All about Data
engineering

TOP 50 SQL QUERIES FOR INTERVIEW

TOP 50 SQl queries for interview

```
-- Q-1. Write an SQL query to fetch “FIRST_NAME” from
Worker table using the alias name as <WORKER_NAME>.
select first_name AS WORKER_NAME from worker;

-- Q-2. Write an SQL query to fetch “FIRST_NAME” from
Worker table in upper case.
select UPPER(first_name) from worker;

-- Q-3. Write an SQL query to fetch unique values of
DEPARTMENT from Worker table.
SELECT distinct department from worker;

-- Q-4. Write an SQL query to print the first three
characters of  FIRST_NAME from Worker table.
select substring(first_name, 1, 3) from worker;

-- Q-5. Write an SQL query to find the position of the
alphabet (‘b’) in the first name column ‘Amitabh’ from
Worker table.
select INSTR(first_name, 'B') from worker where
first_name = 'Amitabh';

-- Q-6. Write an SQL query to print the FIRST_NAME from
Worker table after removing white spaces from the right
side.
select RTRIM(first_name) from worker;

-- Q-7. Write an SQL query to print the DEPARTMENT from
Worker table after removing white spaces from the left
side.
select LTRIM(first_name) from worker;

-- Q-8. Write an SQL query that fetches the unique
values of DEPARTMENT from Worker table and prints its
length.
select distinct department, LENGTH(department) from
worker;
```

```
-- Q-9. Write an SQL query to print the FIRST_NAME from
Worker table after replacing ‘a’ with ‘A’.
select REPLACE(first_name, 'a', 'A')  from worker;

-- Q-10. Write an SQL query to print the FIRST_NAME and
LAST_NAME from Worker table into a single column
COMPLETE_NAME.
-- A space char should separate them.
select CONCAT(first_name, ' ', last_name) AS
COMPLETE_NAME from worker;

-- Q-11. Write an SQL query to print all Worker details
from the Worker table order by FIRST_NAME Ascending.
select * from worker ORDER by first_name;

-- Q-12. Write an SQL query to print all Worker details
from the Worker table order by
-- FIRST_NAME Ascending and DEPARTMENT Descending.
select * from worker order by first_name, department
DESC;

-- Q-13. Write an SQL query to print details for
Workers with the first name as “Vipul” and “Satish”
from Worker table.
select * from worker where first_name IN ('Vipul',
'Satish');

-- Q-14. Write an SQL query to print details of workers
excluding first names, “Vipul” and “Satish” from Worker
table.
select * from worker where first_name NOT IN ('Vipul',
'Satish');

-- Q-15. Write an SQL query to print details of Workers
with DEPARTMENT name as “Admin*”.
select * from worker where department LIKE 'Admin%';

-- Q-16. Write an SQL query to print details of the
Workers whose FIRST_NAME contains ‘a’.
select * from worker where first_name LIKE '%a%';
```

```
-- Q-17. Write an SQL query to print details of the
Workers whose FIRST_NAME ends with ‘a’.
select * from worker where first_name LIKE '%a';

-- Q-18. Write an SQL query to print details of the
Workers whose FIRST_NAME ends with ‘h’ and contains six
alphabets.
select * from worker where first_name LIKE '_____h';

-- Q-19. Write an SQL query to print details of the
Workers whose SALARY lies between 100000 and 500000.
select * from worker where salary between 100000 AND
500000;

-- Q-20. Write an SQL query to print details of the
Workers who have joined in Feb’2014.
select * from worker where YEAR(joining_date) = 2014
AND MONTH(joining_date) = 02;

-- Q-21. Write an SQL query to fetch the count of
employees working in the department ‘Admin’.
select department, count(*) from worker where
department = 'Admin';

-- Q-22. Write an SQL query to fetch worker full names
with salaries >= 50000 and <= 100000.
select concat(first_name, ' ', last_name) from worker
where salary between 50000 and 100000;

-- Q-23. Write an SQL query to fetch the no. of workers
for each department in the descending order.
select department, count(worker_id) AS no_of_worker
from worker group by department
ORDER BY no_of_worker desc;

-- Q-24. Write an SQL query to print details of the
Workers who are also Managers.
```

```
select w.* from worker as w inner join title as t on
w.worker_id = t.worker_ref_id where t.worker_title =
'Manager';

-- Q-25. Write an SQL query to fetch number (more than
1) of same titles in the ORG of different types.
select worker_title, count(*) as count from title group
by worker_title having count > 1;

-- Q-26. Write an SQL query to show only odd rows from
a table.
-- select * from worker where MOD (WORKER_ID, 2) != 0;
select * from worker where MOD (WORKER_ID, 2) <> 0;

-- Q-27. Write an SQL query to show only even rows from
a table.
select * from worker where MOD (WORKER_ID, 2) = 0;

-- Q-28. Write an SQL query to clone a new table from
another table.
CREATE TABLE worker_clone LIKE worker;
INSERT INTO worker_clone select * from worker;
select * from worker_clone;

-- Q-29. Write an SQL query to fetch intersecting
records of two tables.
select worker.* from worker inner join worker_clone
using(worker_id);

-- Q-30. Write an SQL query to show records from one
table that another table does not have.
-- MINUS
select worker.* from worker left join worker_clone
using(worker_id) WHERE worker_clone.worker_id is NULL;

-- Q-31. Write an SQL query to show the current date
and time.
-- DUAL
select curdate();
select now();
```

```
-- Q-32. Write an SQL query to show the top n (say 5)
records of a table order by descending salary.
select * from worker order by salary desc LIMIT 5;

-- Q-33. Write an SQL query to determine the nth (say
n=5) highest salary from a table.
select * from worker order by salary desc LIMIT 4,1;

-- Q-34. Write an SQL query to determine the 5th
highest salary without using LIMIT keyword.
select salary from worker w1
WHERE 4 = (
SELECT COUNT(DISTINCT (w2.salary))
from worker w2
where w2.salary >= w1.salary
);

-- Q-35. Write an SQL query to fetch the list of
employees with the same salary.
select w1.* from worker w1, worker w2 where w1.salary =
w2.salary and w1.worker_id != w2.worker_id;

-- Q-36. Write an SQL query to show the second highest
salary from a table using sub-query.
select max(salary) from worker
where salary not in (select max(salary) from worker);

-- Q-37. Write an SQL query to show one row twice in
results from a table.
select * from worker
UNION ALL
select * from worker ORDER BY worker_id;

-- Q-38. Write an SQL query to list worker_id who does
not get bonus.
select worker_id from worker where worker_id not in
(select worker_ref_id from bonus);
```

```
-- Q-39. Write an SQL query to fetch the first 50%
records from a table.
select * from worker where worker_id <= ( select
count(worker_id)/2 from worker);

-- Q-40. Write an SQL query to fetch the departments
that have less than 4 people in it.
select department, count(department) as depCount from
worker group by department having depCount < 4;

-- Q-41. Write an SQL query to show all departments
along with the number of people in there.
select department, count(department) as depCount from
worker group by department;

-- Q-42. Write an SQL query to show the last record
from a table.
select * from worker where worker_id = (select
max(worker_id) from worker);

-- Q-43. Write an SQL query to fetch the first row of a
table.
select * from worker where worker_id = (select
min(worker_id) from worker);

-- Q-44. Write an SQL query to fetch the last five
records from a table.
(select * from worker order by worker_id desc limit 5)
order by worker_id;

-- Q-45. Write an SQL query to print the name of
employees having the highest salary in each department.
select w.department, w.first_name, w.salary from
 (select max(salary) as maxsal, department from worker
group by department) temp
inner join worker w on temp.department = w.department
and temp.maxsal = w.salary;

-- Q-46. Write an SQL query to fetch three max salaries
from a table using co-related subquery
```

```
select distinct salary from worker w1
where 3 >= (select count(distinct salary) from worker
w2 where w1.salary <= w2.salary) order by w1.salary
desc;
-- DRY RUN AFTER REVISING THE CORELATED SUBQUERY
CONCEPT FROM LEC-9.
select distinct salary from worker order by salary desc
limit 3;

-- Q-47. Write an SQL query to fetch three min salaries
from a table using co-related subquery
select distinct salary from worker w1
where 3 >= (select count(distinct salary) from worker
w2 where w1.salary >= w2.salary) order by w1.salary
desc;

-- Q-48. Write an SQL query to fetch nth max salaries
from a table.
select distinct salary from worker w1
where n >= (select count(distinct salary) from worker
w2 where w1.salary <= w2.salary) order by w1.salary
desc;

-- Q-49. Write an SQL query to fetch departments along
with the total salaries paid for each of them.
select department , sum(salary) as depSal from worker
group by department order by depSal desc;

-- Q-50. Write an SQL query to fetch the names of
workers who earn the highest salary.
select first_name, salary from worker where salary =
(select max(Salary) from worker);
```

COMPLEX QUERIES

1.To find The Nth Maximum Salary.

SELECT DISTINCT SAL FROM EMP A WHERE &N=(SELECT COUNT (DISTINCT B.SAL)
FROM EMP B WHERE A.SAL<=B.SAL);

2.To find the no. of columns for particular table.

SELECT COUNT (COLUMN_NAME) FROM USER_TAB_COLUMNS
WHERE TABLE_NAME = 'DEPT'

3.To use Exists Clause.

SELECT DNAME, DEPTNO FROM DEPT WHERE EXISTS (SELECT * FROM EMP WHERE
DEPT.DEPTNO = EMP.DEPTNO)
4. To Find The Not Null Column Alone In A Table.SELECT COLUMN_NAME FROM USER_TAB_COLUMNS WHERE NULLABLE = 'N' AND TABLE_NAME = 'COUNTRY'

5.To delete The Duplicate Rows Alone in A Table.

DELETE DEPT WHERE ROWID NOT IN (SELECT MAX (ROWID) FROM DEPT GROUP BY
DEPTNO HAVING COUNT (*) >=1)

6.To find The Max Salary without MAX Function.

1. SELECT DISTINCT SAL FROM EMP1 WHERE SAL NOT IN
(SELECT SAL FROM EMP1 WHERE SAL < ANY (SELECT SAL FROM EMP1))

2.SELECT SAL FROM EMP WHERE SAL >= ALL (SELECT SAL FROM EMP)

7. Alternate for DESC.

SELECT COLUMN_NAME NAME, DECODE (NULLABLE,'N','NOT NULL','Y',' ')
"NULL", CONCAT (DATA_TYPE, DATA_LENGTH) TYPE FROM USER_TAB_COLUMNSWHERE TABLE_NAME = 'DEPT'
8.SQL> Example for startwith, connect by and prior
SELECT ENAME, JOB, LEVEL, EMPNO, MGR FROM EMP111 CONNECT BY PRIOR EMPNO=MGR
START WITH ENAME = 'RAJA'

SELECT EMPNO, LPAD (' ', 6*(LEVEL – 1)) || ENAME "EMPLOYEE NAME" FROM EMP START WITH ENAME='KING' CONNECT BY PRIOR EMPNO = MGR

9. To find the database name

SELECT * FROM GLOBAL_NAME;

10.To convert the given no to word

SELECT TO_CHAR (TO_DATE (&NUM,'J'),'JSP') FROM DUAL;
11. To reverse

12. How can I eliminate duplicate values in a table?

Choose one of the following queries to identify or remove duplicate rows from a table leaving one record:

Method 1:

DELETE FROM table_name A WHERE ROWID > (SELECT min (rowid) FROM table_name B

WHERE A.key_values = B.key_values);

Method 2:

```
SQL> create table table_name2 as select distinct * from table_name1;
SQL> drop table_name1;
SQL> rename table_name2 to table_name1;
```

Method 3: (thanks to Kenneth R Vanluvanee)

SQL> Delete from my_table where rowid not in (select max (rowid) from my_table group by my_column_name);

Method 4: (thanks to Dennis Gurnick)

SQL> delete from my_table t1 where exists (select 'x' from my_table t2 where t2.key_value1 = t1.key_value1
And t2.key_value2 = t1.key_value2and t2.rowid > t1.rowid);

Note: If you create an index on the joined fields in the inner loop, you for all intensive purposes eliminate N^2 operations (no need to loop through the entire table on each pass by a record).

13. How can I generate primary key values for my table?

Create your table with a NOT NULL column (say SEQNO). This column can now be populated with unique values:

SQL> UPDATE table_name SET seqno = ROWNUM;

Or use a sequence generator:

SQL> CREATE SEQUENCE sequence_name START WITH 1 INCREMENT BY 1;
SQL> UPDATE table_name SET seqno = sequence_name. NEXTVAL;

Finally, create a unique index on this column.

14. How can I get the time difference between two date columns?

Select floor ((date1-date2)*24*60*60)/3600) || ' HOURS ' || floor ((((date1-date2)*24*60*60) -
Floor (((date1-date2)*24*60*60)/3600)*3600)/60) || ' MINUTES ' || round ((((date1-date2)*24*60*60) -
Floor (((date1-date2)*24*60*60)/3600)*3600 - (floor ((((date1-date2)*24*60*60) -
Floor (((date1-date2)*24*60*60)/3600)*3600)/60)*60))) || ' SECS ' time_difference from...

15. How does one count different data values in a column?

Select dept, sum (decode (sex,'M', 1,0)) MALE, sum (decode (sex,'F', 1,0)) FEMALE, count (decode (sex,'M', 1,'F', 1)) TOTAL from my_emp_table group by dept;

16. How does one count/sum RANGES of data values in a column?

A value x will be between values y and z if GREATEST (x, y) = LEAST (x, z). Look at this example:

```
Select f2, count (decode (greatest (f1, 59), least (f1, 100), 1, 0)) "Range 60-100",
                Count (decode (greatest (f1, 30), least (f1, 59), 1, 0)) "Range 30-59",
                Count (decode (greatest (f1, 29), least (f1, 0), 1, 0)) "Range 00-29"
From my_table group by f2;
```

For equal size ranges it might be easier to calculate it with DECODE (TRUNC (value/range), 0, rate_0, 1, rate_1,).

E.g.

Select ename "Name", sal "Salary", decode (trunc (f2/1000, 0), 0, 0.0,1, 0.1, 2, 0.2, 3, 0.31) "Tax rate"
From my_table;

17. Can one only retrieve the Nth row from a table?

Ravi Pachalla provided this solution:

```
SELECT f1 FROM t1 WHERE rowid = (SELECT rowid FROM t1 WHERE rownum <= 10
  MINUS
SELECT rowid FROM t1 WHERE rownum < 10);
```

18. Can one only retrieve rows X to Y from a table?

To display rows 5 to 7, construct a query like this:

```
SELECT * FROM tableX WHERE rowid in (SELECT rowid FROM tableX WHERE rownum <= 7
  MINUS
SELECT rowid FROM tableX WHERE rownum < 5);
```

19. How does one select EVERY Nth row from a table?

One can easily select all even, odd, or Nth rows from a table using SQL queries like this:

Method 1: Using a subquery

SELECT *FROM EMP WHERE (ROWID, 0) IN (SELECT ROWID, MOD (ROWNUM, 4) FROM EMP);

Method 2: Use dynamic views (available from Oracle7.2):

SELECT * FROM (SELECT rownum rn, empno, ename FROM EMP) temp WHERE MOD (temp. ROWNUM, 4) = 0;

20. How does one select the TOP N rows from a table?

SELECT * FROM my_table a WHERE 10 >= (SELECT COUNT (DISTINCT maxcol) FROM my_table b
WHERE b.maxcol >= a.maxcol) ORDER BY maxcol DESC;

21. How does one code a tree-structured query?

This is definitely non-relational (enough to kill Codd and then make him roll in his grave) and is a feature I have not seen in the competition.

The definitive example is in the example SCOTT/TIGER database, when looking at the EMP table (EMPNO and MGR columns). The MGR column contains the employee number of the "current" employee's boss.

You have available an extra pseudo-column, LEVEL, that says how deep in the tree you are. Oracle can handle queries with a depth up to 255.

Select LEVEL, EMPNO, ENAME, MGR from EMP connect by prior EMPNO = MGR start with MGR is NULL;

You can get an "indented" report by using the level number to sub-string or lpad a series of spaces and
Concatenate that to the string.

```
Select lpad (' ', LEVEL * 2) || ENAME...
```

You use the start with clause to specify the start of the tree(s). More than one record can match the starting condition. One disadvantage of a "connect by prior" is that you cannot perform a join to other tables. Still, I have not managed to see anything else like the "connect by prior" in the other vendor offerings and I like trees. Even trying to doing this programmatic ally in embedded SQL is difficult as you have to do the top level query, for each of them open a cursor to look for child nodes, for each of these open a cursor.... Pretty soon you blow the cursor limit for your installation.

The way around this is to use PL/SQL, open the driving cursor with the "connect by prior" statement, and the select matching records from other tables on a row-by-row basis, inserting the results into a temporary table for later retrieval.

22. How to implement if-then-else in a select statement?

The Oracle decode function acts like a procedural statement inside an SQL statement to return different values or columns based on the values of other columns in the select statement.

Some examples:

```
Select decode (sex, 'M', 'Male', 'F', 'Female', 'Unknown') from employees;

Select a, b, decode( abs (a-b), a-b, 'a > b',0,   'a = b','a < b') from  tableX;

Select decode (GREATEST (A, B), A, 'A is greater than B', 'B is greater than A')...
```

Note: The decode function is not ANSI SQL and are rarely implemented in other RDBMS offerings. It is one of the good things about Oracle, but use it sparingly if portability is required.

23. How can one dump/ examine the exact content of a database column?

```
SELECT DUMP (col1) FROM tab1 WHERE cond1 = val1;

DUMP (COL1)
-----------------------------------
Typ=96 Len=4: 65,66,67,32
```

For this example the type is 96, indicating CHAR, and the last byte in the column is 32, which is the ASCII code for a space. This tells us that this column is blank-padded.

24. Can one drop a column from a table?

Oracle does not provide a way to DROP a column (reference: Enhancement Request 51118). However, Joseph S. Testa wrote a DROP COLUMN package that can be downloaded from

http://www.oracle-dba.com/ora_scr.htm. Apparently Oracle 8.1.X will have an "ALTER TABLE table_name DROP COLUMN column_name" command.

