which the development cost is less than thousand.

INPUT SQL>select devin, count(title) from software where dcost < 1000 group by devin;
RESULT

DEVIN	COUNT(TITLE)
cobol	1

7. Display each institute name with number of students.

INPUT SQL>select splace, count(pname) from study group by splace;
RESULT

SPLACE	COUNT(PNAME)
BDPS	2
BITS	1
BNRILLIANI	1
COIT	1
HYD	1

5 rows selected.

8. How many copies of package have the least difference between development and selling cost, were sold?

INPUT SQL>select sold from software where scost – dcost=(select min(scost – dcost) from software);
RESULT

SOLD
11

9. Which is the costliest package developed in Pascal.

INPUT SQL>select title from software where devin = 'PASCAL' and dcost = (select max(dcost)from software where devin = 'PASCAL');
RESULT
no rows selected

10. Which language was used to develop most no .of packages.

INPUT SQL>select devin, count (*) from software group by devin having count(*) = (select max(count(*)) from software group by devin);
RESULT

DEVIN	COUNT(*)
jsp	2

11.Who are the male programmers earning below the average salary of female programmers?

INPUT SQL>select pname from programmer where sal < (select avg(sal) from programmer where sex = 'F') and sex = 'M';
RESULT

PNAME
vijay

12. Display the details of software developed by the male programmers earning more than 3000/-.

INPUT SQL>select programmer.pname, title, devin from programmer, software where sal > 3000 and sex = 'M' and programmer.pname = software.pname;
RESULT

no rows selected

13. Display the details of software developed in c language by female programmers of pragathi.

INPUT SQL>select software.pname, title, devin, scost, dcost, sold from programmer, software, study where devin = 'c' and sex ='F' and splace = 'pragathi' and programmer.pname = software.pname and software.pname = study.pname;

14. Which language has been stated by the most of the programmers as proficiency one?

INPUT SQL>select prof1, count(*) from programmer group by prof1 having count (*) = (select max (count (*)) from programmer group by prof1);

Writing Nested Queries.

To write queries using Set operations and to write nested queries.

Set Operations:

UNION	-	OR
INTERSECT	-	AND
EXCEPT -	-	NOT

NESTED QUERY:- A nested query makes use of another sub-query to compute or retrieve the information.

1. Find the name of the institute in which the person studied and developed the costliest package.

INPUT SQL>select splace, pname from study where pname = (select pname from software where scost = (select max (scost) from software);

RESULT

SPLACE	*PNAME*
SAHBHARI	MARY

2. **Find the salary and institute of a person who developed the highest selling package.**

INPUT SQL> select study.pname, sal, splace from study, programmer where study.pname = programmer.pname and study.pname = (select pname from software where scost = (select max (scost) from software));

RESULT

PNAME	*SAL*	*SPLACE*
MARY	4500	SABHARI

3. How many packages were developed by the person who developed the cheapest package.

INPUT SQL>select pname, count (title) from software where dcost = (select min(dcost) from software) group by pname;

RESULT

PNAME	COUNT(TITLE)
VIJAY	1

4. Calculate the amount to be recovered for those packages whose development cost has not yet recovered.

INPUT SQL>select title , (dcost-scost) from software where dcost > scost;

5. Display the title, scost, dcost, difference of scost and dcost in the descending order of difference.

INPUT SQL> select title, scost, dcost, (scost - dcost) from software descending order by (scost-dcost);

6. Display the details of those who draw the same salary.

INPUT SQL> select p.pname, p.sal from programmer p, programmer t where p.pname <> t.pname and p.sal = t.sal;(or)

INPUT SQL>select pname,sal from programmer t where pname<>t.pname and sal= t.sal;

Writing Queries using functions.

AIM: To write queries using single row functions and group functions.

1. Display the names and dob of all programmers who were born in january.

INPUT SQL>select pname , dob from programmer where to_char (dob,'MON')='JAN';

2. Calculate the experience in years of each programmer and display along with programmer name in descending order.

INPUT SQL> select pname, round (months_between(sysdate, doj)/12, 2) "EXPERIENCE" from programmer order by months_between (sysdate, doj) desc;

3. List out the programmer names who will celebrate their birthdays during current month.

INPUT SQL>select pname from programmer where to_char(dob,'MON') like to_char (sysdate, 'MON');

4. Display the least experienced programmer's details.

INPUT SQL>select * from programmer where doj = (select max (doj) from programmer);

5. Who is the most experienced programmer knowing pascal.

INPUT SQL>select pname from programmer where doj = (select min (doj) from programmer);

6. Who is the youngest programmer born in 1965.

INPUT SQL> select pname , dob from programmer where dob = (select max (dob) from programmer where to_char (dob,'yy') = 65);

7. In which year, most of the programmers are born.

INPUT SQL>select to_char (dob , 'YY') from programmer group by to_char (dob, 'YY') having count(*) = (select max (count(*)) from programmer group by to_char(dob,'YY');

8. In which month most number of programmers are joined.

INPUT SQL>select to_char (doj,'YY') from programmer group by to_char (doj,'YY') having count (*) = (select max (count(*)) from programmer group by to_char (doj,'YY');

9. What is the length of the shortest name in programmer table ?

INPUT SQL>select length (pname) from programmer where length (pname) = select min (length (pname) from programmer);

10. Display the names of the programmers whose name contains up to 5 characters.

INPUT SQL>select pname from programmer where length (pname) <=5;

11. Display all packages names in small letters and corresponding programmer names in uppercase letters.

INPUT SQL>select lower (title), upper (pname) from software;

Writing Queries on views.

AIM: To write queries on views.

1. Create a view from single table containing all columns from the base table.

SQL>create view view1 as (select * from programmer);

2. Create a view from single table with selected columns.

SQL>create a view view2 as (select pname,dob,doj,sex,sal from programmer);

3. Create a view from two tables with all columns.

SQL>create view xyz as select * from programmer full natural join software;

4. Create a view from two tables with selected columns.

SQL> create view lmn as (select programmer, pname, title, devin from programmer, software where sal < 3000 and programmer.pname = software.pname);

5. Check all DML commands with above 4 views.

INPUT SQL> insert into view1 values ('ramu','12-sep-03','28-jan-85','f','dbase','oracle',74000);

RESULT

1 row created;

INPUT SQL>update view1 set salary =50000 where pname like 'raju';

RESULT 1 row updated.

Note: update command does not works for all queries on views.

INPUT SQL>delete from view1 where pname like 'raju';

RESULT 1 row deleted.

6. Drop views which you generated.

INPUT SQL>drop view view1;

RESULT View dropped;

INPUT SQL>drop view view2;

RESULT View dropped;

INPUT SQL>drop view xyz;

Writing PL/SQL block for insertion into a table.

To write a PL/SQL block for inserting rows into EMPDET table with the following Calculations:
HRA=50% OF BASIC
DA=20% OF BASIC
PF=7% OF BASIC
NETPAY=BASIC+DA+HRA-PF

INPUT

```
DECLARE
        ENO1 empdet.eno%type;
        ENAME1 empdet.name%type;
        DEPTNO1  empdet.deptno%type;
        BASIC1 empdet.basic%type;
        HRA1 empdet.HRA%type;
        DA1 empdet.DA%type;
        PF1 empdet.pf%type;
        NETPAY1 empdet.netpay%type;
BEGIN
        ENO1:=&ENO1;
        ENAME1:='&ENAME1';
        DEPTNO1:=&DEPTNO1;
        BASIC1:=&BASIC1;
        HRA1:=(BASIC1*50)/100;
        DA1:=(BASIC1*20)/100;
        PF1:=(BASIC1*7)/100;
        NETPAY1:=BASIC1+HRA1+DA1-PF1;

        INSERT INTO EMPDET VALUES (ENO1, ENAME1, DEPTNO1, BASIC1, HRA1,
DA1, PF1, NETPAY1);
END;
```

RESULT:

```
SQL> @BASIC
Enter value for eno1: 104
old  11: ENO1:=&ENO1;
new  11: ENO1:=104;
Enter value for ename1: SRINIVAS REDDY
old  12: ENAME1:='&ENAME1';
new  12: ENAME1:='SRINIVAS REDDY';
Enter value for deptno1: 10
old  13: DEPTNO1:=&DEPTNO1;
new  13: DEPTNO1:=10;
Enter value for basic1: 6000
old  14: BASIC1:=&BASIC1;
new  14: BASIC1:=6000;

PL/SQL procedure successfully completed.


SQL>/
Enter value for eno1: 105
old  11: ENO1:=&ENO1;
new  11: ENO1:=105;
Enter value for ename1: CIRAJ
old  12: ENAME1:='&ENAME1';
new  12: ENAME1:='CIRAJ';
Enter value for deptno1: 10
old  13: DEPTNO1:=&DEPTNO1;
new  13: DEPTNO1:=10;
Enter value for basic1: 6000
old  14: BASIC1:=&BASIC1;
new  14: BASIC1:=6000;

PL/SQL procedure successfully completed.

SQL> SELECT * FROM EMPDET;
```

RESULT

ENO	NAME	DEPTNO	BASIC	HRA	DA	PF	NETPAY
101	SANTOSH	10	5000	2500	1000	350	8150
102	SHANKAR	20	5000	2500	1000	350	8150
103	SURESH	20	5500	2750	1100	385	8965
104	SRINIVASA REDDY	10	6000	3000	1200	420	9780
105	CIRAJ	10	6000	3000	1200	420	9780

Writing PL/SQL block for checking armstrong number

To write a PL/SQL block to check whether given number is Armstrong or not.

INPUT

```
DECLARE
        num number(5);
        rem number(5);
        s number(5):=0;
        num1 number(5);
BEGIN
        num:=&num;
        num1:=num;
        while(num>0)
        loop
                rem:=mod(num,10);
                s:=s+power(rem,3);
                num:=trunc(num/10);
        End loop;
        if (s=num1)then
                dbms_RESULT.put_line(num1||' IS ARMSTRONG NUMBER ');
        else
                dbms_RESULT.put_line(num1||' IS  NOT ARMSTRONG NUMBER ');
         End if;
END;
/
```

RESULT:

```
SQL>@arm
Enter value for num: 153
old   7:  num:=&num;
new   7:  num:=153;
153 IS ARMSTRONG NUMBER

PL/SQL procedure successfully completed.

SQL> /
Enter value for num: 123
old   7:  num:=&num;
new   7:  num:=123;
123 IS  NOT ARMSTRONG NUMBER

PL/SQL procedure successfully completed.
```

Writing a PL/SQL block for checking a number even or odd.

AIM: To write a PL/SQL block to check whether a given number is Even or Odd.

INPUT

```
DECLARE
        num number(5);
        rem number;
BEGIN
        num:=&num;
        rem:=mod(num,2);
        if rem=0
        then
                dbms_RESULT.put_line(' Number '||num||' is Even');
        else
                 dbms_RESULT.put_line(' Number '||num||' is Odd');
        end if;
END;
```

RESULT:

```
SQL>start even
Enter value for num: 6
old   5:   num:=&num;
new   5:   num:=6;
Number 6 is Even

PL/SQL procedure successfully completed.

SQL> /
Enter value for num: 3
old   5:   num:=&num;
new   5:   num:=3;
Number 3 is Odd
PL/SQL procedure successfully completed.
```

Writing PL/SQL block to find sum of digits of a given number.

To write a PL/SQL block to find Sum of Digits of a given Number.

INPUT

```
DECLARE
        num number(5);
        rem number(5);
        sm number(5):=0;
        num1 number(5);
BEGIN
        num:=&num;
        num1:=num;
        while(num>0) loop
                rem:=mod(num,10);
                sm:=sm+rem;
                num:=trunc(num/10);
        end loop;
        dbms_RESULT.put_line('SUM OF DIGITS OF  '||num1||' IS: '||sm);
end;
/
```

RESULT:

```
SQL> @sum
INPUT truncated to 2 characters
Enter value for num: 123
old   7:  num:=&num;
new   7:  num:=123;
SUM OF DIGITS OF  123 IS: 6
PL/SQL procedure successfully completed.

SQL> @sum
INPUT truncated to 2 characters
Enter value for num: 456
old   7:  num:=&num;
new   7:  num:=456;
SUM OF DIGITS OF  456 IS: 15
PL/SQL procedure successfully completed.
```

Writing PL/SQL block for generating Fibonacci series.

To write a PL/SQL block to Generate Fibonacci Series

INPUT

```
DECLARE
        num number(5);
        f1 number(5):=0;
        f2 number(5):=1;
        f3 number(5);
        i number(5):=3;
BEGIN
        num:=&num;
        dbms_RESULT.put_line('THE FIBONACCI SERIES IS:');
        dbms_RESULT.put_line(f1);
        dbms_RESULT.put_line(f2);
        while(i<=num)          loop
                f3:=f1+f2;
                dbms_RESULT.put_line(f3);
                f1:=f2;
                f2:=f3;
                i:=i+1;
        end loop;
END;
/
```

RESULT:

```
SQL> start fib
Enter value for num: 10
old   8:  num:=&num;
new   8:  num:=10;
THE FIBONACCI SERIES IS:
0
1
1
2
3
5
8
13
21
34
PL/SQL procedure successfully completed.
```

Writing PL/SQL block for checking palindrome.

To write a PL/SQL block to Check the Given String is Palindrome or Not.

INPUT

```
DECLARE
        name1 varchar2(20);
        name2 varchar2(20);
        l number(5);
BEGIN
        name1:='&name1';
        l:=length(name1);
        while l>0  loop
                name2:=name2||substr(name1,l,1);
                l:=l-1;
        end loop;
        dbms_RESULT.put_line('REVERSE OF STRING IS:'||NAME2);
        if(name1=name2)       then
                dbms_RESULT.put_line(name1||' IS PALINDROME ');
        else
                dbms_RESULT.put_line(name1||' IS NOT PALINDROME ');
        end if;
END;
/
```

RESULT

```
Enter value for name1: LIRIL
old   6: name1:='&name1';
new   6: name1:='LIRIL';
REVERSE OF STRING IS:LIRIL
LIRIL IS PALINDROME

PL/SQL procedure successfully completed.

SQL> /
Enter value for name1: MADAM
old   6: name1:='&name1';
new   6: name1:='MADAM';
REVERSE OF STRING IS:MADAM
MADAM IS PALINDROME

PL/SQL procedure successfully completed.
```

Writing PL/SQL block to demonstrate Cursors.