Other workarounds:

1. Update t1 set column_to_drop = NULL;
   Rename t1 to t1_base;
   Create view t1 as select <specific columns> from t1_base;

2. Create table t2 as select <specific columns> from t1;
   Drop table t1;
   Rename t2 to t1;

25. Can one rename a column in a table?

No, this is listed as Enhancement Request 163519. Workarounds:

1. Rename t1 to t1_base;
   Create view t1 <column list with new name> as select * from t1_base;

2. create table t2 <column list with new name> as select * from t1;
   Drop table t1;
   Rename t2 to t1;

26. How can I change my Oracle password?

Issue the following SQL command:

ALTER USER <username> IDENTIFIED BY <new_password>

27. Sending Messages to Different Session

```
Declare
        a integer;
        b integer;
 Begin
        a := dbms_pipe.create_pipe('kumaran');
        dbms_pipe.pack_message('kumaran software is a good company');
        b := dbms_pipe.send_message('kumaran');
        if b = 0 then
                dbms_output.put_line('successfully send');
        else
                dbms_output.put_line('not send');
        end if;
end;
```

28. Receiving Messages At Different Session

```
declare
        a integer;
        b varchar2(30);
begin
        a := dbms_pipe.receive_message('kumaran');
        dbms_pipe.unpack_message(b);
        if a = 0 then
                dbms_output.put_line('successfully received');
                dbms_output.put_line(b);
        else
                dbms_output.put_line('not received');
        end if;
end;
```

29. Overloading Concept

```
 create or replace package  pw1 as
 procedure pp1(a  char);
 procedure pp1(a  char,b number);
 end pw1;

 create or replace package body pw1 as
 procedure pp1(a char) is
 begin
          dbms_output.put_line(a);
 end;
```

```
procedure pp1(a char,b number) is
 begin
          dbms_output.put_line(a||to_char(b) );
end;
end pw1;
```

30. Restriction Concept

Only local or packaged subprograms can be overloaded. Therefore, you cannot overload standalone subprograms. Also, you cannot overload two subprograms if their formal parameters differ only in name or parameter mode. For example, you cannot overload the following

```
PROCEDURE reconcile (acctno IN INTEGER) IS
BEGIN    ... END;
PROCEDURE reconcile (acctn out INTEGER) IS
BEGIN   ...END;
```

Finally, you cannot overload two functions that differ only in return type (the datatype of the result value) even if the types are in different families. For example, you cannot overload the following functions:

```
FUNCTION acct_ok (acct_id INTEGER) RETURN BOOLEAN IS
BEGIN ... END;
FUNCTION acct_ok (acct_id INTEGER) RETURN INTEGER IS
BEGIN ... END;
```

31. Dynamic Sql

--- Table Creation ----

```
 create or replace procedure tab_creation is
 cursor_name integer;
 ret integer;
 begin
          cursor_name := dbms_sql.open_cursor;
          dbms_sql.parse(cursor_name,'CREATE TABLE first(tname char(20))',dbms_sql.v7);
          ret := dbms_sql.execute(cursor_name);
 if ret = 0 then
          dbms_output.put_line('Created Successfully');
 else
          Dbms_output.put_line('Creation failed');
 end if;
          dbms_sql.close_cursor(cursor_name);
 end;
/
```

--- Deletion from a Table ----

```
CREATE OR REPLACE PROCEDURE demo(salary IN NUMBER) AS
cursor_name INTEGER;
rows_processed INTEGER;
BEGIN
          cursor_name := dbms_sql.open_cursor;
          dbms_sql.parse(cursor_name, 'DELETE FROM emp WHERE sal > :x',dbms_sql);
          dbms_sql.bind_variable(cursor_name, ':x', salary);
          rows_processed := dbms_sql.execute(cursor_name);
          /dbms_sql.close_cursor(cursor_name);
EXCEPTION WHEN OTHERS THEN
          dbms_sql.close_cursor(cursor_name);
END;
```

FREQUENTLY ASKED QUESTIONS

1. How can I dump internal database structures?

```
-- Dump control file contents
Alter session set events 'immediate trace name CONTROLF level 10'
/

-- Dump file headers
Alter session set events 'immediate trace name FILE_HDRS level 10'
/

-- Dump redoes log headers
Alter session set events 'immediate trace name REDOHDR level 10'
/

-- Dump the system state
Alter session set events 'immediate trace name SYSTEMSTATE level 10'
/

-- Dump optimizer statistics whenever a SQL statement is parsed
Alter session set events '10053 trace name context forever'
/
```

2. What database events can be set?

```
# Prevent block corruption
event = "10210 trace name context forever, level 10"
event = "10211 trace name context forever, level 10"
event = "10231 trace name context forever, level 10"
# performance monitoring
event = "10046 trace name context forever, level 12"
# memory protect cursor
event = "10049 trace name context forever, level  2"
# data block check
event = "10210 trace name context forever, level  2"
# index block check
event = "10211 trace name context forever, level  2"
# memory heap check
event = "10235 trace name context forever, level  1"
# allow 300 bytes memory leak for connections
event = "10262 trace name context forever, level 300"
# Trace SQL and show bind variables in trace output
event = "10046 trace name context forever, level 12"
```

3. Is there any undocumented command in Oracle?

Sure there is, but it is hard to find them... In Server Manager from Oracle7.3: ORADEBUG HELP

```
SQL> ALTER SESSION SET CURRENT_SCHEMA = SYS;
```

4. How can I coalesce free space?

SMON coalesces free space (extents) into larger, contiguous extents every 2 hours and even then only for a short period of time. SMON will not coalesce free space if a tablespace's default storage parameter "pctincrease" is set to 0. With Oracle 7.3 one can manually coalesce a tablespace using the ALTER TABLESPACE ... COALESCE; command, until then use:

SQL> alter session set events 'immediate trace name coalesce level n';

Where 'n' is the tablespace number you get from SELECT TS#, NAME FROM SYS.TS$;

You can get status information about this process by selecting from the DBA_FREE_SPACE_COALESCED view.

5. How can I prevent tablespace fragmentation?

Always set PCTINCREASE to 0 or 100. Bizarre values for PCTINCREASE will contribute to fragmentation. For example if you set PCTINCREASE to 1 you will see that your extents are going to have weird and wacky sizes: 100K, 100K, 101K, 102K, etc. Such extents of bizarre size are rarely re-used in their entirety. PCTINCREASE of 0 or 100 gives you nice round extent sizes that can easily be reused. E.g. 100K, 100K, 200K, 400K, etc.

6. Where can one find the high water mark for a table?

There is no system table which containts the high water mark (HWM) information. You can calculate the HWM using the results from the following SQL statements:

```
SELECT BLOCKS FROM DBA_SEGMENTS
WHERE OWNER = UPPER (owner) AND SEGMENT_NAME = UPPER (table);

ANALYZE TABLE owner.table ESTIMATE STATISTICS;

SELECT EMPTY_BLOCKS FROM DBA_TABLES
WHERE OWNER = UPPER (owner) AND SEGMENT_NAME = UPPER (table);
```

Thus, the tables' HWM = (query result 1) - (query result 2) - 1

- You can also use the DBMS_SPACE package and calculate the HWM = TOTAL_BLOCKS - UNUSED_BLOCKS - 1.

7. What can I do about ORA-600 Space Leaks?

You can prevent ORA-600 space leak messages during database shutdown by telling the kernel not to check for memory leakage. This undocumented feature :-) was introduced with Oracle 7.1.6 and can be prevented by setting:

event = "10262 trace name context forever, level 1024"

in the INIT.ORA file or by executing the following SQL command:

SQL> ALTER SESSION SET EVENTS "10262 trace name context forever, level 1024"

8. What database block size should I use?

Oracle recommends that your database blocks size matches, or be multiples of your operating system block size. One can go smaller, but the performance cost is significant. Your choice should depend on the type of application you are running. If you have lots of small transaction like with OLTP, use a small block size. With fewer but larger transactions, like with a DSS application, use a large block size. If you are using a volume manager, consider your "operating system block size" to be 8K. This is because volume manager products use 8K blocks (and this is not configurable).

9. Can one rename a database user (schema)?

No, this is listed as Enhancement Request 158508. Workaround:
Do a user-level export of user A
create new user B
import system/manager fromuser =A touser = B

drop user A

10. Can one rename a tablespace's name?

No, this is listed as Enhancement Request 148742. Workaround:
Export all of the objects from the tablespace
Drop the tablespace including contents
Recreate the tablespace
Import the objects back in

11. Can one resize tablespaces and data files?

You can manually increase or decrease the size of a datafile in Oracle 7.2 using the

ALTER DATABASE DATAFILE 'filename2' RESIZE 100M;

Because you can change the sizes of datafiles, you can add more space to your database without adding more datafiles. This is beneficial if you are concerned about reaching the maximum number of datafiles allowed in your database. Manually reducing the sizes of datafiles allows you to reclaim unused space in the database. This is useful for correcting errors in estimates of space requirements. Also, datafiles can be allowed to automatically extend if more space is required. Look at the following command:

```
CREATE TABLESPACE pcs_data_ts
    DATAFILE 'c:\ora_apps\pcs\pcsdata1.dbf' SIZE 3M
        AUTOEXTEND ON NEXT 1M  MAXSIZE UNLIMITED
    DEFAULT STORAGE (  INITIAL 10240
                NEXT 10240
                MINEXTENTS 1
                MAXEXTENTS UNLIMITED
                PCTINCREASE 0)
    ONLINE
    PERMANENT;
```

12. How does one create a standby database?

While your production database is running, take an ON-LINE backup and restore it on duplicate hardware. Note that an export will not work! On your standby database, issue the following commands:

ALTER DATABASE CREATE STANDBY CONTROLFILE AS...
ALTER DATABASE MOUNT STANDBY DATABASE;
RECOVER STANDBY DATABASE;

Write a job to copy archived redo log files from your primary database to the standby system, and apply the redo log files to the standby database (pipe it). Remember the database is recovering and will prompt you for the next log file to apply.

When you need the standby database stop the recovery process and activate it:

ALTER DATABASE ACTIVATE STANDBY DATABASE;

13. Where can I get a list of all hidden Oracle parameters?

Oracle initialization or INIT.ORA parameters with a underscore in front is hidden or unsupported parameters. You can get a list of all hidden parameters by doing:

```
SELECT *
FROM   SYS.X$KSPPI
WHERE  SUBSTR(KSPPINM,1,1) = '_';
```

14. How can I give developers access to trace files (required as input to tkprof)?

The "alter session set SQL_TRACE = true" command generates trace files in USER_DUMP_DEST that is used by developers as input to tkprof. On Unix the default file mask for these files are "rwx r-- ---".

There is an undocumented INIT.ORA parameter that will allow everybody to read (rwx r-- r--) this trace files:

_trace_files_public = true

Include this in your INIT.ORA file and bounce your database for it to take effect.

15. How can I see what the uptime for my database is?

```
column STARTED format a18 head 'STARTUP TIME'

SELECT
        C.INSTANCE,
        TO_DATE(JUL.VALUE, 'J')
        || TO_CHAR(FLOOR(SEC.VALUE/3600), '09') || ':'
        || SUBSTR (TO_CHAR(MOD(SEC.VALUE/60, 60), '09'), 2, 2)|| '.'
        || SUBSTR (TO_CHAR(MOD(SEC.VALUE,    60), '09'), 2, 2)  STARTED
FROM
        V$INSTANCE JUL,
        V$INSTANCE SEC,
        V$THREAD C
WHERE
        JUL.KEY LIKE '%JULIAN%'
AND     SEC.KEY LIKE '%SECOND%';

select logon_time from v$session where sid=1 /* that's pmon */
/
```

16. How can I become another user in Oracle?

Of course it is not advisable to bridge Oracle's security, but look at this example:

```
SQL> select password from dba_users where username = 'SCOTT';

PASSWORD
 ------------------------------
 F894844C34402B67

SQL> Alter user scott identified by lion;
User altered.

SQL> connect scott/lion
Connected.

REM Do whatever you like...

SQL> connect system/manager
Connected.

SQL> altar user scott identified by values 'F894844C34402B67';
User altered.

SQL> connect scott/tiger
Connected.
```

17. Is it true that OPS$ accounts are a security risk in a C/S environment?

If you allow people to log in with OPS$ accounts from Windows95, you cannot tell who that really is. With terminals, you can rely on passwords, with Win95, you cannot. If you set REMOTE_OS_AUTHENT=TRUE in your init.ora file, Oracle Assumes that the remote OS has authenticated the user.

If REMOTE_OS_AUTHENT is set to FALSE, you have no exposure from remote clients - you also won't be able to connect from a remote client without a password (recommended). IDENTIFIED EXTERNALLY will only be in effect from the local host. Also, if you're using OPS$ as your prefix, you'll be able to log on locally with or without a password, regardless of whether you've identified your ID with a password or defined it to be IDENTIFIED EXTERNALLY.

18. How can one see who is using a temporary segment?

For every user using temporary space, there is an entry in SYS.V$_LOCK with type 'TS'. All temporary segments are named 'ffff.bbbb' where 'ffff' is the file it is in and 'bbbb' is first block of the segment.

If your temporary tablespace is set to TEMPORARY, all sorts are done in one large temporary segment. For usage status, see SYS.V_$SORT_SEGMENT

QUESTIONS & ANSWERS

1. Explain SCN.

Whenever a transaction is committed, LGWR writes transactions redo entries from the redo log buffer of SGA to an online redo file and a (System Change Number) SCN is assigned to identify the redo entries for each committed transaction.

2. What is High SCN?

High SCN:- During the prepare phase (The Global coordinator asks participants to prepare (to promise to Commit or Rollback the transaction, even if there is a failure). The highest SCN at all node in the transaction is determined. The transaction is then committed with the high SCN at the commit point site. The SCN is then sent to all prepared nodes along with the commit decision.

3. What is High Water Mark?

High water mark is the Highest Block Number in which data has been stored in the Segment.

4. What is Row Chaining, Row Migration?

Row Chaining: If an update to a row causes that row to no longer completely fit in a Single data block, then that row may be moved to another data block or the row may be Chained to another block. If row length is greater that the Oracle block size, the row will be chained.

Row Migration: If a row in a data block is updated so that overall row length increases and the block's free space has been completely filled, the data for the entire row is Migrated to a new data block, assuming the entire row can fit in a new block. Oracle preserves the original row piece of a migrated row to point to the new block containing the migrated row; the ROWID of a migrated row does not change.

5. Difference between Row chaining and Row Migration.

Chained rows data is stored in a chain of data blocks and in Row Migration the entire row is shifted to a new location.

6. Coalesce Details.

Is the process by which SMON automatically Coalesces neighboring free extents in a single large free space. It is done automatically if PCT Increase of the Tablespace is non zero or can be done manually by issuing "ALTER TABLESPACE <table_space_name>COALESCE".

7. Define Row Header.

The Header contains general block information, such as Block Address, Segment Type, such as data, Index or Rollback. Some headers are fixed in size, the total block overhead size is variable. On an Average fixed & variable size of data block overhead is total 84 to 107 bytes.

8. What is Buffer Aging.

When oracle process accesses a buffer the process moves the buffer to the most-recently used (MRU) end of the LRU list. As most of the buffers moved to the MRU the dirty "Age" towards the LRU end of the LRU list. This process is called Buffer Aging.

9. What is Honey Comb Fragmentation?

(My point of view on Honey Comb Fragmentation is)
If a tablespace may have two pieces of free space but in between the two, there is a permanent object. This type of Fragmentation is known as Honey Comb Fragmentation.

10. Details about Control File Information.

The Control file of a database is a small binary file necessary for the database to start and operate successfully. A control file is updated continuously by Oracle during database use, so it must be available for writing whenever the database is open. Each control file is associated with only one Oracle Database. Among other things, a control file contains information such as

- The database name
- The timestamp of database creation
- The names and locations of associated database and online redo log files
- The current log sequence number
- Checkpoint information

Each time a data file or a online redo log file is added to, renamed in, or dropped from the database, the control file is updated to reflect this physical structure change. These changes are recorded so that

- Oracle can identify the datafiles and online redo log files to open during database startup.
- Oracle can identify files that are required or available in case database recovery is necessary.
-

It is highly recommended that we backup up our Control file as soon as we make some change to the physical structure of the database.

11. Use of Optimal Size parameter.

The PCTINCREASE parameter has been replaced by a parameter called OPTIMAL. This specifies the optimal size of a rollback segment in bytes. It can also be specified in kilobytes or megabytes. The RDBMS tries to keep the segment at its specified optimal size. The size is rounded up to the extent boundary, which means that the RDBMS tries to have the fewest number of extents such that the total size is greater than or equal to the size specified as OPTIMAL. If additional space is needed beyond the optimal size, it will eventually deallocate extents to shrink back to this size. The process of deallocating extents is performed when the head moves from one extent to the next. At this time, the segment size is checked and the RDBMS determines if the next extent should be deallocated. The extent can only be deallocated if there are no active transaction in it. If necessary, the RDBMS will deallocate multiple extents at one time until the segment has shrunk back to its optimal size. The RDBMS always deallocates the oldest inactive extents as they are the least likely to be used for read consistency.

12. Checkpoint 0

(I Don't know what is Checkpoint 0 means)
By the way, here is some information about Checkpoint process.
Checkpoint (CKPT): When a checkpoint occurs, Oracle must update the headers of all datafiles to indicate the checkpoint. In normal situations, this job is performed by LGWR. However, if checkpoints significantly degrade system performance (usually, when there are many datafiles), you can enable the Checkpoint process (CHPT) to separate the work of performing a checkpoint form other work performed by LGWR, the log writer process (LGWR). For most applications, the CKPT process is not necessary. If your database has many datafiles and the performance of the LGWR process is reduced significantly during checkpoints, you may want to enable the CHPT process.

- CHECKPOINT_PROCESS: Which just enables and disables the checkpoint process.
- Checkpoint Event can be set by two parameters.
  LOG_CHECKPOINT_INTERVAL: - The number of newly filled redo log file blocks needed to trigger a checkpoint. Regardless of this value, a checkpoint always occurs when switching from one online redo log file to another.

LOG_CHECKPOINT_TIMEOUT: - The amount of time to pass before another checkpoint occurs.

Checkpoint process does not hamper the performance of the database but incorrect values for the above two parameters can cause performance degradation.

13. Where analyzed information stored. The analyze information is stored in views like

- DBA_TABLES
- ALL_TABLES
- USER_TABLES

14. How to Activate/Deactivate Index.

There is nothing like activating an Index. But I can say "Oracle automatically maintains and uses indexes once they are created." There is a possibility of forcing a specific index to be used in our query by using Hints.

Example: SELECT (+INDEX name_idx) emp_id, name FROM EMP WHERE name = "ALAM";

The example can be considered as an Activation of an index. (If every reader agrees).

But we can deactivate or disable the indexes or make the optimizer not to use the indexes.

Example: - If an index exists on the Name column of the table EMP.
Case 1: SELECT emp_id, name FROM EMP WHERE name ="ALAM";

When executing the above statement Oracle optimizer will use the index available on the table to resolve the query. But if we want oracle not to use the index we can rewrite the query as follows.

Case 2: SELECT emp_id, name FROM EMPWHERE name ||' ' ="ALAM";

This will intern disable the index. Hope this explains.

15. PCTFREE/PCTUSED functionality

Two space management parameters, control the use of free space for inserts of and updates to the row in data blocks.

PCTFREE: - The PCTFREE parameter is used to set the percentage of a block to be reserved (kept free) for possible updates to rows that already are contained in the blocks.

PCTUSED: - After a data block becomes full, as determined by PCTFREE, oracle does not consider the block is for the insertion of new rows until the percentage of the block being used falls below the parameter PCTUSED.

16. Use of Temporary Tablespace

When processing the queries, Oracles often requires TEMPORARY workspace for intermediate stages of SQL statement processing. Oracle automatically allocates this disk space called a TEMPORARY SEGMENT. Typically, oracle requires a temporary segment as a work area for sorting. Oracle does not create a segment if the sorting operation can be done in memory or if oracle finds some other way to perform the operation using indexes.

Commands requires Temporary segment:

CREATE INDEX
SELECT …ORDER BY
SELECT DISTINCT…
SELECT … GROUP BY
SELECT … UNION
SELECT … INTERSECT
SELECT … MINUS

Unindexed joins
Certain correlated subqueries.

17. Use of SQL* plus trace utility.

SQL* plus trace utility provides information on tuning that can be used in improving the performance of the system.

18. Use of Profile

A profile is a named set of resource limits. If resources limits are turned on, oracle limits user 's use of database and instance resources to that given in his profile. We can assign a profile to a user, and a default profile to all users who do not have specific profiles.

19. How many blocks forms extents, extent form segments?

Oracle stores data in DATA BLOCKS also called as oracle blocks. One data blocks correspond to a specific number of bytes of physical database space on disk. It is set using the parameter DB_BLOCK_SIZE usually 2K or 4K. No of blocks for an extents depends on the size of the Extent itself.

Approximately No_of_blocks = Size_of_extent / DB_BLOCK_SIZE.

Extent is a logical unit of database storage space allocation made up of a number of contiguous data blocks. The extents are allocated based on the storage parameters specified, while creating the objects. No matter what type, each segment in a database is created with at least one extent to hold its data. This extent is called the segment's Initial extent. Exception to this rule is the Rollback Segments; they always have at least two extents.

20. How do you calculate "PCTINCREASE" value?

(As per my knowledge)
There is no way to calculate the value of PCTINCREASE. But, PCTINCREASE specifies the percent by which each extent after the second grows over the previous extent. The default is 50%. We cannot specify PCTINCREASE for Rollback Segments. It is always set to 0 For Rollback Segments.