To write a Cursor to display the list of Employees and Total Salary Department wise.

INPUT

```
DECLARE
        cursor c1 is select * from dept;
        cursor c2 is select * from emp;
       s emp.sal%type;

BEGIN
       for i in c1       loop
              s:=0;
              dbms_RESULT.put_line('-----------------------------------------------');
              dbms_RESULT.put_line('Department is :' || i.deptno ||' Department name is:' ||
i.dname);
              dbms_RESULT.put_line('-----------------------------------------------');
              for j in c2    loop
                     if ( i.deptno=j.deptno) then
                            s:=s+j.sal;
                           dbms_RESULT.put_line(j.empno|| '    '|| j.ename || '    '|| j.sal );
                      end if;
              end loop;
              dbms_RESULT.put_line('-----------------------------------------------');
              dbms_RESULT.put_line('Total salary is:   '|| s);
              dbms_RESULT.put_line('-----------------------------------------------');
       end loop;
END;
```

RESULT:

```
SQL> @abc
-------------------------------------------------------------------------------
Department is :10      Department name is : ACCOUNTING
-------------------------------------------------------------------------------
7782    CLARK    2450
7839    KING    5000
7934    MILLER    1300
-------------------------------------------------------------------------------
Total salary is:   8750
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Department is :20  Department name is:RESEARCH
-------------------------------------------------------------------------------
```

```
7369    SMITH    800
7566    JONES    2975
7788    SCOTT    3000
7876    ADAMS    1100
7902    FORD    3000
--------------------------------------------------------------------------------
Total salary is:   10875
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Department is :30  Department name is:SALES
--------------------------------------------------------------------------------
7499    ALLEN    1600
7521    WARD    1250
7654    MARTIN    1250
7698    BLAKE    2850
7844    TURNER    1500
7900    JAMES    950
--------------------------------------------------------------------------------
Total salary is:   9400
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Department is  :40  Department name is:OPERATIONS
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Total salary is:   0
--------------------------------------------------------------------------------
PL/SQL procedure successfully completed.
```

Writing PL/SQL CURSOR

To write a Cursor to display the list of employees who are Working as a Managers or Analyst.

INPUT

```
DECLARE
        cursor c(jb varchar2) is select ename from emp where job=jb;
        em emp.job%type;
BEGIN
        open c('MANAGER');
        dbms_RESULT.put_line(' EMPLOYEES WORKING AS MANAGERS ARE:');
        loop
                fetch c into em;
                exit when c%notfound;
                dbms_RESULT.put_line(em);
        end loop;
        close c;
```

```
        open c('ANALYST');
        dbms_RESULT.put_line(' EMPLOYEES WORKING AS ANALYST ARE:');
        loop
                fetch c into em;
                exit when c%notfound;
                dbms_RESULT.put_line(em);
        end loop;
        close c;
END;
```

RESULT:

```
EMPLOYEES WORKING AS MANAGERS ARE:
JONES
BLAKE
CLARK
EMPLOYEES WORKING AS ANALYST ARE:
SCOTT
FORD

PL/SQL procedure successfully completed.
```

Writing PL/SQL CURSOR

To write a Cursor to display List of Employees from Emp Table in PL/SQL block

INPUT

```
DECLARE
        cursor c is select empno, ename, deptno, sal from emp ;
        i emp.empno%type;
        j emp.ename%type;
        k emp.deptno%type;
        l emp.sal%type;
BEGIN
        open c;
        dbms_RESULT.put_line('Empno, name, deptno, salary of employees are:= ');
        loop
                fetch c into i, j, k, l;
                exit when c%notfound;
                dbms_RESULT.put_line(i||' '||j||' '||k||' '||l);
        end loop;
        close c;
END;
```

RESULT:

```
SQL> @EMP
Empno,name,deptno,salary of employees are:=
7369      SMITH     20    800
7499      ALLEN     30    1600
7521      WARD      30    1250
7566      JONES     20    2975
7654      MARTIN    30    1250
7698      BLAKE     30    2850
7782      CLARK     10    2450
7788      SCOTT     20    3000
7839      KING      10    5000
7844      TURNER    30    1500
7876      ADAMS     20    1100
7900      JAMES     30    950
7902      FORD      20    3000
7934      MILLER    10    1300
PL/SQL procedure successfully completed.
```

Writing PL/SQL CURSOR

To write a Cursor to find employee with given job and deptno.

INPUT

```
DECLARE
        cursor c1(j varchar2, dn number) is select empno, ename from emp where job=j and
deptno=dn;
        row1 emp%rowtype;
        jb emp.job%type;
        d emp.deptno%type;
BEGIN
        jb:='&jb';
        d:=&d;
        open c1(jb,d);
        fetch c1 into row1.empno,row1.ename;
        if c1%notfound then
                dbms_RESULT.put_line('Employee does not exist');
        else
                dbms_RESULT.put_line('empno is:'||row1.empno||'  ' ||'employee name
is:'||row1.ename);
        end if;
END;
```

RESULT:

```
SQL> @CUR
Enter value for jb: MANAGER
old   7: jb:='&jb';
new   7: jb:='MANAGER';
Enter value for d: 20
old   8: d:=&d;
new   8: d:=20;
empno is:7566  employee name is:JONES
PL/SQL procedure successfully completed.

SQL> /
Enter value for jb: CLERK
old   7: jb:='&jb';
new   7: jb:='CLERK';
Enter value for d: 40
old   8: d:=&d;
new   8: d:=40;
Employee does not exist
PL/SQL procedure successfully completed.
```

Writing PL/SQL BLOCK using string functions.

To write a PL/SQL block to apply String Functions on a given input String.

INPUT

```
DECLARE
        a varchar2(20);
        l number(5);
BEGIN
        a:='&a';
        l:=length(a);
        dbms_RESULT.put_line('Using Lower Function:' ||   lower(a));
        dbms_RESULT.put_line('Using  UPPER Function:' ||  upper(a));
        dbms_RESULT.put_line('Using Initcap Function:' || initcap(a));
        dbms_RESULT.put_line('Using Substring Function:' || substr(a,l,1));
        dbms_RESULT.put_line('Using Substring  Function:' || substr(a,1,3));
        dbms_RESULT.put_line('Using Ltrim function for xxxabcxxxx:' ||
ltrim('xxxabcxxxx','x'));
        dbms_RESULT.put_line('Using Rtrim function for xxxabcxxxx:'||
rtrim('xxxabcxxxx','x'));
        dbms_RESULT.put_line('Using Lpad function :'|| lpad(a,l+4,'*'));
        dbms_RESULT.put_line('Using Rpad function :'|| rpad(a,l+4,'*'));
END;
```

RESULT:

```
SQL>@STR
Enter value for a: santosh reddy
old   5:    a:='&a';
new   5:    a:='santosh reddy';
Using Lower Function:santosh reddy
Using  UPPER Function:SANTOSH REDDY
Using Initcap Function:Santosh Reddy
Using Substring Function:y
Using Substring  Function:san
Using Ltrim function for xxxabcxxxx:abcxxxx
Using Rtrim function for xxxabcxxxx:xxxabc
Using Lpad function :****santosh reddy
Using Rpad function :santosh reddy****

PL/SQL procedure successfully completed.
SQL> /
Enter value for a: UMA SHANKAR
old   5:    a:='&a';
new   5:    a:='UMA SHANKAR';
Using Lower Function:uma shankar
Using  UPPER Function:UMA SHANKAR
Using Initcap Function:Uma Shankar
Using Substring Function:R
Using Substring  Function:UMA
Using Ltrim function for xxxabcxxxx:abcxxxx
Using Rtrim function for xxxabcxxxx:xxxabc
Using Lpad function :****UMA SHANKAR
Using Rpad function :UMA SHANKAR****

PL/SQL procedure successfully completed
```

Writing PL/SQL triggers

To write a TRIGGER to ensure that DEPT TABLE does not contain duplicate of null values in DEPTNO column.

INPUT

```
CREATE OR RELPLACE TRIGGER trig1 before insert on dept for each row
DECLARE
        a number;
BEGIN
        if(:new.deptno is Null) then
                raise_application_error(-20001,'error::deptno cannot be null');
        else
                select count(*) into a from dept where deptno=:new.deptno;
                if(a=1) then
                        raise_application_error(-20002,'error:: cannot have duplicate deptno');
                end if;
        end if;
END;
```

RESULT:

```
SQL> @trigger
Trigger created.

SQL> select * from dept;
  DEPTNO DNAME         LOC
   --------- -------------- -------------
     10 ACCOUNTING     NEW YORK
     20 RESEARCH       DALLAS
     30 SALES          CHICAGO
     40 OPERATIONS     BOSTON

SQL> insert into dept values(&deptnp,'&dname','&loc');
Enter value for deptnp: null
Enter value for dname: marketing
Enter value for loc: hyd
old   1: insert into dept values(&deptnp,'&dname','&loc')
new   1: insert into dept values(null,'marketing','hyd')
insert into dept values(null,'marketing','hyd')
        *
ERROR at line 1:
ORA-20001: error::deptno cannot be null
ORA-06512: at "SCOTT.TRIG1", line 5
ORA-04088: error during execution of trigger 'SCOTT.TRIG1'
```

```
SQL> /
Enter value for deptnp: 10
Enter value for dname: manager
Enter value for loc: hyd
old   1: insert into dept values(&deptnp,'&dname','&loc')
new   1: insert into dept values(10,'manager','hyd')
insert into dept values(10,'manager','hyd')
            *
ERROR at line 1:
ORA-20002: error:: cannot have duplicate deptno
ORA-06512: at "SCOTT.TRIG1", line 9
ORA-04088: error during execution of trigger 'SCOTT.TRIG1'

SQL> /
Enter value for deptnp: 50
Enter value for dname: MARKETING
Enter value for loc: HYDERABAD
old   1: insert into dept values(&deptnp,'&dname','&loc')
new   1: insert into dept values(50,'MARKETING','HYDERABAD')

1 row created.
SQL> select * from dept;

    DEPTNO DNAME         LOC
    --------- -------------- -------------
     10 ACCOUNTING     NEW YORK
     20 RESEARCH       DALLAS
     30 SALES          CHICAGO
     40 OPERATIONS     BOSTON
     50 MARKETING      HYDE
```

Locking Table.

AIM: To learn commands related to Table Locking

LOCK TABLE Statement Manually lock one or more tables.

Syntax:

LOCK TABLE [*schema*.] *table* [*options*] IN *lockmode* MODE [NOWAIT]

LOCK TABLE [*schema*.] *view* [*options*] IN *lockmode* MODE [NOWAIT]

Options:

PARTITION (*partition*)
SUBPARTITION (*subpartition*)
@dblink

lockmodes:
EXCLUSIVE
SHARE
ROW EXCLUSIVE
SHARE ROW EXCLUSIVE
ROW SHARE* | SHARE UPDATE*

If NOWAIT is omitted Oracle will wait until the table is available.

Several tables can be locked with a single command - separate with commas

e.g. LOCK TABLE table1,table2,table3 IN ROW EXCLUSIVE MODE;

Default Locking Behaviour :

A pure SELECT will not lock any rows.

INSERT, UPDATE or DELETE's - will place a ROW EXCLUSIVE lock.

SELECT...FROM...FOR UPDATE NOWAIT - will place a ROW EXCLUSIVE lock.

Multiple Locks on the same rows with LOCK TABLE

Even when a row is locked you can always perform a SELECT (because SELECT does not lock any rows) in addition to this, each type of lock will allow additional locks to be granted as follows.

ROW SHARE = Allow ROW EXCLUSIVE or ROW SHARE or SHARE locks to be granted to the locked rows.

ROW EXCLUSIVE = Allow ROW EXCLUSIVE or ROW SHARE locks to be granted to the locked rows.

SHARE ROW EXCLUSIVE = Allow ROW SHARE locks to be granted to the locked rows.

SHARE = Allow ROW SHARE or SHARE locks to be granted to the locked rows.

EXCLUSIVE = Allow SELECT queries only

Although it is valid to place more than one lock on a row, UPDATES and DELETE's may still cause a *wait* if a conflicting row lock is held by another transaction.

Generation of Forms using ORACLE FORM BUILDER

To design a form using Oracle Developer 2000

Introduction

Use Form Builder to simplify for the creation of data-entry screens, also known as Forms. Forms are the applications that connect to a database, retrieve information requested by the user, present it in a layout specified by Form designer, and allow the user to modify or add information. Form Builder allows you to build forms quickly and easily.

In this Hands-On, you learn how to: Create a Data block for the “Customer” table, Create a layout, Use “content” canvas, Use “execute query”, Navigate a table, Use next, previous record, Enter query, Manipulate table’s record, Insert, Update, Delete and Save record.

Form Builder Tool

Open the "Form Builder" tool.

Welcome window

You will get the ‘Welcome to the Form Builder’ window. If you don’t want to get this window anymore uncheck the ‘Display at startup’ box. You can start your work with any of the following options:

Use the data Block Wizard
Build a new form manually
Open an existing form
Build a form based on a template

The default is ‘Use the data Block Wizard.’ If you want to build a new form manually, click on "Cancel” or check ‘Build a new form manually’ and click ‘OK.’

Connect to database

In the ‘Object Navigator’ window, highlight "Database Objects." Go to the Main menu and choose "File," then "Connect."

In the ‘Connect’ window, login in as “scott” password “tiger,” then click “CONNECT.”

Notice that the box next to ‘Database Objects’ is not empty anymore and it has a ‘+’ sign in it. That will indicate that this item is expandable and you are able to see its entire objects.

Click on the ‘+’ sign next to the ‘Database Objects’ to expand all database schemas.

Create a Module

In the ‘Object Navigator’ window, highlight module1. This is a default name. Go to the Main menu and choose “File,” select “Save as” to store the new object in the “iself” folder and save it as customer data entry. "c:_de." In this example the ‘DE’ abbreviation stands for Data Entry.

Create a Data Block

In the ‘Object Navigator’ window, highlight "Data Blocks,” and click on the "create” icon. The ‘Create’ icon is in the vertical tool bar in the ‘Object Navigator’ window. It is a green ‘+’ sign. If you drag your cursor on the icon a tooltip will show ‘Create.’

New Data Block

In the ‘New Data Block’ window, choose the default option “Data Block Wizard” and click "OK."

Welcome Data Block

In the ‘Welcome Data Block Wizard’ window click on the “NEXT” icon.

Type of Data Block

Select the type of data block you would like to create by clicking on a radio button. Select the default option ‘Table or View’ and then click “NEXT” again.

Selecting Tables

Click on “browse.” In the ‘Tables’ window, highlight the "cust11” table; then click "OK."

Selecting columns for the Data Block Wizard

To choose all columns, click on the two arrow signs in the ‘Data Block Wizard’ window. To choose selected columns, click on the one arrow sign. And then select all columns, and click “next.”

Layout Wizard

End of the Data Block Wizard and beginning of the Layout Wizard
In the ‘Congratulations’ screen, use the default checkmark radio button (Create the data block, then call the Layout Wizard), and click "Finish." You can also use the Data Block Wizard to modify your existing data block. Simply select the data block in the Object Navigator and click the Data Block Wizard toolbar button, or choose ‘Data Block wizard’ from the ‘Tools’ menu.

Welcome screen

In the ‘Welcome to the Layout Wizard’ window, click ”Next.”

Selecting canvas

In the ‘Layout Wizard’ window, select the "new canvas" option. Canvas is a place that you will have your objects such as columns, titles, pictures, etc. If you have already had your canvas, select the canvas and then click on the next. The following are different types of canvases: Content, Stacked, Vertical Toolbar, Horizontal Toolbar, and Tab.

Think of the ‘Content’ canvas as one flat place to have all your objects. In the stacked canvas, you can have multiple layers of objects and it is the same as the tab canvas. You use the vertical or horizontal toolbar canvases for your push buttons. Check the different types of canvases by clicking on the ‘down arrow’ box next to the ‘Type’ field. Select "content," then click “Next.”

Selecting Columns for the Layout Wizard

In the ‘Layout Wizard’ window, select all the columns. These are the columns that you want to be displayed on the canvas. Then click “Next.”

Change your objects appearances

Change size or prompt if needed. In this window, you can enter a prompt, width, and height for each item on the canvas. You can change the measurement units. As a default the default units for item width and height are points. You can change it to inch or centimeter. When you change size, click “Next.”

Selecting a layout style

Select a layout style for your frame by clicking a radio button. Select "Form," if you want one record at a time to be displayed. Select “Tabular,” if you want more than one record at a time to be displayed. Select "Forms," and then click “next.”

Record layout

Type the "Frame Title" and click "next." Checkmark the ‘Display Scrollbar’ box when you use multiple records or the ‘Tabular’ option.

Congratulation Screen

In the ‘Congratulations’ window, click "Finish."
You will see the output layout screen.
Make some window adjustments and then run the form. To run the form, click on the ‘Run’ icon. The ‘Run’ icon is on the horizontal toolbar in the ‘CUSTOMER_DE’ canvas.

The object module should be compiled successfully before executing the Form.

Execute Query

Click on the "Execute Query" icon below the main menu. If you drag the cursor on the toolbar in the ‘Forms Runtime’ window, a tooltip will be displayed and you see ‘Execute Query.’
So to know all your option, drag your cursor to view all the icon descriptions.

Next Record

Click on the "Next Record" icon to navigate to the next record.

Previous Record

Click on the "Previous Record" icon to navigate to the previous record.
This is an easy way to navigate through the “Customer” table.

Enter Query

Click on the "Enter Query" icon to query selected records.

Insert Record

Click "Insert Record" to add new customer. All items on the forms will be blanked. You can either type all the customer information or duplicate it from pervious record.

Duplicate Record

To duplicate the previous record, go to the main menu and select the ‘Record’ sub-menu. A drop down menu will be displayed. Select the ‘Duplicate’ option in the sub-menu.

Apply the changes. Remember in this stage, your record was inserted but not committed yet.

Next and Previous Record

Click "next record" and "previous record" to navigate through the records and the one was added.

Save transactions

Click "Save" to commit the insert statement.

Delete Record

Click "Remove Record" to delete the record.

Lock a Record

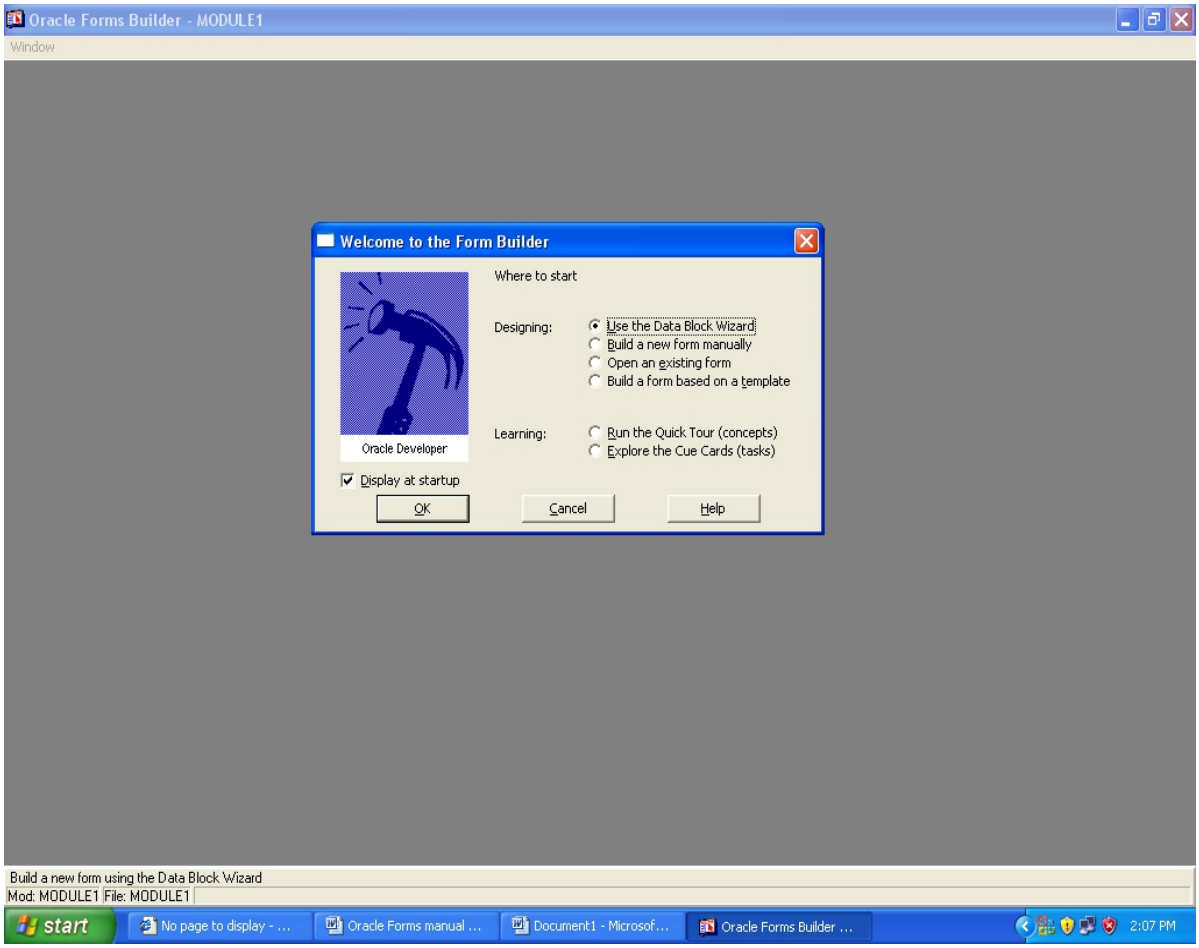
You can also lock the record.

Exit from Form Runtime

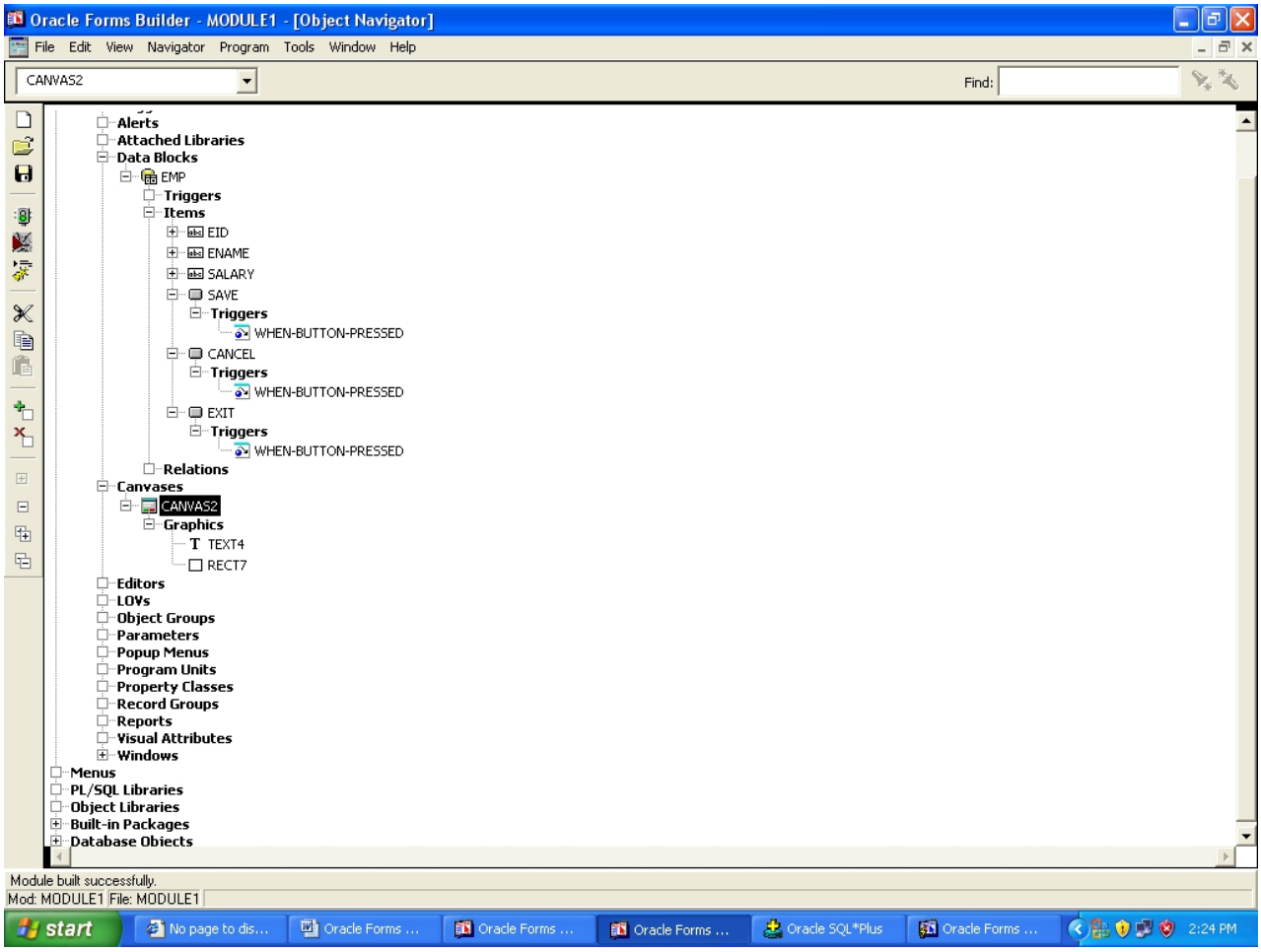
Exit the FORM Runtime. If you have not committed any transaction, you will be prompted to save changes. Click “YES” to save changes.
Click “OK” for acknowledgement.

Don’t forget to save the Form.