ORACLE 7 CONCEPTS AND ARCHITECTURE

1. What are the components of Physical database structure of ORACLE database?

ORACLE database is comprised of three types of files: one or more Data files, two or more Redo log file, and one or more Control files.

2. What are the components of Logical database structure of ORACLE database?

Tablespaces and Database's Schema Objects.

3. What is a Tablespace?

A database is divided into logical storage units called TABLESPACES.
A Tablespace is used to group related logical structures together.

4. What is SYSTEM Tablespace and when is it created?

Every ORACLE database contains a Tablespace named SYSTEM, which is automatically created when the database is created. The SYSTEM Tablespace always contains the data dictionary tables for the entire database.

5. Explain the relationship among Database, Tablespace and Data File.

Each database is logically divided into one or more tablespaces. One or more data files are explicitly created for each Tablespace.

6. What is a Schema?

A schema is a collection of database objects of a User.

7. What are Schema Objects?

Schema objects are logical structures that directly refer to the database's data. Schema objects include tables, views, sequences, synonyms, indexes, clusters, database triggers, procedures, functions, packages and database links.

8. Can Objects of the same Schema reside in different tablespaces?

Yes.

9. Can a Tablespace hold objects from different schemas?

Yes.

10. What is a Table?

A table is the basic unit of data storage in an ORACLE database.

QUESTIONS & ANSWERS

1. SNAPSHOT is used for
   [DBA] a] Synonym, b] Table space, c] System server, d] Dynamic data replication

Ans : D

2. We can create SNAPSHOTLOG for
[DBA] a] Simple snapshots, b] Complex snapshots, c] Both A & B, d] Neither A nor B

Ans : A

3. Transactions per rollback segment is derived from
[DBA] a] Db_Block_Buffers, b] Processes, c] Shared_Pool_Size, d] None of the above

Ans : B

4. ENQUEUE resources parameter information is derived from
[DBA] a] Processes or DDL_LOCKS and DML_LOCKS, b] LOG_BUFFER,
c] DB__BLOCK_SIZE..

Ans : A

5. LGWR process writes information into
a] Database files, b] Control files, c] Redolog files, d] All the above.

Ans : C

6. SET TRANSACTION USE ROLLBACK SEGMENT <rbs_name> is used to create user objects in a particular Tablespace
a] True, b] False

Ans : False

7. Databases overall structure is maintained in a file called
a] Redolog file, b] Data file, c] Control file, d] All of the above.

Ans : C

8. These following parameters are optional in init.ora parameter file DB_BLOCK_SIZE, PROCESSES
a] True, b] False

Ans : False

9. Constraints cannot be exported through EXPORT command
a] True, b] False

Ans : False

10. It is very difficult to grant and manage common privileges needed by different groups of database users using the roles
a] True, b] False

Ans : False

11. What is difference between a DIALOG WINDOW and a DOCUMENT WINDOW regarding moving the window with respect to the application window
a] Both windows behave the same way as far as moving the window is concerned.
b] A document window can be moved outside the application window while a dialog window cannot be moved
c] A dialog window can be moved outside the application window while a document window cannot be moved

Ans : C

12. What is the difference between a MESSAGEBOX and an ALERT
a] A messagebox can be used only by the system and cannot be used in user application while an alert can be used in user application also.
b] A alert can be used only by the system and cannot be use din user application while an messagebox can be used in user application also.
c] An alert requires an response from the userwhile a messagebox just flashes a message and only requires an acknowledment from the user

d] An message box requires an response from the userwhile a alert just flashes a message an only requires an acknowledment from the user

Ans : C

13. Which of the following is not an reason for the fact that most of the processing is done at the server ?
    a] To reduce network traffic. b] For application sharing, c] To implement business rules centrally, d] None of the above

Ans : D

14. Can a DIALOG WINDOW have scroll bar attached to it ?
    a] Yes, b] No

Ans : B

15. Which of the following is not an advantage of GUI systems ?
    a] Intuitive and easy to use., b] GUI's can display multiple applications in multiple windows
    c] GUI's provide more user interface objects for a developer
    d] None of the above

Ans :D

16. What is the difference between a LIST BOX and a COMBO BOX ?
    a] In the list box, the user is restricted to selecting a value from a list but in a combo box the user can type in a value which is not in the list
    b] A list box is a data entry area while a combo box can be used only for control purposes
    c] In a combo box, the user is restricted to selecting a value from a list but in a list box the user can type in a value which is not in the list
    d] None of the above

Ans : A

17. In a CLIENT/SERVER environment , which of the following would not be done at the client ?
    a] User interface part, b] Data validation at entry line, c] Responding to user events,
    d] None of the above

Ans : D

18. Why is it better to use an INTEGRITY CONSTRAINT to validate data in a table than to use a STORED PROCEDURE ?
    a] Because an integrity constraint is automatically checked while data is inserted into or updated in a table while a stored procedure has to be specifically invoked
    b] Because the stored procedure occupies more space in the database than a integrity constraint definition
    c] Because a stored procedure creates more network traffic than a integrity constraint definition

Ans : A

19. Which of the following is not an advantage of a client/server model ?
    a] A client/server model allows centralised control of data and centralised implementation of business rules.
    b] A client/server model increases developer;s productivity
    c] A client/server model is suitable for all applications
    d] None of the above.

Ans : C

20. What does DLL stands for ?
    a] Dynamic Language Library
    b] Dynamic Link Library
    c] Dynamic Load Library
    d] None of the above

Ans : B

21. POST-BLOCK trigger is a

a] Navigational trigger
b] Key trigger
c] Transactional trigger
d] None of the above

Ans : A

22. The system variable that records the select statement that SQL * FORMS most recently used to populate a block is
a] SYSTEM.LAST_RECORD
b] SYSTEM.CURSOR_RECORD
c] SYSTEM.CURSOR_FIELD
d] SYSTEM.LAST_QUERY

Ans: D

23. Which of the following is TRUE for the ENFORCE KEY field
a] ENFORCE KEY field characterstic indicates the source of the value that SQL*FORMS uses to populate the field
b] A field with the ENFORCE KEY characterstic should have the INPUT ALLOWED charaterstic turned off
a] Only 1 is TRUE
b] Only 2 is TRUE
c] Both 1 and 2 are TRUE
d] Both 1 and 2 are FALSE

Ans : A

24. What is the maximum size of the page ?
a] Characters wide & 265 characters length
b] Characters wide & 265 characters length
c] Characters wide & 80 characters length
d] None of the above

Ans : B

25. A FORM is madeup of which of the following objects
a] block, fields only,
b] blocks, fields, pages only,
c] blocks, fields, pages, triggers and form level procedures,
d] Only blocks.

Ans : C

26. For the following statements which is true
1] Page is an object owned by a form
2] Pages are a collection of display information such as constant text and graphics.
a] Only 1 is TRUE
b] Only 2 is TRUE
c] Both 1 & 2 are TRUE
d] Both are FALSE

Ans : B

27. The packaged procedure that makes data in form permanent in the Database is
a] Post
b] Post form
c] Commit form
d] None of the above

Ans : C

28. Which of the following is TRUE for the SYSTEM VARIABLE $$date$$
a] Can be assigned to a global variable
b] Can be assigned to any field only during design time
c] Can be assigned to any variable or field during run time
d] None of the above

Ans : B

29. Which of the following packaged procedure is UNRESTRICTED ?
    a] CALL_INPUT, b] CLEAR_BLOCK, c] EXECUTE_QUERY, d] USER_EXIT

Ans : D

30. Identify the RESTRICTED packaged procedure from the following
    a] USER_EXIT, b] MESSAGE, c] BREAK, d] EXIT_FORM

Ans : D

31. What is SQL*FORMS
    a] SQL*FORMS is a 4GL tool for developing & executing Oracle based interactive applications.
    b] SQL*FORMS is a 3GL tool for connecting to the Database.
    c] SQL*FORMS is a reporting tool
    d] None of the above.

Ans : A

32. Name the two files that are created when you generate a form using Forms 3.0
    a] FMB & FMX, b] FMR & FDX, c] INP & FRM, d] None of the above

Ans : C

33. What is a trigger
    a] A piece of logic written in PL/SQL
    b] Executed at the arrival of a SQL*FORMS event
    c] Both A & B
    d] None of the above

Ans : C

34. Which of the folowing is TRUE for a ERASE packaged procedure
    1] ERASE removes an indicated Global variable & releases the memory associated with it
    2] ERASE is used to remove a field from a page
        1] Only 1 is TRUE
        2] Only 2 is TRUE
        3] Both 1 & 2 are TRUE
        4] Both 1 & 2 are FALSE

Ans : 1

35. All datafiles related to a Tablespace are removed when the Tablespace is dropped
    a] TRUE
    b] FALSE

Ans : B

36. Size of Tablespace can be increased by
    a] Increasing the size of one of the Datafiles
    b] Adding one or more Datafiles
    c] Cannot be increased
    d] None of the above

Ans : B

37. Multiple Tablespaces can share a single datafile
    a] TRUE
    b] FALSE

Ans : B

38. A set of Dictionary tables are created
    a] Once for the Entire Database
    b] Every time a user is created
    c] Every time a Tablespace is created
    d] None of the above

Ans : A

39. Datadictionary can span across multiple Tablespaces
    a] TRUE
    b] FALSE

Ans : B

40. What is a DATABLOCK
    a] Set of Extents
    b] Set of Segments
    c] Smallest Database storage unit
    d] None of the above

Ans : C

41. Can an Integrity Constraint be enforced on a table if some existing table data does not satisfy the constraint
    a] Yes
    b] No

Ans : B

42. A column defined as PRIMARY KEY can have NULL's
    a] TRUE
    b] FALSE

Ans : B

43. A Transaction ends
    a] Only when it is Committed
    b] Only when it is Rolledback
    c] When it is Committed or Rolledback
    d] None of the above

Ans : C

44. A Database Procedure is stored in the Database
    a] In compiled form
    b] As source code
    c] Both A & B
    d] Not stored

Ans : C

45. A database trigger doesnot apply to data loaded before the definition of the trigger
    a] TRUE
    b] FALSE

Ans : A

46. Dedicated server configuration is
    a] One server process - Many user processes
    b] Many server processes - One user process
    c] One server process - One user process
    d] Many server processes - Many user processes

Ans : C

47. Which of the following does not affect the size of the SGA
    a] Database buffer
    b] Redolog buffer
    c] Stored procedure
    d] Shared pool

Ans : C

48. What does a COMMIT statement do to a CURSOR

a] Open the Cursor
b] Fetch the Cursor
c] Close the Cursor
d] None of the above

Ans : D

49. Which of the following is TRUE
1] Host variables are declared anywhere in the program
2] Host variables are declared in the DECLARE section
a] Only 1 is TRUE
b] Only 2 is TRUE
c] Both 1 & 2are TRUE
d] Both are FALSE

Ans : B

50. Which of the following is NOT VALID is PL/SQL
a] Bool boolean;
b] NUM1, NUM2 number;
c] deptname dept.dname%type;
d] date1 date := sysdate

Ans : B

51. Declare
fvar number := null; svar number := 5
Begin
goto << fproc>>
if fvar is null then
<< fproc>>
svar := svar + 5
end if;
End;

What will be the value of svar after the execution ?
a] Error
b] 10
c] 5
d] None of the above

Ans : A

52. Which of the following is not correct about an Exception ?
a] Raised automatically / Explicitly in response to an ORACLE_ERROR
b] An exception will be raised when an error occurs in that block
c] Process terminates after completion of error sequence.
d] A Procedure or Sequence of statements may be processed.

Ans : C

53. Which of the following is not correct about User_Defined Exceptions ?
a] Must be declared
b] Must be raised explicitly
c] Raised automatically in response to an Oracle error
d] None of the above

Ans : C

54. A Stored Procedure is a
a] Sequence of SQL or PL/SQL statements to perform specific function
b] Stored in compiled form in the database
c] Can be called from all client environmets

d] All of the above

Ans : D

55. Which of the following statement is false
    a] Any procedure can raise an error and return an user message and error number
    b] Error number ranging from 20000 to 20999 are reserved for user defined messages
    c] Oracle checks Uniqueness of User defined errors
    d] Raise_Application_error is used for raising an user defined error.

Ans : C

56. Is it possible to open a cursor which is in a Package in another procedure ?
    a] Yes
    b] No

Ans : A

57. Is it possible to use Transactional control statements in Database Triggers ?
    a] Yes
    b] No

Ans : B

58. Is it possible to Enable or Disable a Database trigger ?
    a] Yes
    b] No

Ans : A

59. PL/SQL supports datatype(s)
    a] Scalar datatype
    b] Composite datatype
    c] All of the above
    d] None of the above

Ans C

60. Find the ODD datatype out
    a] VARCHAR2
    b] RECORD
    c] BOOLEAN
    d] RAW

Ans : B

61. Which of the following is not correct about the “TABLE” datatype ?
    a] Can contain any no of columns
    b] Simulates a One-dimensional array of unlimited size
    c] Column datatype of any Scalar type
    d] None of the above

Ans : A

62. Find the ODD one out of the following
    a] OPEN
    b] CLOSE
    c] INSERT
    d] FETCH

Ans C

63. Which of the following is not correct about Cursor ?
    a] Cursor is a named Private SQL area
    b] Cursor holds temporary results
    c] Cursor is used for retrieving multiple rows
    d] SQL uses implicit Cursors to retrieve rows

Ans : B

64. Which of the following is NOT VALID in PL/SQL ?
    a] Select ... into
    b] Update
    c] Create
    d] Delete

Ans : C

65. What is the Result of the following 'VIK'||NULL||'RAM' ?
    a] Error
    b] VIK RAM
    c] VIKRAM
    d] NULL

Ans : C

66. Declare
    a number := 5; b number := null; c number := 10;
    Begin
        if a > b AND a < c then
            a := c * a;
        end if;
    End;
What will be the value of 'a' after execution ?
    a] 50
    b] NULL
    c] 5
    d] None of the above

Ans : C

67. Does the Database trigger will fire when the table is TRUNCATED ?
    a] Yes
    b] No

Ans : B

68. SUBSTR(SQUARE ANS ALWAYS WORK HARD,14,6) will return
    a] ALWAY
    b} S ALWA
    c] ALWAYS

Ans : C

69. REPLACE('JACK AND JUE','J','BL') will return
    a] JACK AND BLUE
    b] BLACK AND JACK
    c] BLACK AND BLUE
    d] None of the above

Ans : C

70. TRANSLATE('333SQD234','0123456789ABCDPQRST','0123456789') will return
    a] 333234
    b] 333333
    c] 234333
    d] None of the above

Ans : A

71.

EMPNO	ENAME	SAL	
A822	RAMASWAMY	3500	
A812	NARAYAN	5000	
A973	UMESH		2850
A500	BALAJI	5750	

Use these data for the following Questions

Select SAL from EMP E1 where 3 > (Select count(*) from Emp E2
where E1.SAL > E2.SAL) will retrieve
a] 3500,5000,2500
b] 5000,2850
c] 2850,5750
d] 5000,5750

Ans : A

72. Is it possible to modify a Datatype of a column when column contains data ?
    a] Yes
    b] No

Ans B

73. Which of the following is not correct about a View ?
    a] To protect some of the columns of a table from other users
    b] Ocuupies data storage space
    c] To hide complexity of a query
    d] To hide complexity of a calculations

Ans : B

74. Which is not part of the Data Definiton Language ?
    a] CREATE
    b] ALTER
    c] ALTER SESSION

Ans : C

75. The Data Manipulation Language statements are
    a] INSERT
    b] UPDATE
    c] SELECT
    d] All of the above

Ans : D

76.

EMPNO	ENAME	SAL
A822	RAMASWAMY	3500
A812	NARAYAN	5000
A973	UMESH	

A500 BALAJI 5750

Using the above data
Select count(sal) from Emp will retrieve
a] 1
b] 0
c] 3
d] None of the above

Ans : C

77. If an UNIQUE KEY constraint on DATE column is created, will it accept the rows that are inserted with SYSDATE ?
a] Will
b] Won't

Ans : B

78. What are the different events in Triggers ?
a] Define, Create
b] Drop, Comment
c] Insert, Update, Delete
d] All of the above

Ans : C

79. What built-in subprogram is used to manipulate images in image items ?
a] Zoom_out
b] Zoom_in'
c] Image_zoom
d] Zoom_image

Ans : C

80. Can we pass RECORD GROUP between FORMS ?
a] Yes
b] No

Ans : A

81. SHOW_ALERT function returns
a] Boolean
b] Number
c] Character
d] None of the above

Ans : B

82. What SYSTEM VARIABLE is used to refer DATABASE TIME ?
a] $$dbtime$$
b] $$time$$
c] $$datetime$$
d] None of the above

Ans : A

83. :SYSTEM.EFFECTIVE.DATE varaible is
a] Read only
b] Read & Write
c] Write only

d] None of the above

Ans : C

84. How can you CALL Reports from Forms4.0 ?
a] Run_Report built_in
b] Call_Report built_in
c] Run_Product built_in
d] Call_Product built_in

Ans : C

85. When do you get a .PLL extension ?
a] Save Library file
b] Generate Library file
c] Run Library file
d] None of the above

Ans : A

86. What is built_in Subprogram ?
a] Stored procedure & Function
b] Collection of Subprogram
c] Collection of Packages
d] None of the above

Ans : D

87. GET_BLOCK property is a
a] Restricted procedure
b] Unrestricted procedure
c] Library function
d] None of the above

Ans : D

88. A CONTROL BLOCK can sometimes refer to a BASETABLE ?
a] TRUE
b] FALSE

Ans : B

89. What do you mean by CHECK BOX ?
a] Two state control
b] One state control
c] Three state control
d] none of the above

Ans : C - Please check the Correcness of this Answer (The correct answer is 2)

90. List of Values (LOV) supports
a] Single column
b] Multi column
c] Single or Multi column
d] None of the above

Ans : C

91. What is Library in Forms 4.0 ?
a] Collection of External field

b] Collection of built_in packages
c] Collection of PL/SQl functions, procedures and packages
d] Collection of PL/SQL procedures & triggers

Ans : C

92. Can we use a RESTRICTED packaged procedure in WHEN_TEXT_ITEM trigger ?
a] Yes
b] No

Ans : B

93. Can we use GO_BLOCK package in a PRE_TEXT_ITEM trigger ?
a] Yes
b] No

Ans : B

94. What type of file is used for porting Forms 4.5 applications to various platforms ?
a] .FMB file
b] .FMX file
c] .FMT file
d] .EXE file

Ans : C

95. What built_in procedure is used to get IMAGES in Forms 4.5 ?
a] READ_IMAGE_FILE
b] GET_IMAGE_FILE
c] READ_FILE
d] GET_FILE

Ans A

96. When a form is invoked with CALL_FORM does Oracle forms issues SAVEPOINT ?
a] Yes
b] No

Ans : A

97. Can we attach the same LOV to different fields in Design time ?
a] Yes
b] No

Ans : A

98. How do you pass values from one form to another form ?
a] LOV
b] Parameters
c] Local variables
d] None of the above

Ans : B

99. Can you copy the PROGRAM UNIT into an Object group ?
a] Yes
b] No

Ans : B

100. Can MULTIPLE DOCUMENT INTERFACE (MDI) be used in Forms 4.5 ?
    a] Yes
    b] No

Ans : A

101. When is a .FMB file extension is created in Forms 4.5 ?
    a] Generating form
    b] Executing form
    c] Save form
    d] Run form

Ans : C

102. What is a Built_in subprogram ?
    a] Library
    b] Stored procedure & Function
    c] Collection of Subprograms
    d] None of the above

Ans : D

103. What is a RADIO GROUP ?
    a] Mutually exclusive
    b] Select more than one column
    c] Above all TRUE
    d] Above all FALSE

Ans : A

104. Identify the Odd one of the following statements ?
    a] Poplist
    b] Tlist
    c] List of values
    d] Combo box

Ans : C

105. What is an ALERT ?
    a] Modeless window
    b] Modal window
    c] Both are TRUE
    d] None of the above

Ans : B

106. Can an Alert message be changed at runtime ?
    a] Yes
    b] No

Ans : A

107. Can we create an LOV without an RECORD GROUP ?
    a} Yes
    b] No

Ans : B

108. How many no of columns can a RECORD GROUP have ?
    a] 10

b] 20
c] 50
d] None of the above

Ans D

109. Oracle precompiler translates the EMBEDDED SQL statemens into
a] Oracle FORMS
b] Oracle REPORTS
c] Oracle LIBRARY
d] None of the above

Ans : D

110. Kind of COMMENT statements placed within SQL statements ?
a] Asterisk(*) in column ?
b] ANSI SQL style statements(...)
c] C-Style comments (/*.....*/)
d] All the above

Ans : D

111. What is the appropriate destination type to send the output to a printer ?
a] Screen
b] Previewer
c] Either of the above
d] None of the above

Ans : D

112. What is TERM ?
a] TERM is the terminal definition file that describes the terminal from which you are using R20RUN (Reports run time)
b] TERM is the terminal definition file that describes the terminal from which you are using R20DES (Reports designer)
c] There is no Parameter called TERM in Reports 2.0
d] None of the above

Ans : A

113. If the maximum records retrieved property of a query is set to 10, then a summary value will be calculated
a] Only for 10 records
b] For all the records retrieved
c] For all therecords in the referenced table
d] None of the above

Ans : A

114. With which function of a summary item in the COMPUTE AT option required ?
a] Sum
b] Standard deviation
c] Variance
d] % of Total function

Ans : D

115. For a field in a repeating frame, can the source come from a column which does not exist in the datagroup which forms the base of the frame ?
a] Yes

b] No

Ans : A

116. What are the different file extensions that are created by Oracle Reports ?
a] .RDF file & .RPX file
b] .RDX file & .RDF file
c] .REP file & .RDF file
d] None of the above

Ans : C

117. Is it possible to Disable the Parameter form while running the report ?
a] Yes
b] No

Ans : A

118.What are the SQL clauses supported in the link property sheet ?
a] WHERE & START WITH
b] WHERE & HAVING
c} START WITH & HAVING
d] WHERE, START WITH & HAVING

Ans : D

119. What are the types of Calculated columns available ?
a] Summary, Place holder & Procedure column
b] Summary, Procedure & Formula columns
c] Procedure, Formula & Place holder columns
d] Summary, Formula & Place holder columns

Ans : D

120. If two groups are not linked in the data model editor, what is the hierarchy between them ?
a] There is no hierarchy betweeen unlimked groups
b] The group that is right ranks higher than the group that is to the left
c] The group that is above or leftmost ranks higher than the group that is to right or below it
d] None of the above

Ans : C

121. Sequence of events takes place while starting a Database is
a] Database opened, File mounted, Instance started
b] Instance started, Database mounted & Database opened
c] Database opened, Instance started & file mounted
d] Files mounted, Instance started & Database opened

Ans : B

122. SYSTEM TABLESPACE can be made off-line
a] Yes
b] No

Ans : B

123. ENQUEUE_RESOURCES parameter informatiom is derived from
a] PROCESS or DDL_LOCKS & DML_LOCKS
b] LOG BUFFER

c] DB_BLOCK_SIZE
d] DB_BLOCK_BUFFERS

Ans : A

124. SMON process is used to write into LOG files
a] TRUE
b] FALSE

Ans : B

125. EXP command is used
a] To take Backup of the Oracle Database
b] To import data from the exported dump file
c] To create Rollback segments
d] None of the above

Ans : A

126. SNAPSHOTS cannot be refreshed automatically
a] TRUE
b] FALSE

Ans : B

127. Archieve file name formats can be set by the User
a] TRUE
b] FALSE

Ans : A

128. The following parameters are optional in init.ora parameter file DB_BLOCK_SIZE, PROCESS
a} TRUE
b] FALSE

Ans : B

129. NOARCHIEVELOG parameter is used to enable the database in Archieve mode
a] TRUE
b] FALSE

Ans : B

130. Constraints cannot be exported through Export command ?
a] TRUE
b] FALSE

Ans : B

131. It is very difficult to grant and manage common priveleges needed by different groups of database users using roles
a] TRUE
b] FALSE

Ans : B

132. The status of the Rollback segment can be viewed through
a] DBA_SEGMENTS
b] DBA_ROLES
c] DBA_FREE_SPACES

d] DBA_ROLLBACK_SEG

Ans : D

133. Explicitly we can assign transaction to a rollback segment
a] TRUE
B] FALSE

Ans : A

134. What file is read by ODBC to load drivers ?
a] ODBC.INI
b] ODBC.DLL
c] ODBCDRV.INI
d] None of the above

Ans : A

Can I Update From Another Table?

Yes. For example, if we had a table DEPT_SUMMARY, we could update the number of employees' field as follows:

```
update DEPT_SUMMARY s
set NUM_EMPS = (
      select count(1)
      from EMP E
      where E.DEPTNO = S.DEPTNO
);
```

Can I remove duplicate rows?

Yes, using the ROWID field. The ROWID is guaranteed unique. There are many variations on this theme, but the logic is to delete all but one record for each key value.

```
delete from EMP E
where not E.ROWID = (
      select min(F.ROWID)
      from EMP F
      where  F.EMP_ID = E.EMP_ID
      );
```

Can I implement Tree Structured Queries?

Yes! Those migrating from non-RDBMS application commonly ask this. This is definitely non-relational (enough to kill Codd and then make him roll in his grave) and is a feature I have not seen in the competition.

The definitive example is in the example SCOTT/TIGER database, when looking at the EMP table (EMPNO and MGR columns). The MGR column contains the employee number of the "current" employee's boss.

You have available an extra pseudo-column, LEVEL, that says how deep in the tree you are. Oracle can handle queries with a depth up to 255.

select LEVEL, EMPNO, ENAME, MGR from EMP connect by prior EMPNO = MGR start with MGR is NULL;

You can get an "indented" report by using the level number to sub-string or lpad a series of spaces and concatenate that to the string.

select lpad(' ‘, LEVEL * 2) || ENAME .....

You use the start with clause to specify the start of the tree(s). More than one record can match the starting condition.

One disadvantage of a "connect by prior" is that you cannot perform a join to other tables. Still, I have not managed to see anything else like the "connect by prior" in the other vendor offerings and I like trees. Even trying to do this programmatically in embedded SQL is difficult, as you have to do the top-level query, for each of them open a cursor to look for lower levelrows, for each of these...

Soon you blow the cursor limit for your installation.

The way around this is to use PL/SQL, open the driving cursor with the "connect by prior" statement, and the select matching records from other tables on a row-by-row basis, inserting the results into a temporary table for later retrieval.

Note that you can't trick Oracle by using CONNECT BY PRIOR on a view that does the join.

How can I get information on the row based on group information?

Imagine we have the EMP table and want details on the employee who has the highest salary. You need to use a sub query.

```
select e.ENAME, e.EMPNO, e.SAL
from EMP e
where e.SAL in (
      select max (e2.SAL)
      from EMP e2
);
```

You could get similar info on employees with the highest salary in their departments as follows

```
select e.ENAME, e.DEPTNO, e.SAL
from EMP e
where e.SAL = (
      select max (e2.SAL)
      from EMP e2
      where e2.DEPTNO = e.DEPTNO
);
```

How can I get a name for a temporary table that will not clash?

Use a sequence, and use the number to help you build the temporary table name. Note that SQL-92 is developing specific constructs for using temporary tables.

How can I discover what tables, columns, etc are there?

Oracle maintains a live set of views that you can query to tell you what you have available. In V6, the first two to look at are DICT and DICT_COLUMNS, which act as a directory of the other dictionary views. It is a good idea to be familiar with these. Not all of these views are accessible by all users. If you are a DBA you should also create private DBA synonyms by running

$ORACLE_HOME/rdbms/admin/dba_syn.sql in your account.

How can I rename a column?

There is no way a column can be renamed using normal SQL. It can be done carefully by the DBA playing around with internal SYS dictionary tables and bouncing the database, but this is not supported. (I have successfully done it in V4 through V7). Do backup the database first unless you feel brave. I've written a quick and dirty script rncol.sql to do this. If you can't figure out how to use it from the source you definitely should not run it. You can use a similar dirty trick for changing ownership of tables if storage space is limited.

Is there a formatter for SQL or PL/SQL?

There are a number of "beautifiers" for various program languages. The CB and indent programs for the C language spring to mind (although they have slightly different conventions). As far as I know there is no PD formatter for SQL available.

Given that there are PD general SQL parsers and that the SQL standards are drafted in something close to BNF, maybe someone could base a reformatted based on the grammar.

Note that you CANNOT use CB and indent with Pro *C as both these programs will screw up the embedded SQL code.

I have recently heard that Kumaran Systems (see Vendor list) have a Forms PL/SQL and SQL formatter, but I do not now if they have unbundled it.

How come records for the date I want are missing?

You are trying to retrieve data based on something like:
SELECT fld1, fld2 FROM tbl WHERE date_field = '18-jun-60'

You *know* there are records for that day - but none of them are coming back to you.

What has happened is that your records are not set to midnight (which is the default value if time of day not specified)?

You can either use to_char and to_date functions, which can be a bad move regarding SQL performance, or you can say
WHERE date_field >= '18-jun-60' AND date_field < '19-jun-60'

An alternative could be something like

```
WHERE date_field between '18-jun-1960'
AND to_date('23:59:59 18-jun-60', 'HH24:.....YY')
;
```

How can I interpret a two-digit year after 2000?

When converting to dates from characters when you only have two characters for the year, the picture format "RR" will be interpreted as the year based on a guess that that date is between 1950 and 2049.

What are these V$ tables?

There are a number of tables/views beginnings with V$ that holds gory details for performance monitoring. These are not guaranteed to be stable from minor release to minor release and are for DBAs only.

There are usually no real underlying tables (unlike SYS.OBJ$) and are dummied up by the RDBMS kernel software in much the same way that UNIX System V.4 dummies up the files in the /proc or /dev/proc directories.

If you have any code depending on these (and the widely used tools supplied by Oracle but unsupported are in this category) then you need to verify that everything works each time you upgrade your database. And when a major revision changes, all bets are off.

How do I get a top ten?

This question often gets the response WHERE ROWNUM <= 10 but this will not work (except accidentally) because the ROWNUM pseudo column is generated before the ORDER or WHERE clauses come into effect.

Stowe@aol.com (although it will be a bitch on a large table) suggested one elegant SQL-only approach

```
select a.ordered_column, a.other_stuff
from   table_name a
where  10 > (
       select count(1)
       from   table_name b
       where  b.ordered_column
              < a.ordered_column )
order by a.ordered_columnl;
```

I do not believe that straight SQL is the way to go for such problems when you have PL/SQL available.
My approach is to use PL/SQL instead (in SQL*Plus):

```
variable tenthsal number
declare
       n number;
       cursor c1 is
              select SAL from EMP order BY SAL desc;
begin
       open c1;
       for n in 1..10 loop
       fetch c1 into :tenthsal;
end loop;
close c1;
end:
/
select * from EMP
where SAL <= :tenthsal   order by SAL desc;
```

Late news: index-descending hint to SQL works if you use a dummy restriction to force use of the index. Needs V7, etc.

How do control which rollback segment I use ?

In SQL, you may need to control the rollback segment used as the default rollback segment may be too small for the required transaction, or you may want to ensure that your transaction runs in a special rollback segment, unaffected by others. The statement is as follows:

SET TRANSACTION USE ROLLBACK SEGMENT segment_name;

On a related note, if all you are doing are SELECTS, it is worth telling the database of this using the following:

SET TRANSACTION READ ONLY;

Both these statements must be the first statements of the transaction.

How do I order a union ?

Use the column number.

Say we are getting a list of names and codes and want it ordered by the name, using both EMP and DEPT tables:

```
select DEPTNO, DNAME from DEPT
union
select EMPNO, ENAME from EMP
order by 2;
```

Who are SCOTT, SYSTEM and SYS ?

These three users are common in many databases. See the glossary entries under SCOTT, SCOTT and SYS. Another common user/password is PLSQL/SUPERSECRET used for PL/SQL demo Stuff.

How can I avoid blowing rollback segments ?

The simple answer is make sure you have them big enough and keep your transactions small, but that is being a smartness. More recent versions of Oracle have an option for the session that you can set that commits every so many DML statements. This is OK except for where you are doing your work in a single statement rather than using PL/SQL and a loop construct.

Imagine you have a HUGE table and need to update it, possibly updating the key. You cannot update it in one go because your rollback segments are too small. You cannot open a cursor and commit every n records, because usually the cursor will close. You cannot have a number of updates of a few records each because the keys may change - causing you to visit records more than once.

The solution I have used was to have one process select ROWID from the appropriate rows and pump these (via standard I/O) to another process that looped around reading ROWIDs from standard input, updating the appropriate record and committing every 10 records or so. This was very easy to program and also was quite fast in execution. The number of locks and size of rollback segments required was minimal.

If you are writing in Pro *C and use MODE=ORACLE, there are ways around it too, but not if you are using MODE=ANSI.

How can I restore passwords ?

OK, so this is really a DBA question, but it is worth putting in here because it involves SQL regardless of interface.

First, look at the PASSWORD column in DBA_USERS. It looks like gobbledygook because it is an encrypted password.However you can use this if you have saved it somewhere else. Say you want to impersonate a user in a batch run overnight.First stash the gobbledygook password away

somewhere, grant connect to the user identified by some password you know and then run your batches using the new known password.

To restore the password to what it was use the following syntax (which I think is undocumented).

grant connects to SCOTT identified by passwords GOBBLEDYGOOK;

Note especially the S on the end of PASSWORDS.

Who do various access methods compare ?

How you organize your SQL and indices controls what access methods will be used. The following ranking is valid for V6. I do not know about V7. QUERY PATH RANKING (lowest rank is the best)

Rank	Path
1	ROWID = constant
2	Unique index column(s) = constant(s)
3	Entire unique concatenated index = constant
4	Entire cluster key = corresponding key in another table in same cluster
5	Entire cluster key = constant
6	Entire non-unique concatenated index = constant
7	Non-unique single column index merge
8	Most leading concatenated index = constant
9	Index column BETWEEN low AND hi or LIKE 'C%'
10	Sort/merge (joins only)
11	MAX/MIN of single indexed column
12	ORDER BY entire index
13	Full table scans
14	Unindexed column = constant or column IS NULL or column LIKE '%C%'

Views

Can I update through a view ?

You can do this iff

1. Your view is a simple subset of a single table.
2. All "not null" columns for the table must be in the view.
3. The primary key is in the view.

The typical example is the view on EMP limited to a department and not including salary. Also see CHECK OPTION discussion.

What is CHECK OPTION for a view ?

Imagine we have created a view of EMP limited to a department, (where DEPTNO = 10). Now, that is fine for querying, but you can still write records through this view (either by update or insert) with a value of 20 for DEPTNO. (Next time you query the view, such records will be invisible.)

Now if you want to stop someone doing this (and consider whether you want them to be able to do this or not very carefully) use the "check option" when creating the view:

```
create view DEPT_TEN
as
select EMPNO, DEPTNO, ENAME
from EMP
where DEPTNO = 10
with check option;
```

Are views updated when I update base tables ?

Yes, that is the whole idea of views. The only thing Oracle stores for a view is the text of the definition. When you select from a view, Oracle looks up the text used to define the view and then executes that query.

Should we use complex views that cruel performance ?

Because view queries that involve sorting, grouping, etc can lead to a high performance overhead, it might be better to write some reports with a procedural component that fills up a temporary table and then does a number of queries from it.

While this is non-relational, it can be justified for some cases. Nevertheless, it is useful to have the view definition in the database. You can then test the output from the view against the output from your procedural manipulations. The view definition can also be used as the unambiguous gospel.

How can I get the definition of a view ?

There are a number of dictionary views that include the text of views. You can select these quite happily, but remember, if using SQL*Plus to use the SET command to fiddle with ARRAYSIZE, MAXDATA and LONG parameters

DATA DICTIONARY TABLES

ALL_CATALOG
All tables, views, synonyms, sequences accessible to the user
ALL_COL_COMMENTS
Comments on columns of accessible tables and views
ALL_COL_PRIVS
Grants on columns for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
ALL_COL_PRIVS_MADE
Grants on columns for which the user is owner or grantor
ALL_COL_PRIVS_RECD
Grants on columns for which the user, PUBLIC or enabled role is the grantee
ALL_CONSTRAINTS
Constraint definitions on accessible tables
ALL_CONS_COLUMNS
Information about accessible columns in constraint definitions
ALL_DB_LINKS
Database links accessible to the user
ALL_DEF_AUDIT_OPTS
Auditing options for newly created objects
ALL_DEPENDENCIES
Dependencies to and from objects accessible to the user
ALL_ERRORS
Current errors on stored objects that user is allowed to create
ALL_INDEXES
Descriptions of indexes on tables accessible to the user
ALL_IND_COLUMNS
COLUMNs comprising INDEXs on accessible TABLES
ALL_OBJECTS
Objects accessible to the user
ALL_SEQUENCES
Description of SEQUENCEs accessible to the user
ALL_SNAPSHOTS
Snapshots the user can look at
ALL_SOURCE
Current source on stored objects that user is allowed to create

ALL_SYNONYMS
    All synonyms accessible to the user
ALL_TABLES
    Description of tables accessible to the user
ALL_TAB_COLUMNS
    Columns of all tables, views and clusters
ALL_TAB_COMMENTS
    Comments on tables and views accessible to the user
ALL_TAB_PRIVS
    Grants on objects for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
ALL_TAB_PRIVS_MADE
    User's grants and grants on user's objects
ALL_TAB_PRIVS_RECD
    Grants on objects for which the user, PUBLIC or enabled role is the grantee
ALL_TRIGGERS
    Triggers accessible to the current user
ALL_TRIGGER_COLS
    Column usage in user's triggers or in triggers on user's tables
ALL_USERS
    Information about all users of the database
ALL_VIEWS
    Text of views accessible to the user
AUDIT_ACTIONS
    Description table for audit trail action type codes. Maps action type numbers to action type names
CAT Synonym for USER_CATALOG
CLU Synonym for USER_CLUSTERS
COLS
    Synonym for USER_TAB_COLUMNS
COLUMN_PRIVILEGES
    Grants on columns for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
DBA_2PC_NEIGHBORS
    information about incoming and outgoing connections for pending transactions
DBA_2PC_PENDING
    info about distributed transactions awaiting recovery
DBA_ANALYZE_OBJECTS
DBA_AUDIT_EXISTS
    Lists audit trail entries produced by AUDIT NOT EXISTS and AUDIT EXISTS
DBA_AUDIT_OBJECT
    Audit trail records for statements concerning objects, specifically: table, cluster, view, index, sequence, [public]
    database link, [public] synonym, procedure, trigger, rollback segment, table space, role, user
DBA_AUDIT_SESSION
DBA_AUDIT_STATEMENT
    Audit trail records concerning grant, revoke, audit, noaudit and alter system
DBA_AUDIT_TRAIL
    All audit trail entries
DBA_CATALOG
    All database Tables, Views, Synonyms, Sequences
DBA_CLUSTERS
    Description of all clusters in the database
DBA_CLU_COLUMNS
    Mapping of table columns to cluster columns
DBA_COL_COMMENTS
    Comments on columns of all tables and views
DBA_COL_PRIVS
    All grants on columns in the database
DBA_CONSTRAINTS
    Constraint definitions on all tables