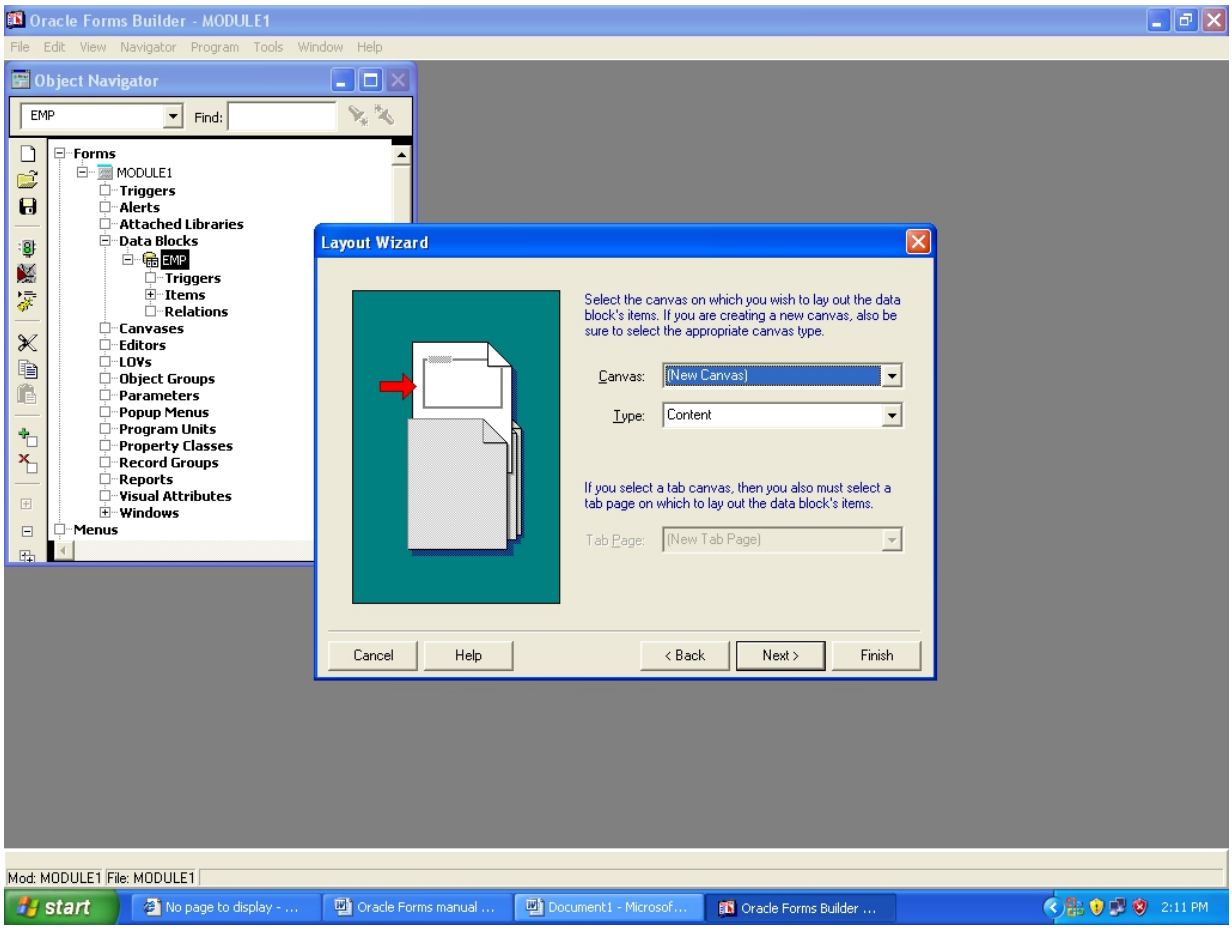
RABAD



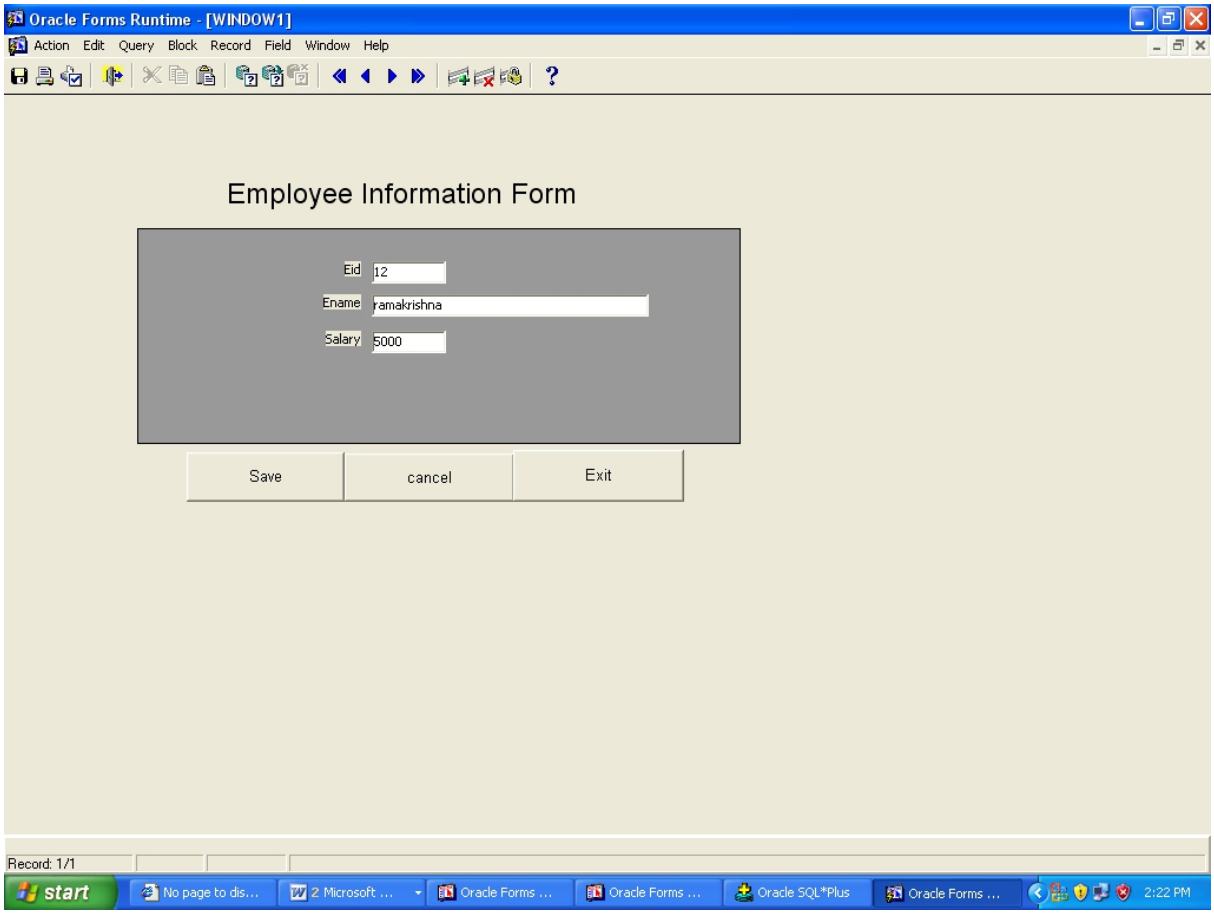
Selecting the type of form to create



Object wizard



Selecting the canvas on which data block can be displayed



Form showing the Employee details

EXPT#28. Generating REPORTS using Oracle Developer 2000

AIM: To design reports using Oracle Developer 2000

Introduction

Tabular report shows data in a table format. It is similar in concept to the idea of an Oracle table. Oracle, by default, returns output from your select statement in tabular format.

Hands-on

In this Hands-On, your client is a stock broker that keeps track of its customer stock transactions. You have been assigned to write the reports based on their reports layout requirements.
Your client wants you to create a simple listing report to show list of the stock trades by using stocks table for their brokerage company

Your tasks are:
1- Write a tabular report.
2- Apply user layout Format mask.
3- Run the report.
4- Test the repot.

You will learn how to: use report wizard, object navigator, report builder, “date model”, property palette, work on query and group box, see report style, use tabular style, navigating through report’s record, change the format mask for dollar, numeric and date items.

Open Report Builder tool

Open the "Report Builder" tool.

Connect to database

In the Object Navigator, highlight "Database Objects,” choose "File," then select the "Connect" option.
In the ‘Connect’ window, login as “iself” password schooling, then click “CONNECT.”

Save a report

In the Object Navigator, highlight the "untitled" report, choose “File,” and select the “Save as” option.
In the ‘Save as’ window, make sure to save the report in the ISELF folder and name it "rpt01_stock_history,” report number 1 stock history.

Data Model

In the Object Navigator, double click on the "Data Model" icon.

Create SQL box

In the Data Model window, click on the "SQL Query" icon. Then drag the plus sign cursor and click it anywhere in the “Data Model” screen where you wish your object to be.

In the ‘SQL Query Statement’ window, write a query to read all the stocks record sorted by their symbol.
(SQL Query Statement)
SELECT * FROM stocks
ORDER BY symbol
Click “OK.”

Change SQL box’s name
In the Data Model window, in the “SQL” box, right click on the ‘Q_1’ and open its property palette.
In its property palette, change the name to Q_STOCKS. Then close the window.

Change GROUP box’s name
In the Data Model, right click on the group box (G_SYMBOL) and open its property palette.
In the Group property palette, change the name to ‘G_STOCKS,’ and close the window.

Open Report Wizard
In the Data Model, click on the ‘Report Wizard’ icon on the horizontal tool bar.
In the Style tab, on the Report Wizard window, type ‘Stock History’ in the Title box and choose the report style as ‘Tabular.’
Notice that when you change the report style a layout of that report will be displayed on the screen.
Choose a different style to display its layout of its report style.

Data, Fields, Totals, Labels and Template tabs
Click “NEXT” to go to the Data tab. In the ‘SQL Query Statement’ verify your query.
Click “NEXT” to navigate to the Fields tab, select the fields that you would like to be display in your report. Select all the columns to be display.
Click “NEXT” to navigate to Totals tab, select the fields for which you would like to calculate totals. We have none in this hands-on exercise.

Click “NEXT” to open the Labels tab, modify the labels and widths for your fields and totals as desired.
Click “NEXT” again to go to the Template tab, and choose a template for your report. Your report will inherit the template’s colors, fonts, line widths, and structure.
Use the default template and click “finish.”

Running a report
Now, you should have your output report on the screen.

Resize an object
Maximize the output report and format the report layout. To resize an object , select it and drag its handler to the preferred size.

Move an object
To move an object, select and drag it while the cursor is on the object.

This is a simple report.

Navigate through the output

To navigate through the output report in the Report Editor - Live Pre-viewer, click on the "next page" or "previous page" icon on the horizontal toolbar.
Do the same with the "first page" or "last page" icon.

Use the “zoom in” and “zoom out” icon to preview the report.

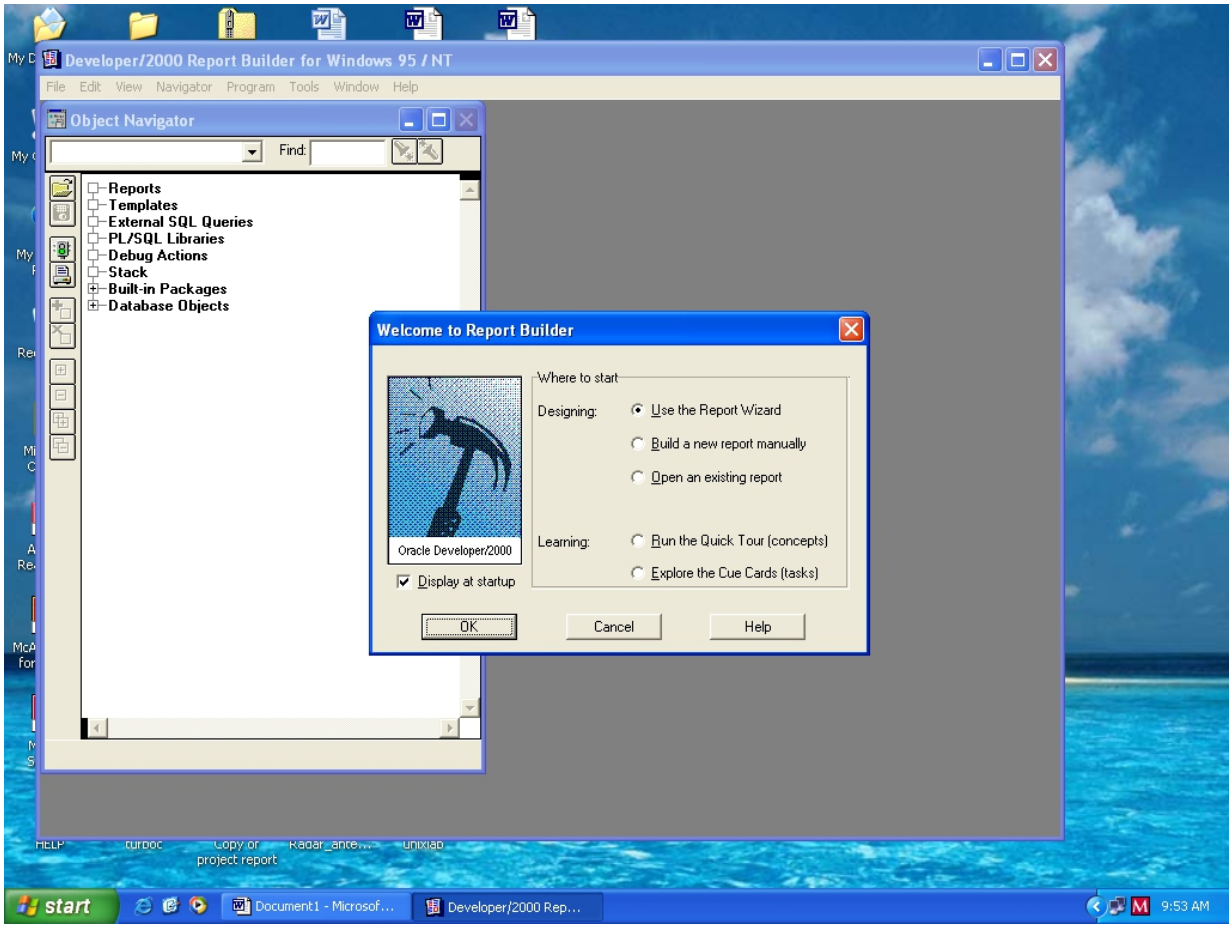
Know report’s functions

To know each icon functionalities, drag your cursor on it and a tooltip will display its function.