DBA_CONS_COLUMNS
Information about accessible columns in constraint definitions
DBA_DATA_FILES
Information about database files
DBA_DB_LINKS
All database links in the database
DBA_DEPENDENCIES
Dependencies to and from objects
DBA_ERRORS
Current errors on all stored objects in the database
DBA_EXP_FILES
Description of export files
DBA_EXP_OBJECTS
Objects that have been incrementally exported
DBA_EXP_VERSION
Version number of the last export session
DBA_EXTENTS
Extents comprising all segments in the database
DBA_FREE_SPACE
Free extents in all tablespaces
DBA_INDEXES
Description for all indexes in the database
DBA_IND_COLUMNS
COLUMNs comprising INDEXES on all TABLES and CLUSTERS
DBA_OBJECTS
All objects in the database
DBA_OBJECT_SIZE
Sizes, in bytes, of various PL/SQL objects
DBA_OBJ_AUDIT_OPTS
Auditing options for all tables and views
DBA_PRIV_AUDIT_OPTS
Describes current system privileges being audited across the system and by use r
DBA_PROFILES
Display all profiles and their limits
DBA_ROLES
All Roles which exist in the database
DBA_ROLE_PRIVS
Roles granted to users and roles
DBA_ROLLBACK_SEGS
Description of rollback segments
DBA_SEGMENTS
Storage allocated for all database segments
DBA_SEQUENCES
Description of all SEQUENCES in the database
DBA_SNAPSHOTS
All snapshots in the database
DBA_SNAPSHOT_LOGS
All snapshot logs in the database
DBA_SOURCE
Source of all stored objects in the database
DBA_STMT_AUDIT_OPTS
Describes current system auditing options across the system and by user
DBA_SYNONYMS
All synonyms in the database
DBA_SYS_PRIVS
System privileges granted to users and roles
DBA_TABLES
Description of all tables in the database
DBA_TABLESPACES
Description of all tablespaces

DBA_TAB_COLUMNS
    Columns of all tables, views and clusters
DBA_TAB_COMMENTS
    Comments on all tables and views in the database
DBA_TAB_PRIVS
    All grants on objects in the database
DBA_TRIGGERS
    All triggers in the database
DBA_TRIGGER_COLS
    Column usage in all triggers
DBA_TS_QUOTAS
    Tablespace quotas for all users
DBA_USERS
    Information about all users of the database
DBA_VIEWS
    Text of all views in the database
DICT Synonym for DICTIONARY
DICTIONARY
    Description of data dictionary tables and views
DICT_COLUMNS
    Description of columns in data dictionary tables and views
DUAL
GLOBAL_NAME
    global database name
IND Synonym for USER_INDEXES
INDEX_HISTOGRAM
    statistics on keys with repeat count
INDEX_STATS
    statistics on the b-tree
OBJ Synonym for USER_OBJECTS
RESOURCE_COST
    Cost for each resource
ROLE_ROLE_PRIVS
    Roles which are granted to roles
ROLE_SYS_PRIVS
    System privileges granted to roles
ROLE_TAB_PRIVS
    Table privileges granted to roles
SEQ Synonym for USER_SEQUENCES
SESSION_PRIVS
    Privileges which the user currently has set
SESSION_ROLES
    Roles which the user currently has enabled.
SYN Synonym for USER_SYNONYMS
TABLE_PRIVILEGES
    Grants on objects for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
TABS Synonym for USER_TABLES
USER_AUDIT_OBJECT
    Audit trail records for statements concerning objects, specifically: table, cluster, view, index, sequence, [public] database link, [public] synonym, procedure, trigger, rollback segment, tablespace, role, user
USER_AUDIT_SESSION
USER_AUDIT_STATEMENT
    Audit trail records concerning grant, revoke, audit, noaudit and alter system
USER_AUDIT_TRAIL
    Audit trail entries relevant to the user
USER_CATALOG
    Tables, Views, Synonyms and Sequences owned by the user
USER_CLUSTERS

Descriptions of user's own clusters
USER_CLU_COLUMNS
Mapping of table columns to cluster columns
USER_COL_COMMENTS
Comments on columns of user's tables and views
USER_COL_PRIVS
Grants on columns for which the user is the owner, grantor or grantee
USER_COL_PRIVS_MADE
All grants on columns of objects owned by the user
USER_COL_PRIVS_RECD
Grants on columns for which the user is the grantee
USER_CONSTRAINTS
Constraint definitions on user's own tables
USER_CONS_COLUMNS
Information about accessible columns in constraint definitions
USER_DB_LINKS
Database links owned by the user
USER_DEPENDENCIES
Dependencies to and from a users objects
USER_ERRORS
Current errors on stored objects owned by the user
USER_EXTENTS
Extents comprising segments owned by the user
USER_FREE_SPACE
Free extents in tablespaces accessible to the user
USER_INDEXES
Description of the user's own indexes
USER_IND_COLUMNS
COLUMNs comprising user's INDEXes or on user's TABLES
USER_OBJECTS
Objects owned by the user
USER_OBJECT_SIZE
Sizes, in bytes, of various PL/SQL objects
USER_OBJ_AUDIT_OPTS
Auditing options for user's own tables and views
USER_RESOURCE_LIMITS
Display resource limit of the user
USER_ROLE_PRIVS
Roles granted to current user
USER_SEGMENTS
Storage allocated for all database segments
USER_SEQUENCES
Description of the user's own SEQUENCEs
USER_SNAPSHOTS
Snapshots the user can look at
USER_SNAPSHOT_LOGS
All snapshot logs owned by the user
USER_SOURCE
Source of stored objects accessible to the user
USER_SYNONYMS
The user's private synonyms
USER_SYS_PRIVS
System privileges granted to current user
USER_TABLES
Description of the user's own tables
USER_TABLESPACES
Description of accessible tablespaces
USER_TAB_COLUMNS
Columns of user's tables, views and clusters
USER_TAB_COMMENTS

Comments on the tables and views owned by the user
USER_TAB_PRIVS
Grants on objects for which the user is the owner, grantor or grantee
USER_TAB_PRIVS_MADE
All grants on objects owned by the user
USER_TAB_PRIVS_RECD
Grants on objects for which the user is the grantee
USER_TRIGGERS
Triggers owned by the user
USER_TRIGGER_COLS
Column usage in user's triggers
USER_TS_QUOTAS
Tablespace quotas for the user
USER_USERS
Information about the current user
USER_VIEWS
Text of views owned by the user
V$ACCESS
Synonym for V_$ACCESS
V$ARCHIVE
Synonym for V_$ARCHIVE
V$BACKUP
Synonym for V_$BACKUP
V$BGPROCESS
Synonym for V_$BGPROCESS
V$CIRCUIT
Synonym for V_$CIRCUIT
V$CONTROLFILE
Synonym for V_$CONTROLFILE
V$DATABASE
Synonym for V_$DATABASE
V$DATAFILE
Synonym for V_$DATAFILE
V$DBFILE
Synonym for V_$DBFILE
V$DBLINK
Synonym for V_$DBLINK
V$DB_OBJECT_CACHE
Synonym for V_$DB_OBJECT_CACHE
V$DISPATCHER
Synonym for V_$DISPATCHER
V$ENABLEDPRIVS
Synonym for V_$ENABLEDPRIVS
V$FILESTAT
Synonym for V_$FILESTAT
V$FIXED_TABLE
Synonym for V_$FIXED_TABLE
V$LATCH
Synonym for V_$LATCH
V$LATCHHOLDER
Synonym for V_$LATCHHOLDER
V$LATCHNAME
Synonym for V_$LATCHNAME
V$LIBRARYCACHE
Synonym for V_$LIBRARYCACHE
V$LICENSE
Synonym for V_$LICENSE
V$LOADCSTAT
Synonym for V_$LOADCSTAT
V$LOADTSTAT

Synonym for V_$LOADTSTAT
V$LOCK
Synonym for V_$LOCK
V$LOG
Synonym for V_$LOG
V$LOGFILE
Synonym for V_$LOGFILE
V$LOGHIST
Synonym for V_$LOGHIST
V$LOG_HISTORY
Synonym for V_$LOG_HISTORY
V$MLS_PARAMETERS
Synonym for V_$MLS_PARAMETERS
V$MTS
Synonym for V_$MTS
V$NLS_PARAMETERS
Synonym for V_$NLS_PARAMETERS
V$OPEN_CURSOR
Synonym for V_$OPEN_CURSOR
V$PARAMETER
Synonym for V_$PARAMETER
V$PROCESS
Synonym for V_$PROCESS
V$QUEUE
Synonym for V_$QUEUE
V$RECOVERY_LOG
Synonym for V_$RECOVERY_LOG
V$RECOVER_FILE
Synonym for V_$RECOVER_FILE
V$REQDIST
Synonym for V_$REQDIST
V$RESOURCE
Synonym for V_$RESOURCE
V$ROLLNAME
Synonym for V_$ROLLNAME
V$ROLLSTAT
Synonym for V_$ROLLSTAT
V$ROWCACHE
Synonym for V_$ROWCACHE
V$SESSION
Synonym for V_$SESSION
V$SESSION_CURSOR_CACHE
Synonym for V_$SESSION_CURSOR_CACHE
V$SESSION_EVENT
Synonym for V_$SESSION_EVENT
V$SESSION_WAIT
Synonym for V_$SESSION_WAIT
V$SESSTAT
Synonym for V_$SESSTAT
V$SGA
Synonym for V_$SGA
V$SGASTAT
Synonym for V_$SGASTAT
V$SHARED_SERVER
Synonym for V_$SHARED_SERVER
V$SQLAREA
Synonym for V_$SQLAREA
V$STATNAME
Synonym for V_$STATNAME
V$SYSSTAT

Synonym for V_$SYSSTAT
V$SYSTEM_CURSOR_CACHE
Synonym for V_$SYSTEM_CURSOR_CACHE
V$SYSTEM_EVENT
Synonym for V_$SYSTEM_EVENT
V$THREAD
Synonym for V_$THREAD
V$TIMER
Synonym for V_$TIMER
V$TRANSACTION
Synonym for V_$TRANSACTION
V$TYPE_SIZE
Synonym for V_$TYPE_SIZE
V$VERSION
Synonym for V_$VERSION
V$WAITSTAT
Synonym for V_$WAITSTAT
V$_LOCK
Synonym for V_$_LOCK SQL

What is SQL and where does it come from?

Structured Query Language (SQL) is a language that provides an interface to relational database systems. SQL was developed by IBM in the 1970s for use in System R. SQL is a defacto standard, as well as an ISO and ANSI standard. SQL is often pronounced SEQUEL.

In common usage SQL also encompasses DML (Data Manipulation Language), for INSERTs, UPDATEs, DELETEs and DDL (Data Definition Language), used for creating and modifying tables and other database structures.

The development of SQL is governed by standards. A major revision to the SQL standard was completed in 1992 called SQL2. SQL3 supports object extensions and will be (partially?) implemented in Oracle8.

How can I eliminate duplicate values in a table?

Choose one of the following queries to identify or remove duplicate rows from a table leaving one record:

Method 1:

```
DELETE FROM TABLE_NAME A WHERE ROWID > (SELECT MIN (ROWID) FROM TABLE_NAME
B WHERE A.KEY_VALUES = B.KEY_VALUES);
```

Method 2:

```
CREATE TABLE TABLE_NAME2 AS SELECT DISTINCT * FROM TABLE_NAME1;
DROP TABLE_NAME1;
RENAME TABLE_NAME2 TO TABLE_NAME1;
```

Method 3:

```
DELETE FROM MY_TABLE WHERE ROWID NOT IN (SELECT MAX(ROWID) FROM MY_TABLE
GROUP BY MY_COLUMN_NAME );
```

Method 4:

```
DELETE FROM MY_TABLE T1 WHERE EXISTS (SELECT 'X' FROM MY_TABLE T2 WHERE
T2.KEY_VALUE1 = T1.KEY_VALUE1 AND T2.KEY_VALUE2 = T1.KEY_VALUE2 AND T2.ROWID
> T1.ROWID);
```

Note: If you create an index on the joined fields in the inner loop, you for all intensive purposes eliminate N^2 operations (no need to loop through the entire table on each pass by a record).

How can I generate primary key values for my table?

Create your table with a NOT NULL column (say SEQNO). This column can now be populated with unique values:

```
UPDATE TABLE_NAME SET SEQNO = ROWNUM;
```

or use a sequences generator:

```
CREATE SEQUENCE SEQUENCE_NAME START WITH 1 INCREMENT BY 1;
UPDATE TABLE_NAME SET SEQNO = SEQUENCE_NAME.NEXTVAL;
```

Finally, create a unique index on this column.

How can I get the time difference between two date columns?

```
SELECT FLOOR ((DATE1-DATE2)*24*60*60)/3600) || ' HOURS ' || FLOOR ((((DATE1-
DATE2)*24*60*60) – FLOOR (((DATE1-DATE2)*24*60*60)/3600)*3600)/60) || ' MINUTES ' || ROUND
((((DATE1-DATE2)*24*60*60) – FLOOR (((DATE1-DATE2)*24*60*60)/3600)*3600 - (FLOOR
((((DATE1-DATE2)*24*60*60) – FLOOR (((DATE1-DATE2)*24*60*60)/3600)*3600)/60)*60))) || '
SECS ' TIME_DIFFERENCE FROM...
```

How does one count different data values in a column?

SELECT DEPT, SUM (DECODE (SEX,'M', 1,0)) MALE, SUM (DECODE (SEX,'F', 1,0)) FEMALE, COUNT (DECODE (SEX,'M', 1,'F', 1)) TOTAL FROM MY_EMP_TABLE GROUP BY DEPT;

How does one count/sum RANGES of data values in a column?

A value x will be between values y and z if GREATEST (x, y) = LEAST(x, z). Look at this example:

SELECT F2, COUNT (DECODE (GREATEST (F1, 59), LEAST (F1, 100), 1, 0)) "RANGE 60-100", COUNT (DECODE (GREATEST (F1, 30), LEAST (F1, 59), 1, 0)) "RANGE 30-59", COUNT (DECODE (GREATEST (F1, 29), LEAST (F1, 0), 1, 0)) "RANGE 00-29" FROM MY_TABLE GROUP BY F2;

For equal size ranges it might be easier to calculate it with DECODE (TRUNC (value/range), 0, rate_0, 1, rate_1...).

Example:
SELECT ENAME "NAME", SAL "SALARY", DECODE (TRUNC (F2/1000, 0), 0, 0.0,1, 0.1, 2, 0.2, 3, 0.31) "TAX RATE" FROM MY_TABLE;

Can one only retrieve the Nth row from a table?

SELECT F1 FROM T1 WHERE ROWID =
(SELECT ROWID FROM T1 WHERE ROWNUM <= 10
MINUS
SELECT ROWID FROM T1 WHERE ROWNUM < 10);

Can one only retrieve rows X to Y from a table?

To display rows 5 to 7, construct a query like this:

SELECT * FROM TABLEX WHERE ROWID IN (SELECT ROWID FROM TABLEX WHERE ROWNUM <= 7
MINUS
SELECT ROWID FROM TABLEX WHERE ROWNUM < 5);

How does one select EVERY Nth row from a table?

One can easily select all even, odd, or Nth rows from a table using SQL queries like this:

Method 1: Using a sub-query

SELECT * FROM EMP WHERE (ROWID, 0) IN (SELECT ROWID, MOD (ROWNUM,4) FROM EMP);

Method 2: Use dynamic views (available from Oracle7.2):

SELECT * FROM (SELECT ROWNUM RN, EMPNO, ENAME FROM EMP) TEMP WHERE MOD (TEMP.ROWNUM, 4) = 0;

How does one select the TOP N rows from a table?

SELECT * FROM MY_TABLE A WHERE 10 >= (SELECT COUNT (DISTINCT MAXCOL) FROM MY_TABLE B WHERE B.MAXCOL >= A.MAXCOL) ORDER BY MAXCOL DESC;

How does one code a tree-structured query?

This is definitely non-relational (enough to kill Codd and then make him roll in his grave) and is a feature I have not seen in the competition.

The definitive example is in the example SCOTT/TIGER database, when looking at the EMP table (EMPNO and MGR columns). The MGR column contains the employee number of the "current" employee's boss.

You have available an extra pseudo-column, LEVEL, that says how deep in the tree you are. Oracle can handle queries with a depth up to 255.

SELECT LEVEL, EMPNO, ENAME, MGR from EMP connect by prior EMPNO = MGR start with MGR is NULL;

You can get an "indented" report by using the level number to sub-string or lpad a series of spaces and concatenate that to the string.

```
SELECT LPAD(' ', LEVEL * 2) || ENAME .....
```

You use the start with clause to specify the start of the tree(s). More than one record can match the starting condition. One disadvantage of a "connect by prior" is that you cannot perform a join to other tables. Still, I have not managed to see anything else like the "connect by prior" in the other vendor offerings and I like trees. Even trying to doing this programmatic ally in embedded SQL is difficult as you have to do the top level query, for each of them open a cursor to look for child nodes, for each of these open a cursor. Pretty soon you blow the cursor limit for your installation.

The way around this is to use PL/SQL, open the driving cursor with the "connect by prior" statement, and the select matching records from other tables on a row-by-row basis, inserting the results into a temporary table for later retrieval.

How to implement if-then-else in a select statement?

The Oracle decode function acts like a procedural statement inside an SQL statement to return different values or columns based on the values of other columns in the select statement.

Some examples:

```
SELECT DECODE (SEX, 'M', 'MALE', 'F', 'FEMALE', 'UNKNOWN') FROM EMPLOYEES;

SELECT A, B, DECODE (ABS (A-B), A-B, 'A > B', 0,   'A = B','A < B') FROM TABLEX;


SELECT DECODE (GREATEST (A, B), A, 'A IS GREATER THAN B', 'B IS GREATER THAN A')...
```

Note: The DECODE function is not ANSI SQL and are rarely implemented in other RDBMS offerings. It is one of the good things about Oracle, but use it sparingly if portability is required.

How can one dump/ examine the exact content of a database column?

```
SELECT DUMP (col1) FROM tab1 WHERE cond1 = val1;

DUMP (COL1)
------------------------------------
Typ=96 Len=4: 65,66,67,32
```

For this example the type is 96, indicating CHAR, and the last byte in the column is 32, which is the ASCII code for a space. This tells us that this column is blank-padded.

Can one drop a column from a table?

Oracle does not provide a way to DROP a column (reference: Enhancement Request 51118). However, Joseph S. Testa wrote a DROP COLUMN package that can be downloaded from

Apparently Oracle 8.1.X will have an

"ALTER TABLE table_name DROP COLUMN column_name" command.

Other workarounds:

1. Update t1 set column_to_drop = NULL;
   Rename t1 to t1_base;
   Create view t1 as select <specific columns> from t1_base;

2. Create table t2 as select <specific columns> from t1;
   Drop table t1;
   Rename t2 to t1;

Can one rename a column in a table?

No, this is listed as Enhancement Request 163519. Workarounds:

1. Rename t1 to t1_base;
   Create view t1 <column list with new name> as select * from t1_base;

2. Create table t2 <column list with new name> as select * from t1;
   Drop table t1;
   Rename t2 to t1;

How can I change my Oracle password?

Issue the following SQL command:

ALTER USER <username> IDENTIFIED BY <new_password>
/

From Oracle8 you can just type "password" from SQL* Plus, or if you need to change another user's password, type "password username".

How does one find the next value of a sequence?

Perform an "ALTER SEQUENCE ... NOCACHE" to unload the unused cached sequence numbers from the Oracle library cache. This way, no cached numbers will be lost. If you then select from the USER_SEQUENCES dictionary view, you will see the correct high water mark value that would be returned for the next NEXTVALL call. Afterwards, perform an "ALTER SEQUENCE ... CACHE" to restore caching.

You can use the above technique to prevent sequence number loss before a SHUTDOWN ABORT, or any other operation that would cause gaps in sequence values.