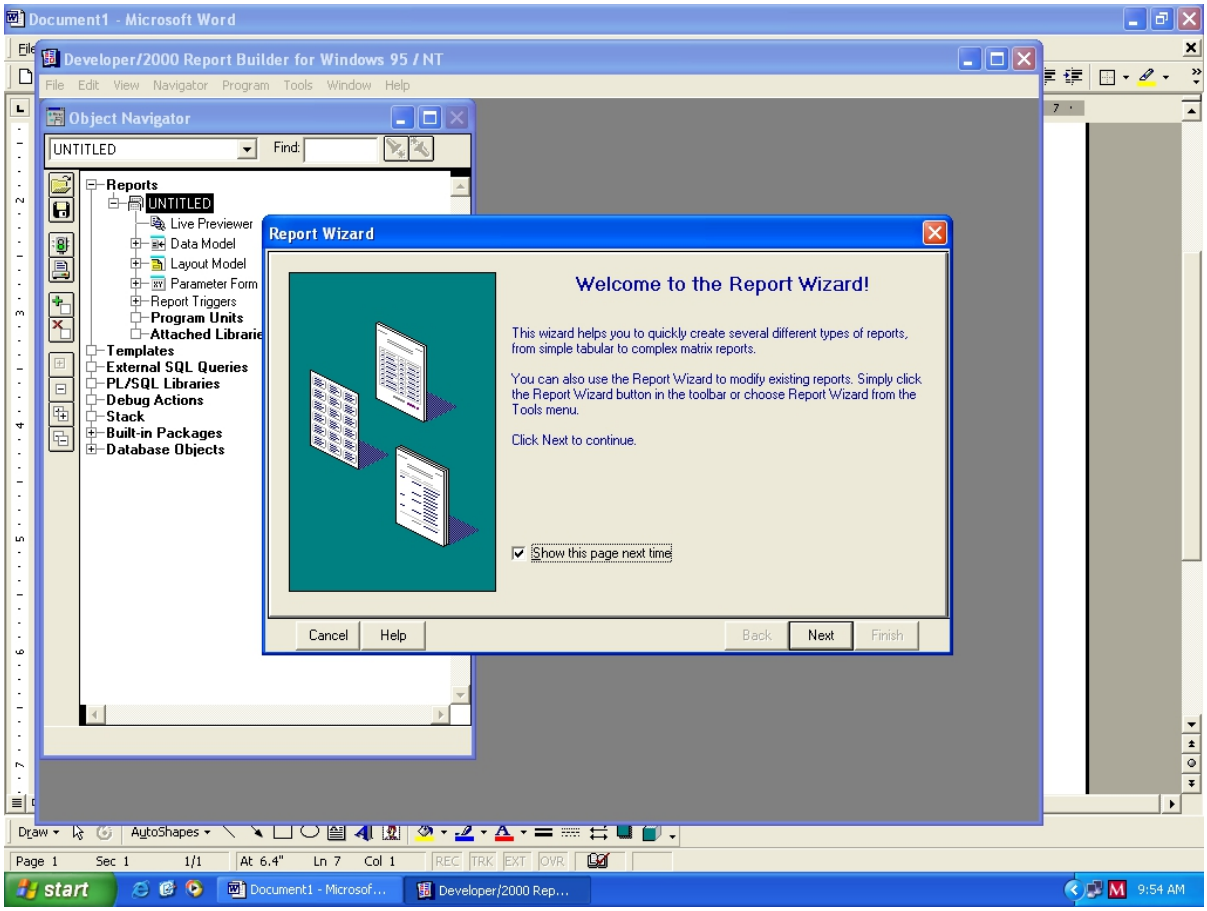
Change Format Mask

To change the "format mask" of a column, the column should be selected. Then go to the toolbar and click on the “$” icon, "add decimal place," and the “right justify” format to the all currency columns (Todays Low, Todays High, and current price)

Select the “traded today” column, and click on the ‘,0’ icon (apply commas), and make it right justify.
Also, you can change any attributes of field by opening its property palette. To open an object’s property palette, right click on it and select the Property Palette option.
Right click on the "trade date" column and open its "property palette."
Change the date "Format Mask" property and make it “year 2000 complaint (MM-DD-RR).”

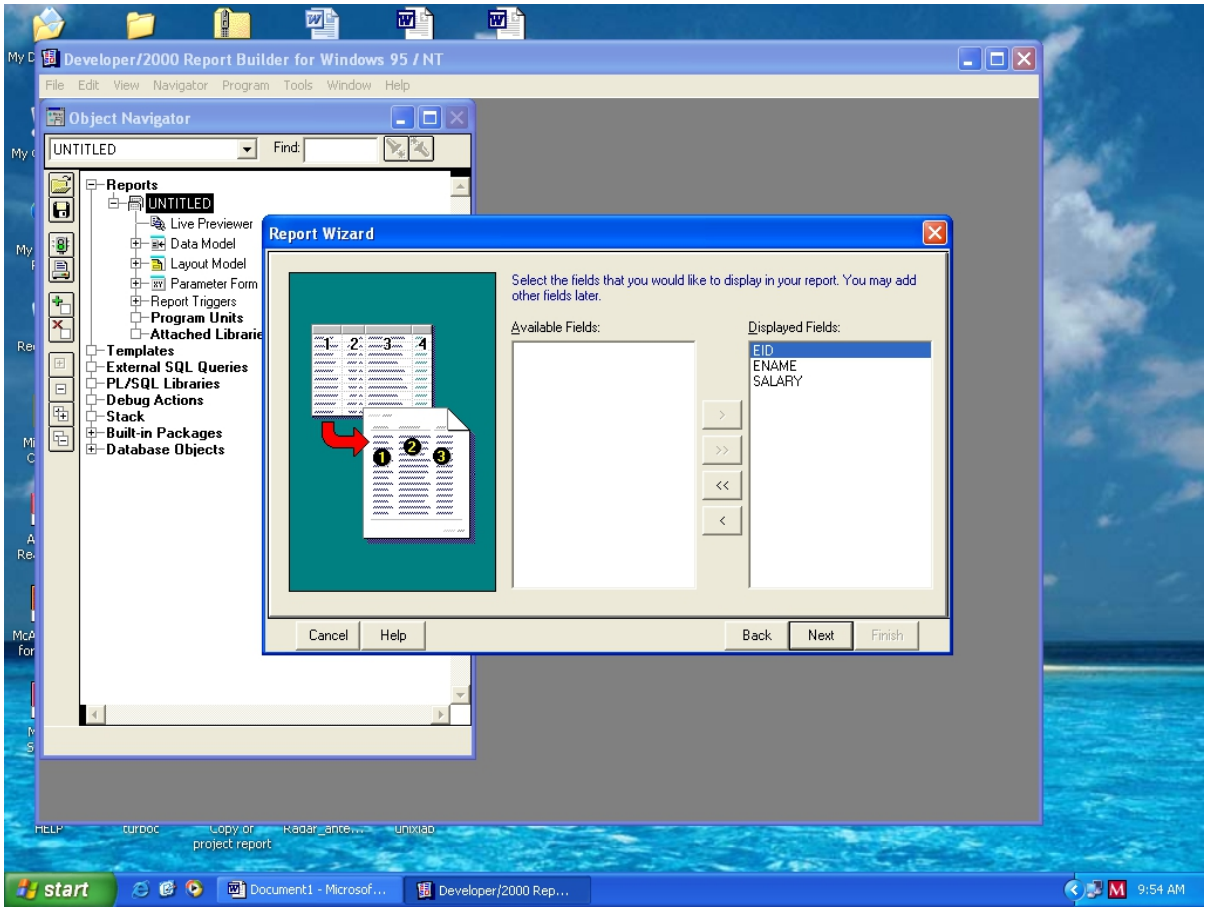


Selecting type of report

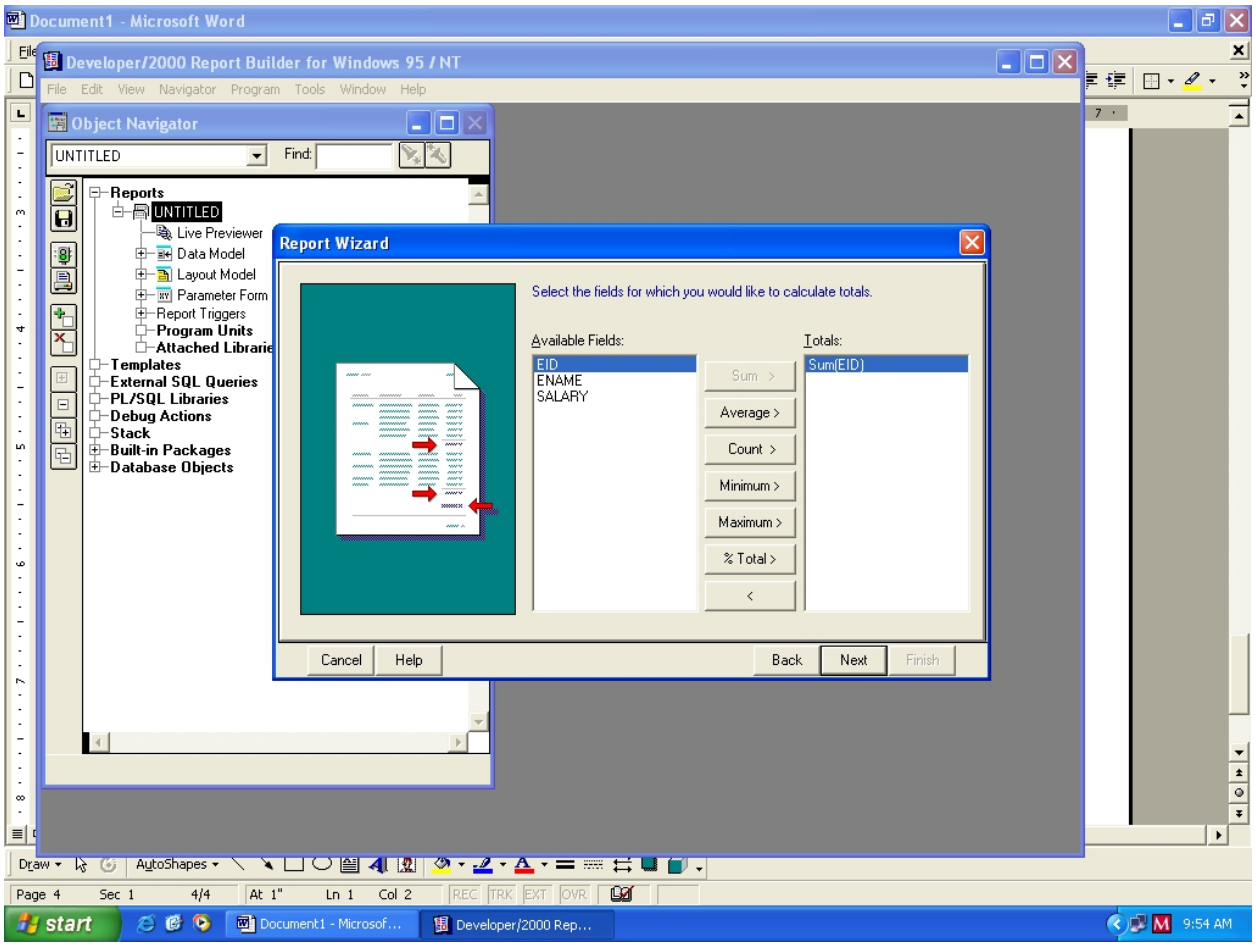


Creating reports

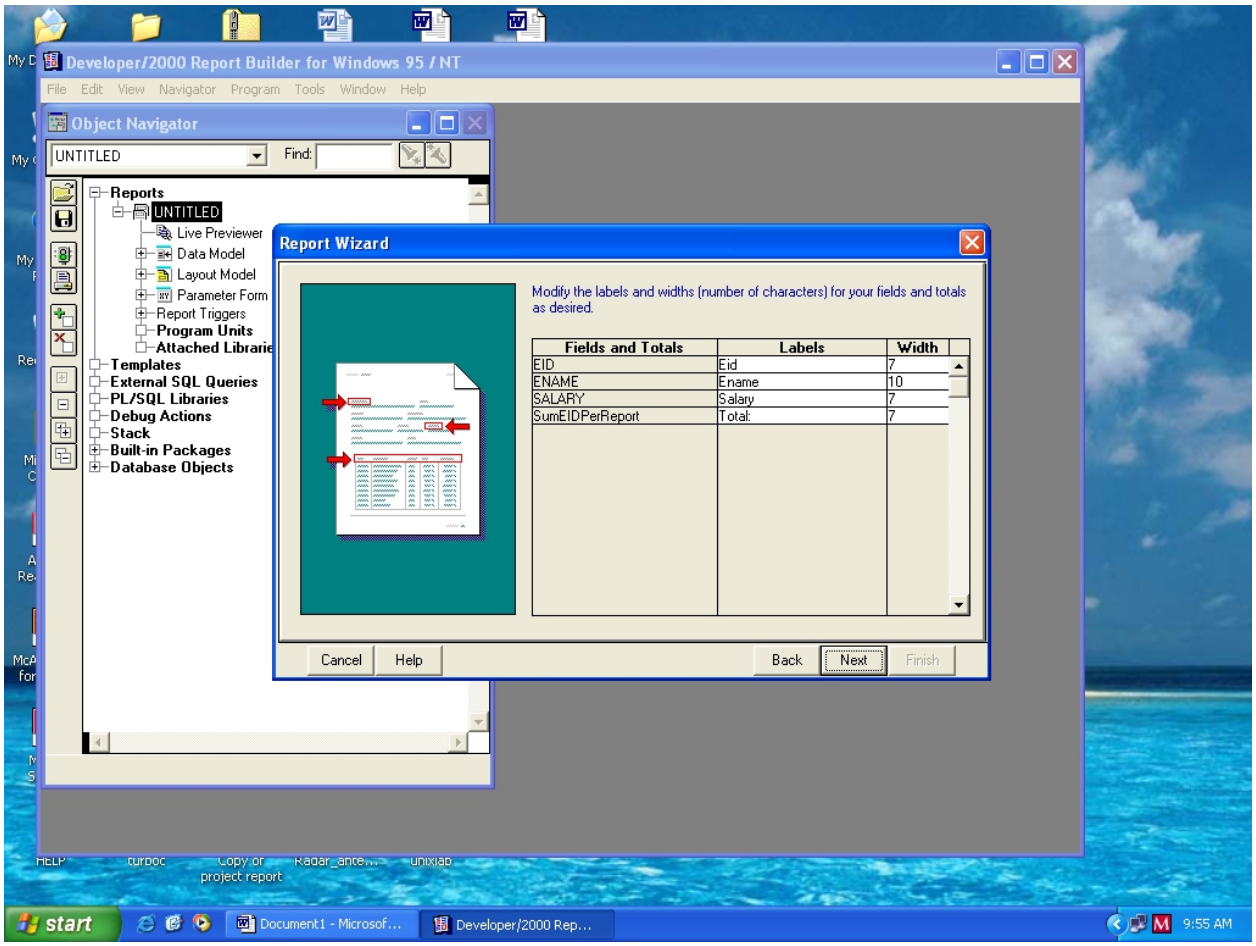
Selecting the format of reports



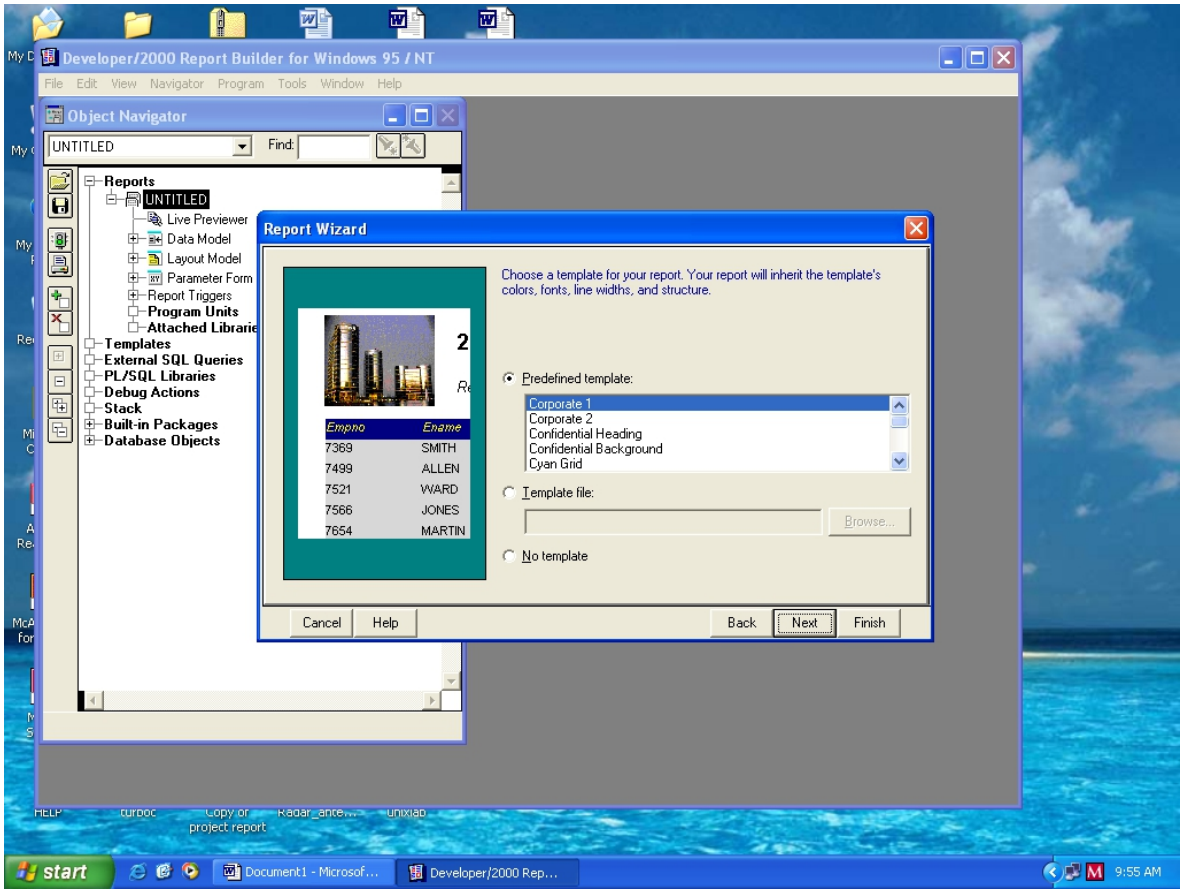
Selecting the Table in database



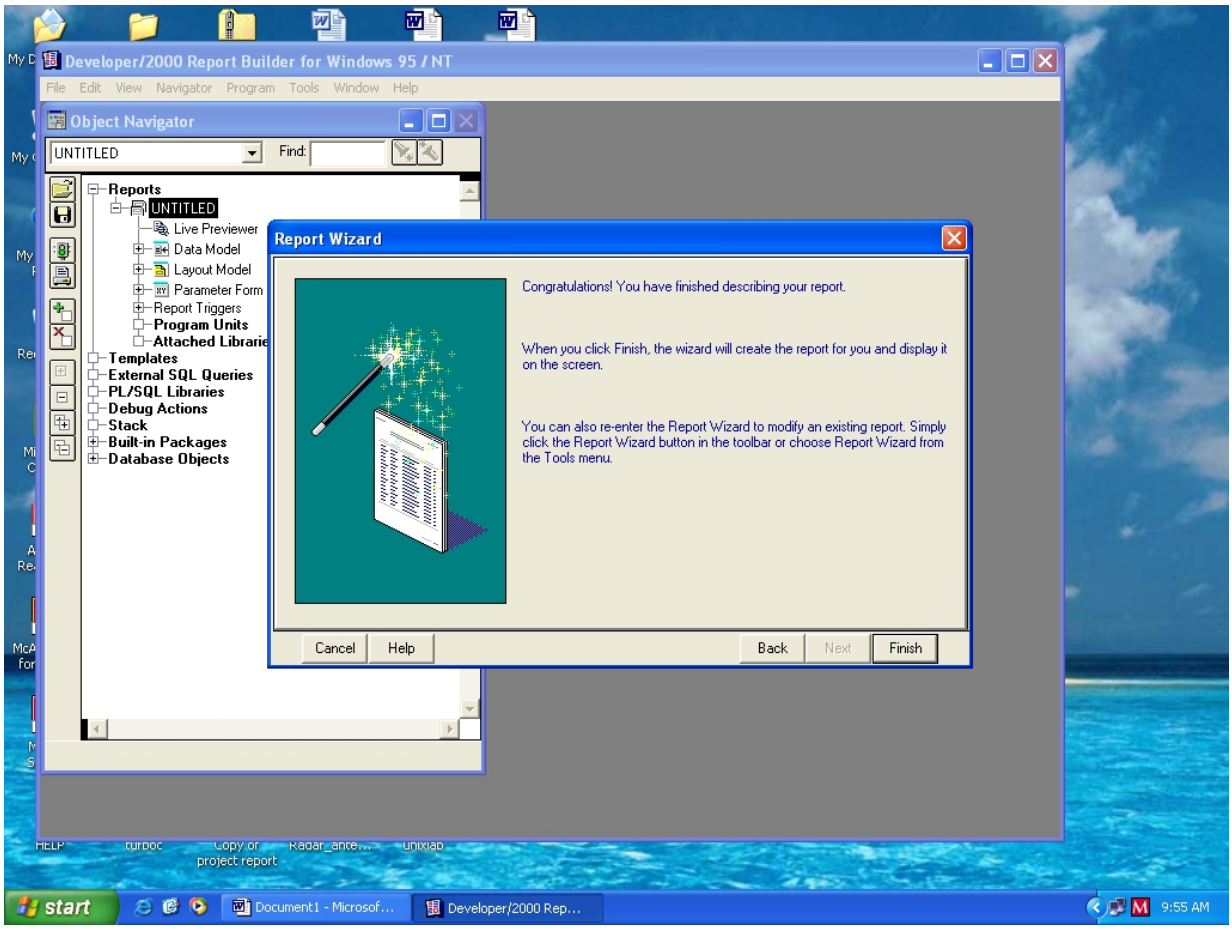
Selecting the columns in the report



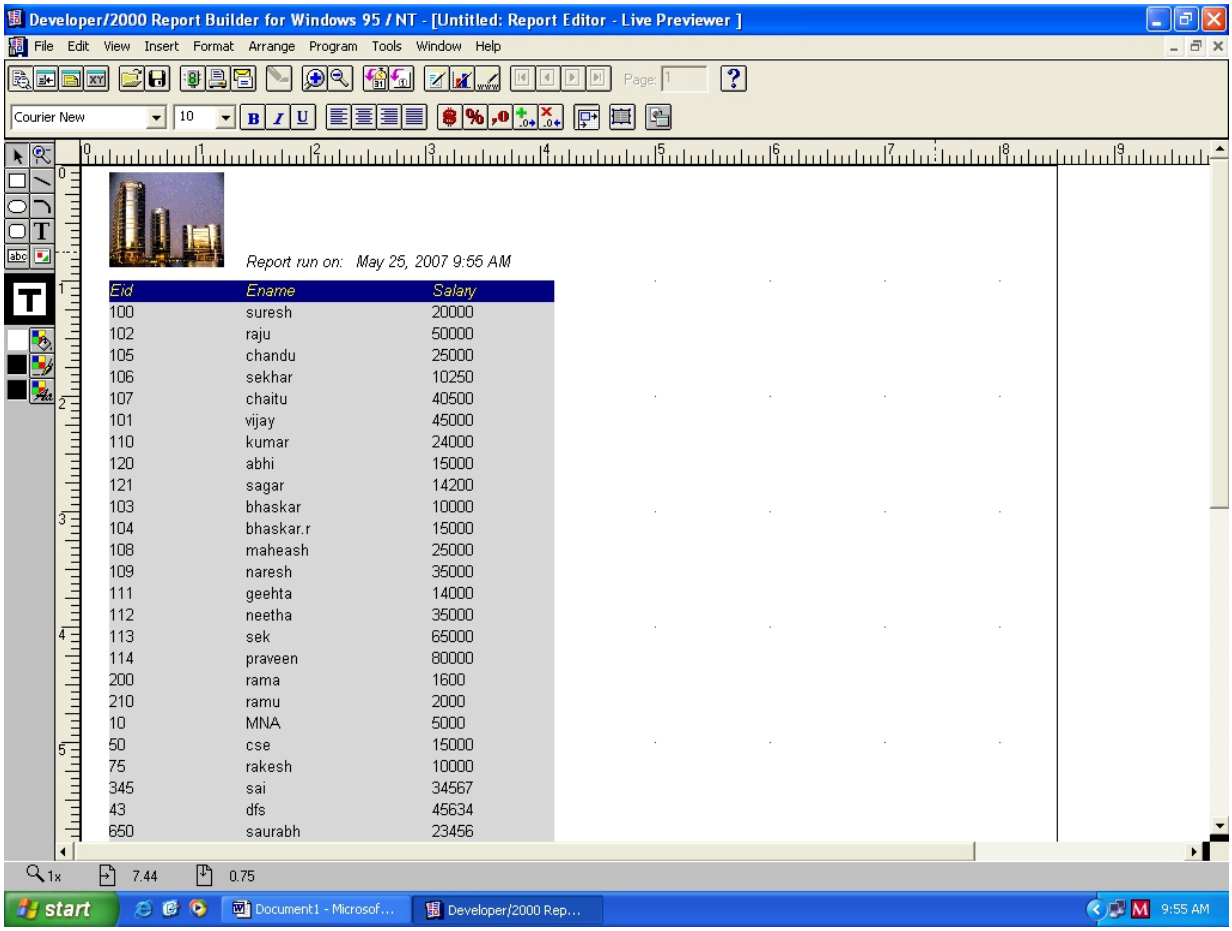
Modify the labels in a table



Choose a template to represent the report



To specify the completion of report generation


Developer/2000 Report Builder for Windows 95 / NT - [Untitled: Report Editor - Live Previewer]
File Edit View Insert Format Arrange Program Tools Window Help
Courier New
10
Report run on: May 25, 2007 9:55 AM
Eid Ename Salary
100 suresh 20000
102 raju 50000
105 chandu 25000
106 sekhar 10250
107 chaitu 40500
101 vijay 45000
110 kumar 24000
120 abhi 15000
121 sagar 14200
103 bhaskar 10000
104 bhaskar.r 15000
108 maheash 25000
109 naresh 35000
111 geehta 14000
112 neetha 35000
113 sek 65000
114 praveen 80000
200 rama 1600
210 ramu 2000
10 MNA 5000
50 cse 15000
75 rakesh 10000
345 sai 34567
43 dfs 45634
650 saurabh 23456
1x
7.44
0.75
start
Document1 - Microsof...
Developer/2000 Rep...
9:55 AM

\

EXPT#29. Providing Security using GRANT and REVOKE.
AIM: To learn GRANT and REVOKE commands to restrict privileges.

(1) GRANT Statement
Grant privileges to a user (or to a user role)

Syntax:

Grant System-wide Privs:

GRANT *system_priv(s)* TO *grantee*
[IDENTIFIED BY *password*] [WITH ADMIN OPTION]

GRANT *role* TO *grantee*
[IDENTIFIED BY *password*] [WITH ADMIN OPTION]

GRANT ALL PRIVILEGES TO *grantee*
[IDENTIFIED BY *password*] [WITH ADMIN OPTION]

Grant privs on specific objects:

GRANT *object_priv* [(*column*, *column*,...)]
ON [schema.]object
TO *grantee* [WITH GRANT OPTION] [WITH HIERARCHY OPTION]

GRANT **ALL PRIVILEGES** [(column, column,...)]
ON [schema.]object
TO *grantee* [WITH GRANT OPTION] [WITH HIERARCHY OPTION]

GRANT *object_priv* [(column, column,...)]
ON DIRECTORY directory_name
TO *grantee* [WITH GRANT OPTION] [WITH HIERARCHY OPTION]

GRANT *object_priv* [(column, column,...)]
ON JAVA [RE]SOURCE [schema.]object
TO *grantee* [WITH GRANT OPTION] [WITH HIERARCHY OPTION]

grantee:
user
role
PUBLIC

system_privs:
CREATE SESSION - Allows user to connect to the database
UNLIMITED TABLESPACE - Use an unlimited amount of any tablespace.
SELECT ANY TABLE - Query tables, views, or mviews in any schema

UPDATE ANY TABLE - Update rows in tables and views in any schema
INSERT ANY TABLE - Insert rows into tables and views in any schema
Also System Admin rights to CREATE, ALTER or DROP:
cluster, context, database, link, dimension, directory, index,
materialized view, operator, outline, procedure, profile, role,
rollback segment, sequence, session, synonym, table, tablespace,
trigger, type, user, view. (full list of system privs)

object_privs:
SELECT, UPDATE, INSERT, DELETE, ALTER, DEBUG, EXECUTE, INDEX, REFERENCES

roles:
SYSDBA, SYSOPER, OSDBA, OSOPER, EXP_FULL_DATABASE, IMP_FULL_DATABASE
SELECT_CATALOG_ROLE, EXECUTE_CATALOG_ROLE, DELETE_CATALOG_ROLE
AQ_USER_ROLE, AQ_ADMINISTRATOR_ROLE - advanced queuing
SNMPAGENT - Enterprise Manager/Intelligent Agent.
RECOVERY_CATALOG_OWNER - rman
HS_ADMIN_ROLE - heterogeneous services

plus any user defined roles you have available

Notes:

Several Object_Privs can be assigned in a single GRANT statement
e.g.
GRANT SELECT (empno), UPDATE (sal) ON scott.emp TO emma

WITH HIERARCHY OPTION will grant the object privilege on all subobjects, including any created after the GRANT statement is issued.