Workaround for snapshots on tables with LONG columns

You can use the SQL* Plus COPY command instead of snapshots if you need to copy LONG and LONG RAW variables from one location to another. E.g.

```
COPY TO SCOTT/TIGER@REMOTE     -
CREATE IMAGE_TABLE USING       -
SELECT IMAGE_NO, IMAGE  - FROM IMAGES;
```

Note: If you run Oracle8, convert your LONGs to LOBs, as it can be replicated.

What is PL/SQL and what is it good for?

PL/SQL is Oracle's Procedural Language extension to SQL. PL/SQL's language syntax, structure and data types are similar to that of ADA. The language includes object oriented programming techniques

such as encapsulation, function overloading, information hiding (all but inheritance) and is commonly used to write data-centric programs to manipulate Oracle data.

How can I protect my PL/SQL source code?

PL/SQL V2.2, available with Oracle7.2, implements a binary wrapper for PL/SQL programs to protect the source code.

This is done via a standalone utility that transforms the PL/SQL source code into portable binary object code (somewhat larger than the original). This way you can distribute software without having to worry about exposing your proprietary algorithms and methods. SQL* Plus and SQL*DBA will still understand and know how to execute such scripts. Just be careful, there is no "decode" command available.

The syntax is:

wrap iname = myscript.sql oname = xxxx.plb

Can one print to the screen from PL/SQL?

One can use the DBMS_OUTPUT package to write information to an output buffer. This buffer can be displayed on the screen from SQL* Plus if you issue the SET SERVEROUTPUT ON; command.
For example:

```
  begin
dbms_output.put_line ('Look Ma, I can print from PL/SQL!!!');
  end;
  /
```

But what if you forget to set server output on? No problem, just type SET SERVEROUTPUT ON once you remember, and then EXEC NULL; If you haven't cleared the DBMS_OUTPUT buffer with the disable or enable procedure, SQL* Plus will display the entire contents of the buffer when it executes this dummy PL/SQL block.

Can one read/write files from PL/SQL?

Included in Oracle 7.3 is an UTL_FILE package that can read and write operating system files. The directory you intend writing to has to be in your INIT.ORA file (see UTL_FILE_DIR=...parameter).

Before Oracle 7.3 the only means of writing a file was to use DBMS_OUTPUT with the SQL* Plus SPOOL
command.

```
DECLARE
  fileHandler UTL_FILE.FILE_TYPE;
BEGIN
  fileHandler := UTL_FILE.FOPEN('/tmp', 'myfile', 'w');
  UTL_FILE.PUTF(fileHandler, 'Look ma, I''m writing to a file!!!\n');
  UTL_FILE.FCLOSE(fileHandler);
EXCEPTION
  WHEN utl_file.invalid_path THEN
     raise_application_error(-20000, 'ERROR: Invalid path for file or
path not in INIT.ORA.');
END;
```

Can one use dynamic SQL within PL/SQL?

From PL/SQL V2.1 one can use the DBMS_SQL package to execute dynamic SQL statements. E.g.:

```
CREATE OR REPLACE PROCEDURE DYNSQL AS
```

```
  cur integer;
  rc  integer;
BEGIN
  cur := DBMS_SQL.OPEN_CURSOR;
  DBMS_SQL.PARSE(cur, 'CREATE TABLE X (Y DATE)', DBMS_SQL.NATIVE);
  rc := DBMS_SQL.EXECUTE(cur);
  DBMS_SQL.CLOSE_CURSOR(cur);
END;
/
```

Another example:

```
CREATE OR REPLACE PROCEDURE DEPARTMENTS(NO IN DEPT.DEPTNO%TYPE) AS
  v_cursor integer;
  v_dname  char(20);
  v_rows   integer;
BEGIN
  v_cursor := DBMS_SQL.OPEN_CURSOR;
  DBMS_SQL.PARSE(v_cursor, 'select dname from dept where deptno > :x',
DBMS_SQL.V7);
  DBMS_SQL.BIND_VARIABLE(v_cursor, ':x', no);
  DBMS_SQL.DEFINE_COLUMN_CHAR(v_cursor, 1, v_dname, 20);
  v_rows := DBMS_SQL.EXECUTE(v_cursor);
  loop
    if DBMS_SQL.FETCH_ROWS(v_cursor) = 0 then
       exit;
    end if;
    DBMS_SQL.COLUMN_VALUE_CHAR(v_cursor, 1, v_dname);
    DBMS_OUTPUT.PUT_LINE('Deptartment name: '||v_dname);
  end loop;
  DBMS_SQL.CLOSE_CURSOR(v_cursor);
EXCEPTION
  when others then
       DBMS_SQL.CLOSE_CURSOR(v_cursor);
       raise_application_error(-20000, 'Unknown Exception Raised:
'||sqlcode||' '||sqlerrm);
END;
/
```

How does one get the value of a sequence into a PL/SQL variable?

As you might know, oracle prohibits this:

```
    i := sq_sequence.NEXTVAL;
```

(for some silly reason). But you can do this:

```
    select sq_sequence.NEXTVAL into :i from dual;
```

Can one execute an operating system command from PL/SQL?

In Oracle7 there is no direct way to execute an operating system command from PL/SQL. However, one can write an external program (using one of the pre-compiler languages, OCI or Perl with Oracle access modules) to act as a listener on a DBMS_PIPE. Your PL/SQL then places requests to run commands in the pipe, the listener picks it up and runs them. Results are passes back on a different pipe. For a Pro *C example, see chapter 8 of the Oracle Application Developers Guide. This example is also on the Support Notes CD-ROM that all supported customers should be getting quarterly. Just search on "DBMS_PIPE".

In Oracle8 you can call external 3GL code in a dynamically linked library (DLL or shared object). So you just write a library in C doing what you want, ie in your case a host function taking the command line as input argument. And that function will be callable from PL/SQL.

So what you have to do is more or less:

1. Write C code in host.c: int host (char *command) {.}.
2. Compile C code to DLL or shared object, eg c:\winnt\system32\host.dll.
3. "Create or replace library host as 'c:\winnt\system32\host.dll';"
4. "Create or replace function host (command in varchar2) return pls_integer is external library host name "host" language c calling standard Pascal parameters (host string, return long);"

Let's say I had a PL/SQL block and I wanted to do a "ls -l" to get a directory listing. Is there any way to do that?

In C Language, I can do

```
 {
   x = system("ls -l");
 }
```

The way most people do that is to use the pipe facility of the DBMS kernel. Set up a pipe for use by a sender (the PL/SQL program that needs to invoke a command) and a receiver, written in Pro *C. This is responsible for executing the command piped by the sender. Maybe there are some tools around so one doesn't need to code this, as it involves a listener and multiple server processes on your machine.

Alternatively I have a more complex solution, which uses a table to pass arguments and the command to execute - just as the DBMS_PIPE package does internally. You would insert a row into the table and the listener would execute a command, passing back succession status and indicating "in progress" on long-running commands. This tool allows for non-blocking execution of commands.

How does one loop through tables in PL/SQL?

Look at the following nested loop code example.

```
 DECLARE
   CURSOR dept_cur IS
   SELECT deptno
     FROM dept
    ORDER BY deptno;
   -- Employee cursor all employees for a dept number
   CURSOR emp_cur (v_dept_no DEPT.DEPTNO%TYPE) IS
   SELECT ename
     FROM emp
    WHERE deptno = v_dept_no;
 BEGIN
   FOR dept_rec IN dept_cur LOOP
     dbms_output.put_line('Employees in Department
'||TO_CHAR(dept_rec.deptno));
      FOR emp_rec in emp_cur(dept_rec.deptno) LOOP
        dbms_output.put_line('...Employee is '||emp_rec.ename);
      END LOOP;
  END LOOP;
 END;
```

Back to top of file

Is there a PL/SQL Engine in SQL* Plus?

No. Unlike Oracle Forms, SQL* Plus does not have a PL/SQL engine. Thus, all your PL/SQL are send directly to the database engine for execution. This makes it much more efficient as SQL statements are not stripped off and send to the database individually.

Is there a limit on the size of a PL/SQL block?

Yes, the max size is not an explicit byte limit, but related to the parse tree that is created when you compile the code.

You can run the following select statement to query the size of an existing package or procedure:

SQL> select * from dba_object_size where name = 'procedure_name'

I switched the page size to 11x8.5, but the printer still prints in portrait.

Even though you set the page size in the report properties, there is a another variable in the system parameters section under the data model in the object navigator called orientation. This sets the printer orientation. Oracle starts by setting it to "default" which means that no matter how you set the page size, the user's default printer setup will be used. You can also set it to either "Landscape" or "Portrait" to force the printer orientation no matter what the user has set as default. These sorts of picky, minor details are the ones, which are invariably forgotten when you are designing your report and are the reason I created our two report templates, reptmp_p and reptmp_l (portrait and landscape). For anyone who wants a consistent look in their reports I strongly recommend building a similar pair to save yourself an ulcer, unless you actually like starting from scratch every time.

I moved this field into that repeating frame, but I'm still getting a "frequency below it's group" error.

Moving fields around does not change what enclosing object is considered it's parent group. Oracle carefully remembers what repeating frame a field was originally placed in and assigns that as it's parent. If you then reference a column further down the line of the query structure it will return that error. If you are not exactly sure which repeating frame a field belongs to, try dragging it out of all of them. Whichever frame will not allow it to escape is it's parent. To change a field's parent, first click on the lock button on the speed button bar. It should now look like an unlocked padlock. Now all of the fields on the layout can be repositioned regardless of their original parent items. When you are satisfied with the repositioning click the lock button again to lock the layout. Oracle will parse the layout and assumes that any item fully enclosed in a repeating frame is a child object of that frame. This can be confirmed again by trying to drag an object out of it's parent. (CTRL - Z or EDIT…. UNDO will put it back where it came from)

Sometimes, for unknown and mysterious reasons, this method does not work. The alternative in this case is to highlight the field (or fields), cut it (CTRL-X), and then paste it into the desired frame. The paste does not initially set it into the right frame, but if you drag and drop it there before clicking on any other objects, and then click on something else, Oracle will usually figure what your intent was and assign the object(s) as a child of that frame. This is my preferred method of changing a field's parent as it works much more consistently then the unlock/lock method. One note though, if you are reassigning a group of fields, make sure the frame you are going to move them into is large enough to accept the whole group at once before you do the cut/paste. If you do the paste and then try to grow the frame to fit, you will have to cut and paste again. Once you de-select an object that has just been pasted, Oracle will assign it as a child of whatever it is in at the time.

If this technique also fails, you are probably going to have to delete and then recreate the objects within the desired frame. If the object has triggers attached, save yourself some typing by creating the new object in the right frame, copying over the trigger code, and then deleting the old object.

I must put a repeating frame around these fields. How do I do this easily?

Well congratulations, you have just discovered one of the main reasons why good planning goes a long way. Oracle looks at the layout as a sort of layered inheritance model such that anything created on top of and completely inside another object is by definition a child of that object. Creation order is therefor critical to the layout process. This means that placing a repeating frame on top of a field but

larger than that field fails the ownership criteria. At best, if the new frame is fully enclosed within the same higher level frame as the field thenthe two will be considered sibling children of the higher level frame.

From this point you have two options. First, you can place the new repeating frame in the correct place and then, use the techniques shown above in the "I moved this field but am still getting a frequency error" to reassign the fields into the new frame. There is also a second choice (which can also be used as a solution to the above). Go ahead and draw the new frame around the fields you want to have placed in it. Now if you try to click on one of the fields you will not be able to as they are fully covered by the new frame. Now go to the "Arrange" menu. You will find the options send to back, bring to front, move forwards and move backwards. These are used to alter an object position in the Reports layer ordering. You use the "send backwards" option to move the frame backwards until all of the fields have popped to the front and are now enclosed in it. Oracle reassigns the new repeating frame as each object's parent as they pop to the front.

Note that you can only move an object back and forth amongst it's siblings. You cannot set it back below its parent, or in front of its children. This means that once an object has popped to the front and had a reassignment of parent, you cannot move it back using these tools.

Why does part of a row sometimes get shifted to the next page, but not all of it?

This is due to the way the scan works when Oracle is parsing the layout. If the tops of all the fields in a row are aligned and the fields are all of the same height and font, they should all stay together. I suspect, however, that Reports bases it's decision on the printed size rather than the field size you define to determine which objects are too large and must be shifted to the next page. This means that even if you set two fields top-aligned with the same height and font but one of them is bolded, the bolded field could get shifted to the next page due to it's bigger footprint. The solution is to put the whole row into a regular frame, which is page protected.

What exactly does the "Print Condition" do?

The print condition type First, All, All but first, Last, All but last refer to the frequency with which you want to appear based upon the setting of the print condition object. A print condition object of Enclosing Object is whichever object encloses the current object (could be the parent or a frame within the parent), while Anchoring Object is the parent object (unless you have explicitly anchored the object in which case it is the object to which it is anchored). The key here is that this is about the pages on which the Print Condition Object appears, not the current object. Oracle views First as the first page on which any part of the Print Condition Object is printed, likewise Last is the last page on which any part of the Print Condition Object is printed. For objects inside a repeating frame, this condition is re-evaluated for each instance of the frame.

As an example, assume we have created a field inside a repeating frame with Print Condition Object set to 'anchoring object', and Print Condition Typeset to 'All But First'. On every instance of that repeating frame which is printed entirely within a single page, our object will not print. However, if an instance of that frame spans more than one page then our object will print on the second and every subsequent page that this instance of the repeating frame spans.

For most objects you will not have to play with this print condition setting as the default setting is pretty good at determining what pages to print on, even though it only chooses between 'first' and 'last'. Only such things as heading objects you want reprinted on multiple pages are normally candidates for fooling around with this setting.

How do I create a truly dynamic 'where' condition which the user can input on the parameter form for my select statement

While setting a simple parameter for use in defining the select statement, such as a date, bill_period_id etc. Is simple, there are times when you may wish to allow a user to add any "where" statement they wish. However, if you create a VARCHAR user variable and try to reference it as an SQL condition (e.g. Select * from account where :user condition) you will get an error. The secret is that the variable must be initialized to a valid SQL condition before the Data Model will accept it. This

is done in the "Initial Value" spot on the variable's property form. The usual default is "1 = 1" which simply means all rows meeting whatever other conditions are included in the select statement will pass this condition if the user does not change it in the parameter form.

How do I change a user parameter at runtime from a layout object trigger?

Quite simply, you can't. Once the BeforeReport trigger has fired, reports locks down the user parameters until the report is finished. Oh, I know you can put a statement into a layout trigger at design time and the compiler will accept it, but the moment you run the report you will get a nasty error and the report will die. Why they couldn't catch those problems at compile time I have no idea, except that it probably uses the same PL/SQL compiler as Forms which uses that same syntax for the perfectly acceptable function of changing field values.

That being said, there is valid technique to mimic having a user variable, which can be changed over the course of the report execution. What you have to do is create a PL/SQL package that contains a variable as well as the functions to read and write to that variable. Since variables inside a package are both local to that package and persistent over the duration of the run, you use this to save and change your variable value. I know that this seems like overkill, but it is the most efficient way of handling an issue that is very rarely encountered. As you can probably guess, this technique is a last resort to finding an SQL work around if one exists.

How do I set the initial values of parameters for the parameter form at runtime?

This is what the BeforeForm trigger is primarily used for. Even if you have used a select statement to create a lookup list for the parameter, this statement is fully parsed before the parameter form is opened. Simply setting the parameter to a given value in the BeforeForm trigger will select that option as the default value displayed to the user. For example, assume you have a parameter called p_input_date, which is intended to hold an invoice date. The following example will select the most recent invoice date as the default, and note that it properly handles exceptions to ensure that the report does not arbitrarily die if this default setting fails. Note also that like all report triggers, it must return a true or false value.

```
function BeforePForm return boolean is
begin
select max(bill_period_end_date + 1)
  into :p_input_date
  from billing_period
 where bill_period_end_date <= (select trunc(sysdate)  from dual);
  return (TRUE);
exception
  when others then
     :p_input_date := null;
     return true;
end;
```

Why can't I highlight a bunch of fields and change their entire format masks or prints conditions at once?

You can. If you highlight a bunch of objects and then right click and select "properties..", Oracle gives you a stacked set of the individual properties forms for each of the selected objects. While this may be useful for some things, it requires changing values individually for each object. However, instead you can select the group of fields and then select "Common properties" from the "Tools" menu which will allow you to set the format mask , print conditions etc. for the whole set of objects at once.

How do I change the printed value of a field at runtime?

Triggers are intended to simply provide a true or false return value to determine whether an object should be, printed. It is generally not allowed to change any values held in the cursor, make changes to the database, or change the value of it's objects value. That being said, there is a highly unpublicized method of doing just that using the SRW. Set_Field_Char procedure. The syntax is

SRW.Set_Field_char (0,) and the output of the object that the current trigger is attached to will be replaced by. There are also RW.set_fileld_num and SRW.set_field_date for numeric or date fields.

While these options do work, they should only be used if a suitable NVL or DECODE statement in the original query is not possible as they are much, much slower to run. Also, note that this change of value only applies to the formatted output. It does not change the value held in the cursor and so can not be used for evaluating summary totals.

What is SQL* Loader and what is it good for?

SQL* Loader is a utility used for moving data from external files into the Oracle database. Its syntax is similar to that of the DB2 Load utility, but comes with more options. SQL* Loader supports various load formats, selective filters, and multi-table loads.

How does one use SQL* Loader?

One load data into the Oracle database by using the sqlldr (sqlload on some platforms) utility. Look at the following example:

sqlldr orauser/passwd control = loader.ctl

This is the control file, loader.ctl:

```
load data
  infile *
  replace
  into table departments
  (  dept     position (02:05) char(4),
     deptname position (08:27) char(20)
  )
begindata
  COSC  COMPUTER SCIENCE
  ENGL  ENGLISH LITERATURE
  MATH  MATHEMATICS
  POLY  POLITICAL SCIENCE
```

Can I load variable and fix length data records?

Yes, look at the following control file examples. In the first we will load delimited data (variable length):

```
LOAD DATA
INFILE *
INTO TABLE load_delimited_data
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(  data1,
   data2
)
BEGINDATA
11111,AAAAAAAAAA
22222,"A,B,C,D,"
```

If you need to load positional data (fixed length), look at the following control file example:

```
LOAD DATA
INFILE *
INTO TABLE load_positional_data
(  data1 POSITION(1:5),
   data2 POSITION(6:15)
```

```
)
BEGINDATA
11111AAAAAAAAAA
22222BBBBBBBBBB
```

Can I modify data as it loads into the database?

Yes, look at the following examples:

```
LOAD DATA
INFILE *
INTO TABLE modified_data
(  rec_no                "my_db_sequence.nextval",
   region                CONSTANT '31',
   time_loaded           "to_char(SYSDATE, 'HH24:MI')",
   data1       POSITION(1:5)  ":data1/100",
   data2       POSITION(6:15) "upper(:data2)"
)
BEGINDATA
11111AAAAAAAAAA
22222BBBBBBBBBB

LOAD DATA
INFILE 'mail_orders.txt'
BADFILE 'bad_orders.txt'
APPEND
INTO TABLE mailing_list
FIELDS TERMINATED BY ","
(  addr,
   city,
   state,
   zipcode,
   mailing_addr   "decode(:mailing_addr, null, :addr,:mailing_addr)",
   mailing_city   "decode(:mailing_city, null, :city,:mailing_city)",
   mailing_state
)
```

Can one load data into multiple tables at once?

Look at the following control file:

```
LOAD DATA
INFILE *
REPLACE INTO TABLE EMP
    WHEN empno != ' '
( empno  POSITION(1:4)   INTEGER EXTERNAL,
  ename  POSITION(6:15)  CHAR,
  deptno POSITION(17:18) CHAR,
  mgr    POSITION(20:23) INTEGER EXTERNAL
)
INTO TABLE proj
    WHEN projno != ' '
(  projno POSITION(25:27) INTEGER EXTERNAL,
   empno  POSITION(1:4)   INTEGER EXTERNAL
)
```

Can one selectively load only the data that you need?

Look at this example, (01) is the first character, (30:37) are characters 30 to 37:

```
LOAD DATA
APPEND  INTO TABLE db_trace_19980517
WHEN (01) <> 'H' and (01) <> 'T' and (30:37) = '19980517'
(
  region            CONSTANT '31',
  service_key        POSITION(01:11)   INTEGER EXTERNAL,
  call_b_no          POSITION(12:29)   CHAR
)
```

How do one-load multi-line records?

One can create one logical record from multiple physical records using one of the following two clauses:

CONCATENATE: - use when SQL* Loader should add the same number of physical records together to form one logical record.

CONTINUEIF - use if a condition indicates that multiple records should be treated as one, e.g. '#' in line 1.

How can get SQL* Loader to commit only at the end of the load file?

You can not, but by setting the ROWS= parameter to a large value, committing can be reduced. Make sure you have big rollback segments ready when you use a high value for ROWS=.

Can one improve the performance of SQL* Loader?

1. A very simple but easily overlooked hint, do not have any indexes and/or constraints (primary key) on your load tables during the load process. This will significantly slowdown load times even with ROWS= set to a high value.

2. Add the following option in the command line: DIRECT=TRUE. This will effectively bypass most of the RDBMS processing. However, there are cases when you can't use direct load. Refer to chapter 8 on Oracle server Utilities manual.

3. Turn off database logging by specifying the UNRECOVERABLE option. This option can only be used with direct data loads.

4. Run multiple load jobs concurrently.

What utilities does Oracle supply to download data to a flat file?

Oracle doesn't supply any data unload tools. However, you can use SQL* Plus to select and format your data and then spool it to a file:

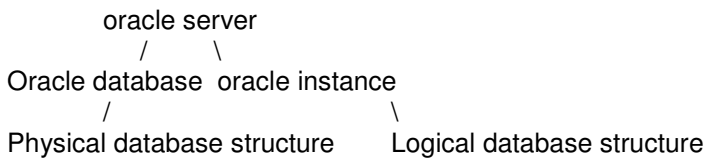
```
set echo off newpage 0 space 0 pagesize 0 feed off head off
trimspool on
spool oradata.txt
select col1 || ',' || col2 || ',' || col3
from   tab1
where  col2 = 'XYZ';
spool off
```

Alternatively use the UTL_FILE PL/SQL package:

List prerequisites to connect to oracle with a administrator previleges

1.User's O.S account has O.S previleges.

2.The user is granted the SYSDBA or SYSOPER previleges and database user the password files to authenticate database administration.
3.The database has a passsword for the internal login.



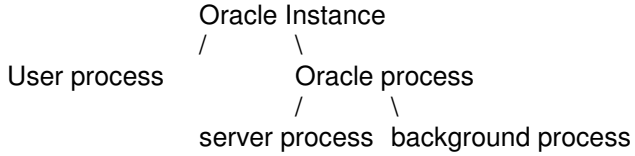
(An oracle database's physical structure is datermined by the O.S files that constitute database)

Each oracle database is made of 3 types of files:

- one or more datafile
- 2 or more redo log files
- 1 or more control files

Oracle Instance

System global area(SGA) is an area of memory used for database admin. shared by the database servers. The combination of the background process and memory buffers is called an instance.



A user process executes the code of an application program or an oracle tool.
Oracle process are server process and background process.
If the user and server process are on differrent computers of a network, if the user process connect to hares server process thro' dispatcher process, the user process and server process communicate thro' on using SQL* NET.

* Normally tablespace is online than offline.

b)Schemas and Schema objects

* Schema is collection objects
* Schema objects are the logical structures that directly refer to the database's data.
Schema objects include structures as tables,views,sequence.
(There is no relationship between a tablespace and schema).
1. table
2.view - A view can also be thought of as a stored query.
3.sequences
4.program unit
5.synonyms
6.Index,cluster and hash clusters
7.database link

Datablocks, Extents and segments

*one datablock corresponds to a specific number of bytes of physical database space on disk.

*An extent is a specific number of continumous datablock obtained in a single allocation used to store a specific.

1. **Which package construct must be declared and defined within the package body?**

- private procedure **(Answer)**

- public procedure
- boolean variable
- exception

2. **Examine this trigger:**

```
CREATE OR REPLACE TRIGGER UPD_PLAYER_STAT
                AFTER INSERT ON PLAYER
                FOR EACH ROW
BEGIN
          INSERT INTO PLAYER_BAT_STAT (PLAYER_ID,
SEASON_YEAR,AT_BATS,HITS)
          VALUES (player_id_seq.currval, 1997, 0,0);
END;
```

After creating this trigger, you test it by inserting a row into the PLAYER table. You receive this Error message

ORA-04091: table SCOTT.PLAYER is mutating, trigger/function may not see it.

Which explanation describes this error?

The storage of the PLAYER table is so fragmented that the trigger cannot run successfully.
Triggers cannot change data in the primary key, foreign key, or unique key of a constraining table. **(Answer)**
References to sequences are not allowed in triggers.
Triggers, unless otherwise specified, are read only.

3. **You have been granted the necessary privilege to invoke a function created by another developer. You are having problems calling this function from your procedure & want to contact the developer who created the function. Which table would you query to determine the owner of this function?**

- USER_OBJECTS
- ALL_FUNCTIONS
- ALL_OBJECTS **(Answer)**
- USER_FUNCTION_CODE

4. **Examine this package:**

```
CREATE OR REPLACE PACKAGE BB_PACK IS
V_MAX_TEAM_SALARY  NUMBER(12,2);
PROCEDURE ADD_PLAYER(V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY
NUMBER);
END BB_PACK;
/
 CREATE OR REPLACE PACKAGE BODY BB_PACK IS
   V_PLAYER_AVG  NUMBER(4,3);

  PROCEDURE UPD_PLAYER_STAT
  (V_ID IN NUMBER, V_AB IN NUMBER DEFAULT 4, V_HITS IN NUMBER)
  IS
  BEGIN
    UPDATE PLAYER_BAT_STAT SET AT_BATS = AT_BATS + V_AB,  HITS  = HITS +
V_HITS
     WHERE PLAYER_ID = V_ID;
     COMMIT;
   END UPD_PLAYER_STAT;

   PROCEDURE ADD_PLAYER
```

```
   (V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY NUMBER)
   IS
   BEGIN
    INSERT INTO PLAYER(ID,LAST_NAME,SALARY) VALUES (V_ID, V_LAST_NAME,
V_SALARY);
        UPD_PLAYER_STAT(V_ID,0,0);
  END ADD_PLAYER;

END BB_PACK;
/
```

The standalone procedure VALIDATE_PLAYER_STAT references a construct in this package. What will happen if the package specification changes?

- The outside procedure and the package body are invalidated. **(Answer)**
- Only the standalone procedure is invalidated.
- Only the package body is invalidated.
- There will be no effect.

5. **Which function will be created successfully in SQL* Plus?**

```
FUNCTION CALC_PLAYER_AVG (V_ID IN NUMBER) RETURN NUMBER IS
  V_AVG NUMBER;
BEGIN
   SELECT HITS/AT_BATS
    INTO V_AVG
    FROM PLAYER_BAT_STAT
    WHERE PLAYER_ID = V_ID;
    RETURN (V_AVG);
 END;
```

Answer:

```
CREATE OR REPLACE FUNCTION CALC_PLAYER_AVG
(V_ID IN NUMBER)
RETURN NUMBER
IS
V_AVG NUMBER;
BEGIN
       SELECT HITS/AT_BATS
       INTO V_AVG
       FROM PLAYER_BAT_STAT
       WHERE PLAYER_ID = V_ID;
       RETURN (V_AVG);
END;

CREATE DATABASE FUNCTION CALC_PLAYER_AVG
(V_ID IN NUMBER)
RETURN NUMBER
IS
V_AVG NUMBER;
BEGIN
      SELECT HITS/AT_BATS
      INTO V_AVG
      FROM PLAYER_BAT_STAT
      WHERE PLAYER_ID = V_ID;
      RETURN (V_AVG);
END;

REPLACE OR CREATE FUNCTION CALC_PLAYER_AVG
(V_ID IN NUMBER)
```

```
RETURN NUMBER
IS
      V_AVG NUMBER;
BEGIN
       SELECT HITS/AT_BATS
       INTO V_AVG
       FROM PLAYER_BAT_STAT
       WHERE PLAYER_ID = V_ID;
       RETURN (V_AVG);
END;
```

6. **The ADD_PLAYER, UPD_PLAYER_STAT and UPD_PITCHER_STAT procedures are grouped together in a package. A variable is shared among all these procedures and will also be read by a different package. Where should you declare this variable?**

   - PACKAGE BODY
   - DATABASE TRIGGER
   - PACKAGE SPECIFICATION **(Answer)**
   - each procedure's DECLARE section

7. **Which table should you query to check the status of a function?**

   - USER_PROCEDURES
   - USER_PROCS
   - USER_OBJECTS **(Answer)**
   - USER_PLSQL_UNITS

8. **Examine this trigger:**

```
CREATE OR REPLACE TRIGGER UPD_TEAM_SALARY
AFTER INSERT ON PLAYER
BEGIN
       UPDATE TEAM
       SET TOT_SALARY = TOT_SALARY + :NEW.SALARY
       WHERE ID = :NEW.TEAM_ID;
END;
```

   Which statement should be added if you want this trigger to execute just once for each INSERT operation on the PLAYER table?

   - FOR EACH ROW
   - FOR EVERY ROW
   - WHEN (NEW.%ROWCOUNT = 1)
   - No additional statement is required. **(Answer)**

9. *Which statement best describes the difference between auditing objects with triggers and auditing objects within the server?*

   - The server can only audit data manipulation statements.
   - Triggers can audit data manipulation, data retrieval, and data definition statements.
   - Triggers can only audit data manipulation statements **(Answer)**
   - Triggers can only audit data definition statements.

10. **Examine this function:**

```
CREATE OR REPLACE FUNCTION CALC_PLAYER_AVG
 (V_ID IN NUMBER)
```

```
RETURN BOOLEAN
IS
    V_AVG NUMBER;
BEGIN
        SELECT HITS / AT_BATS
        INTO V_AVG
        FROM PLAYER_BAT_STAT
        WHERE PLAYER_ID = V_ID;
        IF V_AVG > .333 THEN
           RETURN (TRUE);
        ELSE
           RETURN (FALSE);
        END IF;
END;
```

Why does invoking this function in SQL* Plus cause an error?

- The 'RETURN' statement cannot return BOOLEAN datatypes.
- The 'RETURN' statement can only return VARCHAR2 and NUMBER datatypes.
- The 'RETURN' statement cannot return BOOLEAN datatypes if invoked from SQL* Plus. **(Answer)**
- The function contains two 'RETURN' statements. Only one is allowed.

11. **Examine this package:**

```
CREATE OR REPLACE PACKAGE BB_PACK
IS
V_MAX_TEAM_SALARY  NUMBER(12,2);
PROCEDURE ADD_PLAYER(V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY
NUMBER);
END BB_PACK;
/
CREATE OR REPLACE PACKAGE BODY BB_PACK
IS
V_PLAYER_AVG  NUMBER(4,3);

PROCEDURE UPD_PLAYER_STAT
 (V_ID IN NUMBER, V_AB IN NUMBER DEFAULT 4, V_HITS IN NUMBER)
IS
BEGIN
   UPDATE PLAYER_BAT_STAT SET AT_BATS = AT_BATS + V_AB,  HITS  = HITS +
V_HITS
    WHERE PLAYER_ID = V_ID;
   COMMIT;
END UPD_PLAYER_STAT;

PROCEDURE ADD_PLAYER
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY NUMBER)
IS
BEGIN
   INSERT INTO PLAYER(ID,LAST_NAME,SALARY) VALUES (V_ID, V_LAST_NAME,
V_SALARY);
   UPD_PLAYER_STAT(V_ID,0,0);
END ADD PLAYER;

END BB PACK;
/
```

The standalone procedure VALIDATE_PLAYER_STAT references a construct in this package. What will happen if the package specification changes?

- The outside procedure and the package body are invalidated. **(Answer)**
- Only the standalone procedure is invalidated.
- Only the package body is invalidated.
- There will be no effect.

12. **One of your procedures must create a table during execution. Which Oracle supplied package must you use to accomplish this task?**

- DBMS_CREATE_OBJECT
- DBMS_SQL **(Answer)**
- DBMS_PIPE
- DBMS_TRANSACTION

13. **Examine this function creation statement:**

```
CREATE OR REPLACE FUNCTION TEAM_TAX
 (v_team_id  in player.team_id%type)
RETURN NUMBER
IS
v_team_tax_amnt  number(11,2);
BEGIN
      SELECT SUM(SALARY) * .07
      INTO v_team_tax_amnt
      FROM PLAYER
      WHERE TEAM_ID = V_TEAM_ID;
      RETURN V_TEAM_TAX_AMNT;
END;
```

Which two statements will successfully invoke this function in SQL* Plus? (Choose two.)

- EXECUTE TEAM_TAX(1)
- **VARIABLE G_TEAM_TAX_AMNT NUMBER**
  EXECUTE :G_TEAM_TAX_AMNT := TEAM_TAX(1)
- RUN TEAM_TAX(1)
- **SELECT NAME, TEAM_TAX(ID)**
  FROM TEAM
  WHERE TEAM_TAX(ID) > 300000;
- TEAM_TAX(ID);

14. **Examine this package:**

```
CREATE OR REPLACE PACKAGE BB_PACK
IS
V_MAX_TEAM_SALARY   NUMBER(12,2);
PROCEDURE  ADD_PLAYER(V_ID  IN  NUMBER,  V_LAST_NAME  VARCHAR2,  V_SALARY
NUMBER);
END BB_PACK;
/
CREATE OR REPLACE PACKAGE BODY BB_PACK
IS
V_PLAYER_AVG  NUMBER(4,3);

PROCEDURE UPD_PLAYER_STAT
 (V_ID IN NUMBER, V_AB IN NUMBER DEFAULT 4, V_HITS IN NUMBER)
IS
BEGIN
            UPDATE PLAYER_BAT_STAT
            SET AT_BATS = AT_BATS + V_AB,
```

```
            HITS  = HITS + V_HITS
            WHERE PLAYER_ID = V_ID;
            COMMIT;
END UPD_PLAYER_STAT;

PROCEDURE ADD_PLAYER
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY NUMBER)
IS
BEGIN
             INSERT INTO PLAYER(ID,LAST_NAME,SALARY)
             VALUES (V_ID, V_LAST_NAME, V_SALARY);
             UPD_PLAYER_STAT(V_ID,0,0);
END ADD_PLAYER;

END BB_PACK;
```

The V_PLAYER_AVG variable needs to be assigned a value determined from a calculation. You want this value to be assigned only when the package is first invoked.

In which construct should this calculation be placed?

- PACKAGE SPECIFICATION
- DECLARE SECTION of the first procedure declared in the package.
- PACKAGE BODY **(Answer)**
- PACKAGE EXCEPTION

15. **Examine this package:**

```
CREATE OR REPLACE PACKAGE BB_PACK
IS
V_MAX_TEAM_SALARY   NUMBER(12,2);
PROCEDURE ADD_PLAYER(V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY
NUMBER);
END BB_PACK;
/
CREATE OR REPLACE PACKAGE BODY BB_PACK
IS
V_PLAYER_AVG  NUMBER(4,3);

PROCEDURE UPD_PLAYER_STAT
 (V_ID IN NUMBER, V_AB IN NUMBER DEFAULT 4, V_HITS IN NUMBER)
IS
BEGIN
   UPDATE PLAYER_BAT_STAT
   SET AT_BATS = AT_BATS + V_AB,
   HITS  = HITS + V_HITS
   WHERE PLAYER_ID = V_ID;
   COMMIT;
   VALIDATE_PLAYER_STAT(V_ID);
 END UPD_PLAYER_STAT;

 PROCEDURE ADD_PLAYER
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY NUMBER)
 IS
 BEGIN
    INSERT INTO PLAYER(ID,LAST_NAME,SALARY)   VALUES (V_ID, V_LAST_NAME,
V_SALARY);
    UPD_PLAYER_STAT(V_ID,0,0);
END ADD_PLAYER;
```

END BB_PACK;

SCOTT is responsible for updating the statistics of players. He cannot add players. Which command must you issue to allow SCOTT the use of this package to accomplish his job?

- GRANT EXECUTE ON BB_PACK.UPD_PLAYER_STAT TO SCOTT;
- GRANT EXECUTE ON UPD_PLAYER_STAT TO SCOTT;
- UPD_PLAYER_STAT cannot be executed from outside this package. Therefore, SCOTT cannot use this package to accomplish his job. **(Answer)**
- GRANT EXECUTE ON BB_PACK.BODY.UPD_PLAYER_STAT TO SCOTT;

16. **Your can enter new ballplayers to the PLAYER table from different Oracle Forms applications and from an application written in C. For each new ballplayer, a record must be inserted into the PLAYER_BAT_STAT table. Which action should you perform to accomplish this requirement?**

- Create an additional function.
- Create an additional procedure.
- Create a database trigger on the PLAYER_BAT_STAT table.
- Create a database trigger on the PLAYER table. **(Answer)**

17. **Which table and column can you query to see all procedures and functions that have been marked invalid?**

- USER_OBJECTS table, STATUS column **(Answer)**
- USER_OBJECTS table, INVALID column
- USER_ERRORS table, STATUS column
- USER_ERRORS table, INVALID column

18. **Your procedure references a function created by another application developer. You need to contact this person to discuss the code contained in the function. Which table can you query to determine the owner of this function?**

- USER_DEPENDENCIES **(Answer)**
- USER_REFERENCES
- USER_CONSTRAINTS
- USER_LINKS

19. **The ADD_PLAYER procedure executes the UPD_PLAYER_STAT procedure. The UPD_PLAYER_STAT procedure updates the PLAYER_BAT_STAT table. SCOTT owns all objects. Which statement will find this indirect dependency between the ADD_PLAYER procedure and the PLAYER_BAT_STAT table?**

- EXECUTE DEPTREE_FILL('TABLE','SCOTT', 'PLAYER_BAT_STAT'); **(Answer)**
- EXECUTE FILL_DEPTREE('TABLE','SCOTT','PLAYER_BAT_STAT');
- SELECT * FROM USER_DEPENDENCIES WHERE TYPE = 'INDIRECT';
- SELECT * FROM USER_IND_DEPENDENCIES;

20. **The ADD_PLAYER procedure executes the UPD_PLAYER_STAT procedure. The UPD_PLAYER_STAT procedure updates the HITS column of the PLAYER_BAT_STAT table. What will happen if the HITS column of the PLAYER_BAT_STAT table is modified from NUMBER(3) TO NUMBER(4)?**

- Both the ADD_PLAYER and UPD_PLAYER_STAT procedures are marked invalid. **(Answer)**
- The ADD_PLAYER procedure is marked invalid.
- The UPD_PLAYER_STAT procedure is marked invalid.
- Neither procedure is marked invalid.

21. **Examine this package:**

```
CREATE OR REPLACE PACKAGE BB_PACK
IS
V_MAX_TEAM_SALARY   NUMBER(12,2);
PROCEDURE  ADD_PLAYER(V_ID  IN  NUMBER,  V_LAST_NAME  VARCHAR2,  V_SALARY
NUMBER);
END BB_PACK;
/
CREATE OR REPLACE PACKAGE BODY BB_PACK
IS
PROCEDURE UPD_PLAYER_STAT
 (V_ID IN NUMBER, V_AB IN NUMBER DEFAULT 4, V_HITS IN NUMBER)
IS
BEGIN
    UPDATE PLAYER_BAT_STAT
    SET AT_BATS = AT_BATS + V_AB,
    HITS  = HITS + V_HITS
    WHERE PLAYER_ID = V_ID;
    COMMIT;
END UPD_PLAYER_STAT;

PROCEDURE ADD_PLAYER
(V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY NUMBER)
IS
BEGIN
     INSERT INTO PLAYER(ID,LAST_NAME,SALARY)
     VALUES (V_ID, V_LAST_NAME, V_SALARY);
     UPD_PLAYER_STAT(V_ID,0,0);
END ADD_PLAYER;

END BB_PACK;
```

SCOTT is responsible for adding new ball players to the system. Which command must you issue to allow SCOTT the use of this package to accomplish his job?