WITH GRANT OPTION will enable the grantee to grant those object privileges to other users and roles.

"GRANT ALL PRIVILEGES..." may also be written as "GRANT ALL..."

(ii) REVOKE Statement
Revoke privileges from users or roles.

Syntax:

Roles:
REVOKE *role* FROM {*user*, | *role*, |PUBLIC}

System Privs:
  REVOKE *system_priv(s)* FROM {*user*, | *role*, |PUBLIC}

  REVOKE ALL FROM {*user*, | *role*, |PUBLIC}

Object Privs:
  REVOKE *object_priv* [(column1, column2..)] ON [schema.]object
      FROM {*user*, | role, |PUBLIC} [CASCADE CONSTRAINTS] [FORCE]

  REVOKE *object_priv* [(column1, column2..)] ON [schema.]object
      FROM {*user*, | *role*, |PUBLIC} [CASCADE CONSTRAINTS] [FORCE]

  REVOKE *object_priv* [(column1, column2..)] ON DIRECTORY directory_name
      FROM {*user*, | *role*, |PUBLIC} [CASCADE CONSTRAINTS] [FORCE]

  REVOKE *object_priv* [(column1, column2..)] ON JAVA [RE]SOURCE [schema.]object
      FROM {*user*, | *role*, |PUBLIC} [CASCADE CONSTRAINTS] [FORCE]

key:
  *object_priv*s
  ALTER, DELETE, EXECUTE, INDEX, INSERT,
  REFERENCES, SELECT, UPDATE, ALL PRIVILEGES

  system_privs
  ALTER ANY INDEX, BECOME USER, CREATE TABLE, DROP ANY VIEW
  RESTRICTED SESSION, UNLIMITED TABLESPACE, UPDATE ANY TABLE
  plus too many others to list here

  roles
  Standard Oracle roles -
  SYSDBA, SYSOPER, OSDBA, OSOPER, EXP_FULL_DATABASE,
IMP_FULL_DATABASE
  plus any user defined roles you have available

FORCE, will revoke all privileges from a user-defined-type and mark it's dependent objects INVALID.

The roles CONNECT, RESOURCE and DBA are now deprecated (supported only for backwards compatibility) unless you are still running Oracle 6.0

Error ORA-01927 "cannot REVOKE privileges you did not grant" - This usually means you tried revoking permission from the table owner, e.g.
Oracle will not allow REVOKE select on USER1.Table1 from USER1 Owners of objects ALWAYS have full permissions on those objects. This is one reason it makes sense to place tables in one schema and the packaged prodecures used to access those tables in a separate schema.

REFERENCES:

1. Oracle 9i Release 2 (9.2) SQL Reference, www.cs.ncl.ac.uk/teaching/facilities/swdoc/oracle9i/server.920/a96540/toc.htm.

2. Oracle 9i Release 1 (9.0.1) SQL Reference, http://download-east.oracle.com/docs/cd/A91202_01/901_doc/server.901/a90125/toc.htm.

3. An A-Z Index of Oracle SQL Commands (version 9.2) http://www.ss64.com/ora/.

4. Database Systems Instructor: Prof. Samuel Madden Source: MIT Open Courseware (http://ocw.mit.edu).

5. RDBMS Lab Guide, www.campusconnect.infosys.com userid:demo@infosys and passwork:infosys.

6. Orelly PL/SQL Pocket Reference, http://www.unix.org.ua/orelly/oracle/langpkt/index.htm

7. **PL/SQL User's Guide and Reference, Release 2 (9.2), http://www.lc.leidenuniv.nl/awcourse/oracle/appdev.920/a96624/toc.htm.**

VIVA VOICE QUESTIONS AND ANSWERS

1. What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

2. What is DBMS?
It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

3. What is a Database system?
The database and **DBMS** software together is called as Database system.

4. Advantages of DBMS?
Ø Redundancy is controlled.
Ø Unauthorised access is restricted.
Ø Providing multiple user interfaces.
Ø Enforcing integrity constraints.
Ø Providing backup and recovery.

5. Disadvantage in File Processing System?
Ø Data redundancy & inconsistency.
Ø Difficult in accessing data.
Ø Data isolation.
Ø Data integrity.
Ø Concurrent access is not possible.
Ø Security Problems.

6. Describe the three levels of data abstraction?
The are three levels of abstraction:
Ø Physical level: The lowest level of abstraction describes how data are stored.
Ø Logical level: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
Ø View level: The highest level of abstraction describes only part of entire database.

7. Define the "integrity rules"
There are two Integrity rules.
Ø Entity Integrity: States that “Primary key cannot have NULL value”
Ø Referential Integrity: States that “Foreign Key can be either a NULL value or should be Primary Key value of other relation.

8. What is extension and intension?
Extension - It is the number of tuples present in a table at any instance.This is time dependent.
Intension - It is a constant value that gives the name, structure of table and the constraints laid on it.

9. What is System R? What are its two major subsystems?
System R was designed and developed over a period of 1974-79 at IBM San Jose Research Center. It is a prototype and its purpose was to demonstrate that it is possible to build a Relational System that can be used in a real life environment to solve real life problems, with performance at least comparable to that of existing system.
Its two subsystems are
Ø Research Storage
Ø System Relational Data System.

10. How is the data structure of System R different from the relational structure?
Unlike Relational systems in System R
Ø Domains are not supported
Ø Enforcement of candidate key uniqueness is optional
Ø Enforcement of entity integrity is optional
Ø Referential integrity is not enforced

11. What is Data Independence?
Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.
Two types of Data Independence:
Ø Physical Data Independence: Modification in physical level should not affect the logical level.
Ø Logical Data Independence: Modification in logical level should affect the view level.
NOTE: Logical Data Independence is more difficult to achieve

12. What is a view? How it is related to data independence?
A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.
Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

13. What is Data Model?
A collection of conceptual tools for describing data, data relationships data semantics and constraints.

14. What is E-R model?
This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

15. What is Object Oriented model?
This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object.

These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

16. What is an Entity?
It is a 'thing' in the real world with an independent existence.

17. What is an Entity type?
It is a collection (set) of entities that have same attributes.

18. What is an Entity set?
It is a collection of all entities of particular entity type in the database.

19. What is an Extension of entity type?
The collections of entities of a particular entity type are grouped together into an entity set.

20. What is Weak Entity set?
An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

21. What is an attribute?
It is a particular property, which describes the entity.

22. What is a Relation Schema and a Relation?
A relation Schema denoted by R(A1, A2, …, An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ..., tn). Each tuple is an ordered list of n-values t=(v1,v2, ..., vn).

23. What is degree of a Relation?
It is the number of attribute of its relation schema.

24. What is Relationship?
It is an association among two or more entities.

25. What is Relationship set?
The collection (or set) of similar relationships.

26. What is Relationship type?
Relationship type defines a set of associations or a relationship set among a given set of entity types.

27. What is degree of Relationship type?
It is the number of entity type participating.

28. What is DDL (Data Definition Language)?
A data base schema is specifies by a set of definitions expressed by a special language called DDL.

29. What is VDL (View Definition Language)?
It specifies user views and their mappings to the conceptual schema.

30. What is SDL (Storage Definition Language)?
This language is to specify the internal schema. This language may specify the mapping between two schemas.

31. What is Data Storage - Definition Language?
The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage-definition language.

32. What is DML (Data Manipulation Language)?
This language that enable user to access or manipulate data as organised by appropriate data model.
Ø Procedural DML or Low level: DML requires a user to specify what data are needed and how to get those data.
Ø Non-Procedural DML or High level: DML requires a user to specify what data are needed without specifying how to get those data.

33. What is DML Compiler?
It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

34. What is Query evaluation engine?
It executes low-level instruction generated by compiler.

35. What is DDL Interpreter?
It interprets DDL statements and record them in tables containing metadata.

36. What is Record-at-a-time?
The Low level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.

37. What is Set-at-a-time or Set-oriented?
The High level or Non-procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be Set-at-a-time or Set-oriented.

38. What is Relational Algebra?
It is procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

39. What is Relational Calculus?
It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL ALPHA, QUEL.

40. How does Tuple-oriented relational calculus differ from domain-oriented relational calculus

The tuple-oriented calculus uses a tuple variables i.e., variable whose only permitted values are tuples of that relation. E.g. QUEL

The domain-oriented calculus has domain variables i.e., variables that range over the underlying domains instead of over relation. E.g. ILL, DEDUCE.

41. What is normalization?

It is a process of analysing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

Ø Minimizing redundancy

Ø Minimizing insertion, deletion and update anomalies.

42. What is Functional Dependency?

A Functional dependency is denoted by X Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y]. This means the value of X component of a tuple uniquely determines the value of component Y.

43. When is a functional dependency F said to be minimal?

Ø Every dependency in F has a single attribute for its right hand side.

Ø We cannot replace any dependency X A in F with a dependency Y A where Y is a proper subset of X and still have a set of dependency that is equivalent to F.

Ø We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

44. What is Multivalued dependency?

Multivalued dependency denoted by X Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties

Ø t3[x] = t4[X] = t1[X] = t2[X]

Ø t3[Y] = t1[Y] and t4[Y] = t2[Y]

Ø t3[Z] = t2[Z] and t4[Z] = t1[Z]

where [Z = (R-(X U Y))]

45. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

46. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

47. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency X Y is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

48. What is 2NF?
A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

49. What is 3NF?
A relation schema R is in 3NF if it is in 2NF and for every FD X A either of the following is true
Ø X is a Super-key of R.
Ø A is a prime attribute of R.
In other words, if every non prime attribute is non-transitively dependent on primary key.

50. What is BCNF (Boyce-Codd Normal Form)?
A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD X A, X must be a candidate key.

51. What is 4NF?
A relation schema R is said to be in 4NF if for every Multivalued dependency X Y that holds over R, one of following is true
Ø X is subset or equal to (or) XY = R.
Ø X is a super key.

52. What is 5NF?
A Relation schema R is said to be 5NF if for every join dependency {R1, R2, ..., Rn} that holds R, one the following is true
Ø Ri = R for some i.
Ø The join dependency is implied by the set of FD, over R in which the left side is key of R.

53. What is Domain-Key Normal Form?
A relation is said to be in DKNF if all constraints and dependencies that should hold on the the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

54. What are partial, alternate,, artificial, compound and natural key?
Partial Key: It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.
Alternate Key: All Candidate Keys excluding the Primary Key are known as Alternate Keys.
Artificial Key: If no obvious key, either stand alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.
Compound Key: If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.
Natural Key: When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

55. What is indexing and what are the different kinds of indexing?
Indexing is a technique for determining how quickly specific data can be found.
Types:
Ø Binary search style indexing
Ø B-Tree indexing
Ø Inverted list indexing
Ø Memory resident table
Ø Table indexing

56. What is system catalog or catalog relation? How is better known as?
A RDBMS maintains a description of all the data that it contains, information about every relation and index that it contains. This information is stored in a collection of relations maintained by the system called metadata. It is also called data dictionary.

57. What is meant by query optimization?
The phase that identifies an efficient execution plan for evaluating a query that has the least estimated cost is referred to as query optimization.

58. What is join dependency and inclusion dependency?
Join Dependency:
A Join dependency is generalization of Multivalued dependency.A JD {R1, R2, ..., Rn} is said to hold over a relation R if R1, R2, R3, ..., Rn is a lossless-join decomposition of R .
There is no set of sound and complete inference rules for JD.
Inclusion Dependency:
An Inclusion Dependency is a statement of the form that some columns of a relation are contained in other columns. A foreign key constraint is an example of inclusion dependency.

59. What is durability in DBMS?
Once the **DBMS** informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

60. What do you mean by atomicity and aggregation?
Atomicity:
Either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions. **DBMS** ensures this by undoing the actions of incomplete transactions.
Aggregation:
A concept which is used to model a relationship between a collection of entities and relationships. It is used when we need to express a relationship among relationships.

61. What is a Phantom Deadlock?
In distributed deadlock detection, the delay in propagating local information might cause the deadlock detection algorithms to identify deadlocks that do not really exist. Such situations are called phantom deadlocks and they lead to unnecessary aborts.

62. What is a checkpoint and When does it occur?
A Checkpoint is like a snapshot of the **DBMS** state. By taking checkpoints, the **DBMS** can reduce the amount of work to be done during restart in the event of subsequent crashes.

63. What are the different phases of transaction?
Different phases are
Ø Analysis phase
Ø Redo Phase
Ø Undo phase

64. What do you mean by flat file database?
It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.

65. What is "transparent DBMS"?
It is one, which keeps its Physical Structure hidden from user.

66. Brief theory of Network, Hierarchical schemas and their properties
Network schema uses a graph data structure to organize records example for such a database management system is CTCG while a hierarchical schema uses a tree data structure example for such a system is IMS.

67. What is a query?
A query with respect to **DBMS** relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

68. What do you mean by Correlated subquery?
Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is called a correlated subquery.
A correlated subquery can be easily identified if it contains any references to the parent subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated subquery.
E.g. Select * From CUST Where '10/03/1990' IN (Select ODATE From ORDER Where CUST.CNUM = ORDER.CNUM)

69. What are the primitive operations common to all record management systems?
Addition, deletion and modification.

70. Name the buffer in which all the commands that are typed in are stored
'Edit' Buffer

71. What are the unary operations in Relational Algebra?
PROJECTION and SELECTION.

72. Are the resulting relations of PRODUCT and JOIN operation the same?
No.
PRODUCT: Concatenation of every row in one relation with every row in another.
JOIN: Concatenation of rows from one relation and related rows from another.

73. What is RDBMS KERNEL?
Two important pieces of RDBMS architecture are the kernel, which is the software, and the data dictionary, which consists of the system-level data structures used by the kernel to manage the database
You might think of an RDBMS as an operating system (or set of subsystems), designed specifically for controlling data access; its primary functions are storing, retrieving, and securing data. An RDBMS maintains its own list of authorized users and their associated privileges; manages memory caches and paging; controls locking for concurrent resource usage; dispatches and schedules user requests; and manages space usage within its table-space structures.

74. Name the sub-systems of a RDBMS
I/O, Security, Language Processing, Process Control, Storage Management, Logging and Recovery, Distribution Control, Transaction Control, Memory Management, Lock Management

75. Which part of the RDBMS takes care of the data dictionary? How
Data dictionary is a set of tables and database objects that is stored in a special area of the database and maintained exclusively by the kernel.

76. What is the job of the information stored in data-dictionary?
The information in the data dictionary validates the existence of the objects, provides access to them, and maps the actual physical storage location.

77. Not only RDBMS takes care of locating data it also
determines an optimal access path to store or retrieve the data

76. How do you communicate with an RDBMS?
You communicate with an RDBMS using Structured Query Language (SQL)

78. Define SQL and state the differences between SQL and other conventional programming Languages
SQL is a nonprocedural language that is designed specifically for data access operations on normalized relational database structures. The primary difference between SQL and other conventional programming languages is that SQL statements specify what data operations should be performed rather than how to perform them.

79. Name the three major set of files on disk that compose a database in Oracle
There are three major sets of files on disk that compose a database. All the files are binary. These are
Ø Database files

Ø Control files
Ø Redo logs
The most important of these are the database files where the actual data resides. The control files and the redo logs support the functioning of the architecture itself.
All three sets of files must be present, open, and available to Oracle for any data on the database to be useable. Without these files, you cannot access the database, and the database administrator might have to recover some or all of the database using a backup, if there is one.

80. What is an Oracle Instance?
The Oracle system processes, also known as Oracle background processes, provide functions for the user processes—functions that would otherwise be done by the user processes themselves
Oracle database-wide system memory is known as the SGA, the system global area or shared global area. The data and control structures in the SGA are shareable, and all the Oracle background processes and user processes can use them.
The combination of the SGA and the Oracle background processes is known as an Oracle instance

81. What are the four Oracle system processes that must always be up and running for the database to be useable
The four Oracle system processes that must always be up and running for the database to be useable include DBWR (Database Writer), LGWR (Log Writer), SMON (System Monitor), and PMON (Process Monitor).

82. What are database files, control files and log files. How many of these files should a database have at least? Why?
Database Files
The database files hold the actual data and are typically the largest in size. Depending on their sizes, the tables (and other objects) for all the user accounts can go in one database file—but that's not an ideal situation because it does not make the database structure very flexible for controlling access to storage for different users, putting the database on different disk drives, or backing up and restoring just part of the database.
You must have at least one database file but usually, more than one files are used. In terms of accessing and using the data in the tables and other objects, the number (or location) of the files is immaterial.
The database files are fixed in size and never grow bigger than the size at which they were created Control Files
The control files and redo logs support the rest of the architecture. Any database must have at least one control file, although you typically have more than one to guard against loss. The control file records the name of the database, the date and time it was created, the location of the database and redo logs, and the synchronization information to ensure that all three sets of files are always in step. Every time you add a new database or redo log file to the database, the information is recorded in the control files.
Redo Logs
Any database must have at least two redo logs. These are the journals for the database; the redo logs record all changes to the user objects or system objects. If any type of failure occurs, the changes recorded in the redo logs can be used to bring the database to a consistent

state without losing any committed transactions. In the case of non-data loss failure, Oracle can apply the information in the redo logs automatically without intervention from the DBA. The redo log files are fixed in size and never grow dynamically from the size at which they were created.

83. What is ROWID?

The ROWID is a unique database-wide physical address for every row on every table. Once assigned (when the row is first inserted into the database), it never changes until the row is deleted or the table is dropped.

The ROWID consists of the following three components, the combination of which uniquely identifies the physical storage location of the row.

Ø Oracle database file number, which contains the block with the rows

Ø Oracle block address, which contains the row

Ø The row within the block (because each block can hold many rows)

The ROWID is used internally in indexes as a quick means of retrieving rows with a particular key value. Application developers also use it in SQL statements as a quick way to access a row once they know the ROWID

84. What is Oracle Block? Can two Oracle Blocks have the same address?

Oracle "formats" the database files into a number of Oracle blocks when they are first created—making it easier for the RDBMS software to manage the files and easier to read data into the memory areas.

The block size should be a multiple of the operating system block size. Regardless of the block size, the entire block is not available for holding data; Oracle takes up some space to manage the contents of the block. This block header has a minimum size, but it can grow. These Oracle blocks are the smallest unit of storage. Increasing the Oracle block size can improve performance, but it should be done only when the database is first created.

Each Oracle block is numbered sequentially for each database file starting at 1. Two blocks can have the same block address if they are in different database files.

85. What is database Trigger?

A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can e defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written in PL/SQL.

86. Name two utilities that Oracle provides, which are use for backup and recovery.

Along with the RDBMS software, Oracle provides two utilities that you can use to back up and restore the database. These utilities are Export and Import.

The Export utility dumps the definitions and data for the specified part of the database to an operating system binary file. The Import utility reads the file produced by an export, recreates the definitions of objects, and inserts the data

If Export and Import are used as a means of backing up and recovering the database, all the changes made to the database cannot be recovered since the export was performed. The best you can do is recover the database to the time when the export was last performed.

87. What are stored-procedures? And what are the advantages of using them.
Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result to the client. Stored procedures are used to reduce network traffic.

88. How are exceptions handled in PL/SQL? Give some of the internal exceptions' name
PL/SQL exception handling is a mechanism for dealing with run-time errors encountered during procedure execution. Use of this mechanism enables execution to continue if the error is not severe enough to cause procedure termination.
The exception handler must be defined within a subprogram specification. Errors cause the program to raise an exception with a transfer of control to the exception-handler block. After the exception handler executes, control returns to the block in which the handler was defined. If there are no more executable statements in the block, control returns to the caller.
User-Defined Exceptions
PL/SQL enables the user to define exception handlers in the declarations area of subprogram specifications. User accomplishes this by naming an exception as in the following example:
ot_failure EXCEPTION;
In this case, the exception name is ot_failure. Code associated with this handler is written in the EXCEPTION specification area as follows:
EXCEPTION
when OT_FAILURE then
out_status_code := g_out_status_code;
out_msg := g_out_msg;
The following is an example of a subprogram exception:
EXCEPTION
when NO_DATA_FOUND then
g_out_status_code := 'FAIL';
RAISE ot_failure;
Within this exception is the RAISE statement that transfers control back to the ot_failure exception handler. This technique of raising the exception is used to invoke all user-defined exceptions.
System-Defined Exceptions
Exceptions internal to PL/SQL are raised automatically upon error. NO_DATA_FOUND is a system-defined exception. Table below gives a complete list of internal exceptions.

PL/SQL internal exceptions.

PL/SQL internal exceptions.

Exception Name Oracle Error
CURSOR_ALREADY_OPEN ORA-06511
DUP_VAL_ON_INDEX ORA-00001
INVALID_CURSOR ORA-01001
INVALID_NUMBER ORA-01722
LOGIN_DENIED ORA-01017

NO_DATA_FOUND ORA-01403
NOT_LOGGED_ON ORA-01012
PROGRAM_ERROR ORA-06501
STORAGE_ERROR ORA-06500
TIMEOUT_ON_RESOURCE ORA-00051
TOO_MANY_ROWS ORA-01422
TRANSACTION_BACKED_OUT ORA-00061
VALUE_ERROR ORA-06502
ZERO_DIVIDE ORA-01476

In addition to this list of exceptions, there is a catch-all exception named OTHERS that traps all errors for which specific error handling has not been established.

89. Does PL/SQL support "overloading"? Explain
The concept of overloading in PL/SQL relates to the idea that you can define procedures and functions with the same name. PL/SQL does not look only at the referenced name, however, to resolve a procedure or function call. The count and data types of formal parameters are also considered.
PL/SQL also attempts to resolve any procedure or function calls in locally defined packages before looking at globally defined packages or internal functions. To further ensure calling the proper procedure, you can use the dot notation. Prefacing a procedure or function name with the package name fully qualifies any procedure or function reference.

90. Tables derived from the ERD
a) Are totally unnormalised
b) Are always in 1NF
c) Can be further denormalised
d) May have multi-valued attributes

91. Spurious tuples may occur due to
i. Bad normalization
ii. Theta joins
iii. Updating tables from join
a) i & ii b) ii & iii
c) i & iii d) ii & iii

(a) i & iii because theta joins are joins made on keys that are not primary keys.

92. A B C is a set of attributes. The functional dependency is as follows
AB -> B
AC -> C
C -> B
a) is in 1NF
b) is in 2NF
c) is in 3NF
d) is in BCNF

(a) is in 1NF since (AC)+ = { A, B, C} hence AC is the primary key. Since C B is a FD given, where neither C is a Key nor B is a prime attribute, this it is not in 3NF. Further B is not functionally dependent on key AC thus it is not in 2NF. Thus the given FDs is in 1NF.

93. In mapping of ERD to DFD
a) entities in ERD should correspond to an existing entity/store in DFD
b) entity in DFD is converted to attributes of an entity in ERD
c) relations in ERD has 1 to 1 correspondence to processes in DFD
d) relationships in ERD has 1 to 1 correspondence to flows in DFD

(a) entities in ERD should correspond to an existing entity/store in DFD

94. A dominant entity is the entity
a) on the N side in a 1 : N relationship
b) on the 1 side in a 1 : N relationship
c) on either side in a 1 : 1 relationship
d) nothing to do with 1 : 1 or 1 : N relationship

(b) on the 1 side in a 1 : N relationship

95. Select 'NORTH', CUSTOMER From CUST_DTLS Where REGION = 'N' Order By
CUSTOMER Union Select 'EAST', CUSTOMER From CUST_DTLS Where REGION = 'E' Order By CUSTOMER
The above is
a) Not an error
b) Error - the string in single quotes 'NORTH' and 'SOUTH'
c) Error - the string should be in double quotes
d) Error - ORDER BY clause

(d) Error - the ORDER BY clause. Since ORDER BY clause cannot be used in UNIONS

96. What is Storage Manager?
It is a program module that provides the interface between the low-level data stored in database, application programs and queries submitted to the system.

97. What is Buffer Manager?
It is a program module, which is responsible for fetching data from disk storage into main memory and deciding what data to be cache in memory.

98. What is Transaction Manager?
It is a program module, which ensures that database, remains in a consistent state despite system failures and concurrent transaction execution proceeds without conflicting.

99. What is File Manager?
It is a program module, which manages the allocation of space on disk storage and data structure used to represent information stored on a disk.

100. What is Authorization and Integrity manager?
It is the program module, which tests for the satisfaction of integrity constraint and checks the authority of user to access data.

101. What are stand-alone procedures?
Procedures that are not part of a package are known as stand-alone because they independently defined. A good example of a stand-alone procedure is one written in a SQL*Forms application. These types of procedures are not available for reference from other Oracle tools. Another limitation of stand-alone procedures is that they are compiled at run time, which slows execution.

102. What are cursors give different types of cursors.
PL/SQL uses cursors for all database information accesses statements. The language supports the use two types of cursors
Ø Implicit
Ø Explicit

103. What is cold backup and hot backup (in case of Oracle)?
Ø Cold Backup:
It is copying the three sets of files (database files, redo logs, and control file) when the instance is shut down. This is a straight file copy, usually from the disk directly to tape. You must shut down the instance to guarantee a consistent copy.
If a cold backup is performed, the only option available in the event of data file loss is restoring all the files from the latest backup. All work performed on the database since the last backup is lost.
Ø Hot Backup:
Some sites (such as worldwide airline reservations systems) cannot shut down the database while making a backup copy of the files. The cold backup is not an available option.
So different means of backing up database must be used — the hot backup. Issue a SQL command to indicate to Oracle, on a tablespace-by-tablespace basis, that the files of the tablespace are to backed up. The users can continue to make full use of the files, including making changes to the data. Once the user has indicated that he/she wants to back up the tablespace files, he/she can use the operating system to copy those files to the desired backup destination.
The database must be running in ARCHIVELOG mode for the hot backup option.
If a data loss failure does occur, the lost database files can be restored using the hot backup and the online and offline redo logs created since the backup was done. The database is restored to the most consistent state without any loss of committed transactions.

104. What are Armstrong rules? How do we say that they are complete and/or sound
The well-known inference rules for FDs
Ø Reflexive rule :
If Y is subset or equal to X then X Y.

Ø Augmentation rule:
If X Y then XZ YZ.
Ø Transitive rule:
If {X Y, Y Z} then X Z.
Ø Decomposition rule :

If X YZ then X Y.
Ø Union or Additive rule:
If {X Y, X Z} then X YZ.
Ø Pseudo Transitive rule :
If {X Y, WY Z} then WX Z.
Of these the first three are known as Amstrong Rules. They are sound because it is enough if a set of FDs satisfy these three. They are called complete because using these three rules we can generate the rest all inference rules.

105. How can you find the minimal key of relational schema?
Minimal key is one which can identify each tuple of the given relation schema uniquely. For finding the minimal key it is required to find the closure that is the set of all attributes that are dependent on any given set of attributes under the given set of functional dependency.
Algo. I Determining X+, closure for X, given set of FDs F
1. Set X+ = X
2. Set Old X+ = X+
3. For each FD Y Z in F and if Y belongs to X+ then add Z to X+
4. Repeat steps 2 and 3 until Old X+ = X+

Algo.II Determining minimal K for relation schema R, given set of FDs F
1. Set K to R that is make K a set of all attributes in R
2. For each attribute A in K
a. Compute (K – A)+ with respect to F
b. If (K – A)+ = R then set K = (K – A)+

106. What do you understand by dependency preservation?
Given a relation R and a set of FDs F, dependency preservation states that the closure of the union of the projection of F on each decomposed relation Ri is equal to the closure of F. i.e., ((PR1(F)) U … U (PRn(F)))+ = F+
if decomposition is not dependency preserving, then some dependency is lost in the decomposition.

107. What is meant by Proactive, Retroactive and Simultaneous Update.
Proactive Update:
The updates that are applied to database before it becomes effective in real world .
Retroactive Update:
The updates that are applied to database after it becomes effective in real world .
Simulatneous Update:
The updates that are applied to database at the same time when it becomes effective in real world .

108. What are the different types of JOIN operations?
Equi Join: This is the most common type of join which involves only equality comparisions. The disadvantage in this type of join is that there.