- GRANT EXECUTE ON BB_PACK TO SCOTT; **(Answer)**
- GRANT EXECUTE ON ADD_PLAYER TO SCOTT;
- GRANT EXECUTE ON BB_PACK.ADD_PLAYER TO SCOTT;
- GRANT EXECUTE ON BB_PACK.BODY.ADD_PLAYER TO SCOTT;

22. **Examine this procedure:**

```
CREATE OR REPLACE PROCEDURE DELETE_PLAYER
 (V_ID IN NUMBER)
IS
BEGIN
       DELETE FROM PLAYER
       WHERE V_ID = 31;
       EXCEPTION
       WHEN STATS_EXIST_EXCEPTION
       THEN DBMS_OUTPUT.PUT_LINE
        ('Cannot delete this player, child records exist in PLAYER_BAT_STAT table');
END;
```

Which set of commands must be added to handle the non-predefined error: ORA-02292?

- STATS_EXIST_EXCEPTION NUMBER;
  PRAGMA EXCEPTION_INIT(STATS_EXISTS_EXCEPTION, -2292);

- STATS_EXIST_EXCEPTION EXCEPTION;
  PRAGMA EXCEPTION_INIT(-2292, STATS_EXISTS_EXCEPTION);
- **STATS_EXIST_EXCEPTION EXCEPTION;**
  PRAGMA EXCEPTION_INIT(STATS_EXISTS_EXCEPTION, -2292);
- STATS_EXIST_EXCEPTION EXCEPTION;
  PRAGMA EXCEPTION_INIT(STATS_EXISTS_EXCEPTION, 2292);

23. **Which statement will successfully create the procedure ADD_PLAYER in SQL* Plus?**

(Answer)

```
CREATE OR REPLACE PROCEDURE ADD_PLAYER
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2)
IS
BEGIN
   INSERT INTO PLAYER (ID,LAST_NAME) VALUES (V_ID, V_LAST_NAME);
   COMMIT;
END;

PROCEDURE ADD_PLAYER
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2)
IS
BEGIN
    INSERT INTO PLAYER (ID,LAST_NAME) VALUES (V_ID, V_LAST_NAME);
    COMMIT;
END;

REPLACE OR CREATE PROCEDURE ADD_PLAYER
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2)
IS
BEGIN
        INSERT INTO PLAYER (ID,LAST_NAME)
        VALUES (V_ID, V_LAST_NAME);
         COMMIT;
END;

CREATE OR REPLACE PROCEDURE (V_ID IN NUMBER, V_LAST_NAME VARCHAR2)
ADD_PLAYER
IS
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2)
BEGIN
    INSERT INTO PLAYER (ID,LAST_NAME) VALUES (V_ID, V_LAST_NAME);
    COMMIT;
END;
```

24. **Examine this package:**

```
CREATE OR REPLACE PACKAGE BB_PACK
IS
V_MAX_TEAM_SALARY   NUMBER(12,2);
PROCEDURE  ADD_PLAYER(V_ID  IN  NUMBER,  V_LAST_NAME  VARCHAR2,  V_SALARY
NUMBER);
END BB_PACK;
/
CREATE OR REPLACE PACKAGE BODY BB_PACK
IS

PROCEDURE UPD_PLAYER_STAT
 (V_ID IN NUMBER, V_AB IN NUMBER DEFAULT 4, V_HITS IN NUMBER)
IS
BEGIN
```

```
   UPDATE PLAYER_BAT_STAT
   SET AT_BATS = AT_BATS + V_AB,
   HITS  = HITS + V_HITS
   WHERE PLAYER_ID = V_ID;
   COMMIT;
END UPD_PLAYER_STAT;

PROCEDURE ADD_PLAYER
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY NUMBER)
IS
BEGIN
   INSERT INTO PLAYER(ID,LAST_NAME,SALARY)  VALUES (V_ID, V_LAST_NAME,
V_SALARY);
   UPD_PLAYER_STAT(V_ID,0,0);
END ADD_PLAYER;

END BB_PACK;
```

Which statement will successfully execute the ADD_PLAYER procedure from within SQL* Plus?

- EXECUTE BB_PACK.ADD_PLAYER(37,'Nettles',500000); **(Answer)**
- EXECUTE ADD_PLAYER(37,'Nettles',500000);
- RUN BB_PACK.ADD_PLAYER(37,'Nettles',500000);
- This procedure cannot be executed from SQL*Plus.

25. **Examine this package:**

```
CREATE OR REPLACE PACKAGE BB_PACK
IS
V_MAX_TEAM_SALARY  NUMBER(12,2);
PROCEDURE ADD_PLAYER(V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY
NUMBER);
END BB_PACK;
/
CREATE OR REPLACE PACKAGE BODY BB_PACK
IS

PROCEDURE UPD_PLAYER_STAT
(V_ID IN NUMBER, V_AB IN NUMBER DEFAULT 4, V_HITS IN NUMBER)
IS
BEGIN
   UPDATE PLAYER_BAT_STAT
   SET AT_BATS = AT_BATS + V_AB,
   HITS  = HITS + V_HITS
   WHERE PLAYER_ID = V_ID;
   COMMIT;
END UPD_PLAYER_STAT;

PROCEDURE ADD_PLAYER
(V_ID IN NUMBER, V_LAST_NAME VARCHAR2, V_SALARY NUMBER)
IS
BEGIN
   INSERT INTO PLAYER(ID,LAST_NAME,SALARY)  VALUES (V_ID, V_LAST_NAME,
V_SALARY);
   UPD_PLAYER_STAT(V_ID,0,0);
END ADD_PLAYER;

END BB_PACK;
```

Which statement will successfully execute the UPD_PLAYER_STAT procedure from within SQL* Plus?

- EXECUTE BB_PACK.UPD_PLAYER_STAT(31, 4, 2);
- RUN BB_PACK.UPD_PLAYER_STAT(31, 4, 2);
- UPD_PLAYER_STAT(31, 4, 2);
- This procedure cannot be executed from SQL* Plus. **(Answer)**

26. **Adding ball players to the PLAYER table is limited to the baseball season. You decide to create a database trigger.**

    Which trigger timing should you use to prevent users from inserting into this table during the off-season?

    on-change
    after
    before
    on-insert

27. **What is one benefit of using procedures and functions?**

- Procedures and functions are re-parsed for multiple users by exploiting shared SQL areas.
- Procedures and functions avoid re-parsing for multiple users by exploiting shared SQL areas. **(Answer)**
- Procedures and functions increase the number of calls to the database.
- Testing of procedures and functions requires the database to be restarted to clear out shared SQL areas for future access.

28. **Which statement must be added to make this trigger execute after updating the SALARY column of the PLAYER table?**

- AFTER UPDATE (SALARY) ON PLAYER
- AFTER SALARY UPDATE OF PLAYER
- AFTER UPDATE ON PLAYER
- AFTER UPDATE OF SALARY ON PLAYER **(Answer)**

29. **Examine this function:**

```
CREATE OR REPLACE FUNCTION CALC_PLAYER_AVG
 (V_ID IN NUMBER)
RETURN NUMBER
IS
   V_AVG NUMBER;
BEGIN
SELECT HITS / AT_BATS INTO V_AVG FROM PLAYER_BAT_STAT WHERE PLAYER_ID =
V_ID;
END;
```

Why does this command fail when executed?

- 'RETURN NUMBER' must be placed after the IS keyword..
- Functions can only return VARCHAR2 datatypes.
- **The return datatype function must specify a maximum length.**
- **Functions must return a value in the EXECUTABLE section.** (Answer)

30. **Which code is stored in the database when a procedure or function is created in SQL* Plus?**

- only SOURCE CODE
- both SOURCE CODE and P-CODE **(Answer)**
- only P-CODE
- neither SOURCE CODE or P-CODE

31. **The ADD_PLAYER procedure residing in your local database in Austin executes the UPD_PLAYER_STAT procedure residing in Chicago. The procedure in Chicago has just been changed and recompiled.**

    What will happen?

    - An error will occur next time the ADD_PLAYER procedure is executed. **(Answer)**
    - There will be no effect if the UPD_PLAYER_STAT procedure compiled successfully.
    - The ADD_PLAYER procedure will automatically recompile the next time you invoke it.
    - An error will occur next time the UPD_PLAYER_STAT procedure is executed.

32. **Examine this trigger:**

```
CREATE OR REPLACE TRIGGER UPD_PLAYER_STAT
AFTER INSERT ON PLAYER
FOR EACH ROW
BEGIN
   INSERT INTO PLAYER_BAT_STAT (PLAYER_ID, SEASON_YEAR,AT_BATS,HITS)
   VALUES (player_id_seq.currval, 1997, 0,0);
END;
```

    A user informs you that she is getting an error whenever she inserts a row into the PLAYER table. After researching the problem, you realize that the database trigger on the PLAYER table is inserting a row into the PLAYER_BAT_STAT table. The PLAYER_BAT_STAT table is offline.

    Which action should you take to allow users to insert into the PLAYER table until the PLAYER_BAT_STAT table is online?

    - Drop the trigger.
    - Disable the trigger. **(Answer)**
    - Alter the trigger to fire at a different time.
    - Alter the trigger, changing it from a row level to a statement level trigger.

33. **Examine this function:**

```
CREATE OR REPLACE FUNCTION CALC_PLAYER_AVG
 (V_ID IN NUMBER)
RETURN NUMBER
IS
      V_AVG NUMBER;
BEGIN
   SELECT HITS / AT_BATS INTO V_AVG FROM PLAYER_BAT_STAT WHERE PLAYER_ID
= V_ID;
   RETURN (V_AVG);
END;
```

    Which two statements will successfully invoke this function in SQL* Plus? (Choose two.)

    - EXECUTE CALC_PLAYER_AVG(31);
    - **SELECT CALC_PLAYER_AVG(31)**
      FROM PLAYER_BAT_STAT;
    - **VARIABLE G_AVG NUMBER**
      EXECUTE :G_AVG := CALC_PLAYER_AVG(31);

- CALC_PLAYER_AVG(31);
- START CALC_PLAYER_AVG(31);

34. **For every new ballplayer added to the PLAYER table, a record must be inserted into the PLAYER_BAT_STAT table. You have written a trigger to accomplish this task.**

    To which timing will this trigger be assigned?

    - STATEMENT
    - PRE-INSERT
    - AFTER **(Answer)**
    - BEFORE

35. **The ADD_PLAYER, UPD_PLAYER_STAT and UPD_PITCHER_STAT procedures are executed frequently together from different applications. What steps should be performed to logically group these procedures together in order to increase performance?**

    - Create a SQL* Plus script file that contains each procedure's CREATE statement
    - Create a package specification. Place the name of each procedure in the new specification.
    - Create a package specification and body with the source code of each procedure. Keep the stand-alone procedures for the specification to reference.
    - Create a package specification and body with the source code of each procedure. Drop the old stand-alone procedures. **(Answer)**

36. The ADD_PLAYER procedure calls the UPD_PLAYER_STAT procedure. Both procedures are INVALID.
    Which command can you issue to recompile both procedures?

    - ALTER PROCEDURE UPD_PLAYER_STAT COMPILE;
    - COMPILE PROCEDURE UPD_PLAYER_STAT;
    - COMPILE PROCEDURE ADD_PLAYER;
    - ALTER PROCEDURE ADD_PLAYER COMPILE; **(Answer)**

37. **Examine this procedure:**

```
CREATE OR REPLACE PROCEDURE ADD_PLAYER
 (V_ID IN NUMBER, V_LAST_NAME VARCHAR2)
IS
BEGIN
      INSERT INTO PLAYER (ID,LAST_NAME) VALUES (V_ID, V_LAST_NAME);
      COMMIT;
  END;
```

    SCOTT needs the privilege to invoke this procedure. Which set of privileges must you issue?

    - GRANT INSERT ON PLAYER TO SCOTT;
    - GRANT EXECUTE ON ADD_PLAYER TO SCOTT;
    - GRANT EXECUTE ON ADD_PLAYER TO SCOTT; **(Answer)**
    - GRANT INSERT ON PLAYER TO SCOTT;
    - GRANT SELECT ON PLAYER TO SCOTT;
    - GRANT SELECT ON ADD_PLAYER TO SCOTT;

38. **Examine this procedure created using v.2.3 of PL/SQL:**

```
CREATE OR REPLACE PROCEDURE UPD_BAT_STAT
 (V_ID IN NUMBER, V_AB IN NUMBER DEFAULT 4, V_HITS IN NUMBER)
IS
BEGIN
```

```
    UPDATE PLAYER_BAT_STAT
    SET AT_BATS = AT_BATS + V_AB,
                  HITS  = HITS + V_HITS
    WHERE PLAYER_ID = V_ID;
    COMMIT;
END;
```

Which two statements will successfully invoke this procedure from SQL* Plus? (Choose two.)

- EXECUTE UPD_BAT_STAT
- EXECUTE UPD_BAT_STAT(31)
- EXECUTE UPD_BAT_STAT(31,'FOUR','TWO')
- **EXECUTE UPD_BAT_STAT(V_AB => 4, V_ID => 31, V_HITS => 2)**
- **EXECUTE UPD_BAT_STAT(31,4,2)**

39. The ADD_PLAYER procedure must calculate the amount of social security tax that the team needs to pay. This algorithm is already written in a C program. You decide to call this program from the ADD_PLAYER procedure.
Which Oracle Procedure Builder built-in package must you use to accomplish this task?

- ORA_PROF
- ORA_FFI **(Answer)**
- TOOL_ENV
- STPROC

40. **This statement fails when executed:**

```
CREATE OR REPLACE TRIGGER CALC_TEAM_AVG
AFTER INSERT ON PLAYER
BEGIN
   INSERT INTO PLAYER_BAT_STAT (PLAYER_ID, SEASON_YEAR,AT_BATS,HITS)
VALUES (:NEW.ID, 1997, 0,0);
END;
```

Why?

- This is a statement level trigger and therefore cannot reference :new. **(Answer)**
- This is a row level trigger and therefore cannot reference :new.
- This is a statement level trigger and therefore cannot perform data manipulation statements.
- This is a row level trigger and therefore cannot perform data manipulation statements.

41. **Using the debugger in Procedure Builder, which action must you take to temporarily halt the execution of a procedure?**

- Raise an exception explicitly.
- Click on the STEP INTO icon.
- Click on the STEP OVER icon.
- Insert a breakpoint. **(Answer)**

SQL*PLUS QUESTIONS

Fill in the blanks :

1. No of User variables in SQL*PLUS is **1024**.

2. User can have **100** many numbers of variables per SQL command.

3. User can have **500** many number of lines (assuming 80 characters per line) per SQL command.

4. The size of PL/SQL buffer in Oracle 7 is **2k** and in Oracle6 is **512k**

5. **Start** command is used to run the contents of the specified command file.

6. The **intersect** operator is used to get only those rows that returned by both the query.

7. The **Grand** command is used to set the System privileges, Object privileges.

8. The **Savepoint** command is used to identify the point in a transaction to which you can later Rollback.

9. To perform one of these operations on your current transaction:

* Establish your current transaction as either a read only or a read-write transaction
* Assign your current transaction to a specified rollback segment

The **Set Transaction** command is used.

10. The **to-char** function is used to convert the number datatype to a value of varchar2 datatype.

11. The **Truncate** command is used to remove all rows in a Table or Cluster instantly.
Note : We can not truncate rows from a table, which is part of a cluster.
We cannot truncate rows from a table, which has a referenced integrity constraint.

12. The **Cluster** is a schema object that contains one or more tables that have one or more columns in common.

13. The **Create Role** command is used to set a set of privileges that can be granted to users or to other roles.

14. To perform one of these functions on an index, table, or cluster:

* To collect statistics about the object used by the optimizer and store them in the data dictionary.
* To delete statistics about the object from the data dictionary.
* To validate the structure of the object.
* To identify migrated and chained rows of the table or cluster.

The **Analyze** Command is used.

Select the Correct Answer:

1. An index can have as many as ___ Columns.

a] 255
b] 21
c] 16
d] None of the above

Ans: **16**

2. A number of columns in a table ranges from 1 to ____.

a] 255
b] 254
c] 030
d] None of the above

Ans: **254**

3. The maximum number of components in the **Decode** expression, including searches, results and default is

a] No limitation
b] 255

Ans: **255**

4. ___ Is an alternative name for a table, view, sequence, procedure, stored function, package, snapshot or another synonym.

a] Synonym
b] Data block
c] View
d] None of the above

Ans: **Synonym**

5. The _____ operator is used in character string comparisons with pattern matching

a] Between.. And
b] Equal operator
c] Set operator
d] Like

Ans: **Like**

6. _____ returns only one copy of each set of duplicate rows selected.

a] Unique
b] Distinct
c] Group By
d] None of the above

Ans: **Unique - (I think the answer is distinct.)**

7. _____ is used to lock the selected rows

a] Lock table
b] For update of
c] Object privileges
d] Row share

Ans: **For update of**

8. _____ Clause restricts the groups of rows returned to those groups for the specified condition id True

a] Where clause
b] Having Clause
c] Distinct
d] Exists

Ans: **Having**

9. The _____ option is used to return rows in a hierarchical order

a] Connect by start with
b] Order by

Ans: **Connect by start with**

10. The _____ function is used to return the number of bytes in the internal representation of expression

a] Length
b] Vsize
c] LengthLB
d] None of the above

Ans: **Vsize**

ORACLE QUESTIONS & ANSWERS	
Questions	**Answers**
What is a Database ?	A Database can be one of the two definitions: • A set of dictionary tables and user tables that are treated as a unit. • One or more operating system files in which ORACLE stores the tables, views, and other objects, also, the set of database objects used by a given application. • A database is a collection of interrelated data that are to be stored in a single location. It enables sharing of data among various users as and when required.
What is a Database system?	A Database system is a combination of an Instance and a Database. If the instance is started and connected to an open database, then the database is available for access by the users. A DBMS is a software system with capabilities to organize, manipulate and manage the data. Note:- A DBMS must be able to reliably manage a large amount of data in a multi-user environment so that many users can concurrently access the same data. A DBMS must also be secure from unauthorized access and provides efficient solutions for failure recovery.
What is an RDBMS ?	A relational database Management System (RDBMS) is a computer program for general purpose data storage and retrieval that organizes data into tables consisting of one or more units of information (rows), each containing the same set of data items (columns). ORACLE is a relational database management system.
What are the different Database models ?	• Hierarchical. • Networking. • Relational.

What is SQL ?	• S.Q.L - Structured Query Language. SQL is the ANSI industry standard language, used to manipulate information in a relational database and used in ORACLE and IBM DB2 relational database management systems. SQL is formally pronounced “sequel”, although common usage also pronounces it “SQL” • SQL is a set of commands that all programmers must use to access data within the tables of Database.
What are the benefits of SQL ?	1. It is flexible, Powerful and easy to learn. 2. It is a non-procedural language. It a] Processes set of records rather than just one at a time and b] Provides automatic navigation to the data. 3. It provides commands for a variety of tasks including : a] Querying data b] Creating, Updating and Replacing objects and Inserting, Updating and Deleting rows. 4. All RDBMS supports SQL Thus one can transfer the skills gained with SQL from one RDBMS to another. 5. Programs written in SQL are portable, they can often be moved from one database to another with little modification.
What is SQL*PLUS ?	SQL*PLUS is the ORACLE database language which includes ANSI standard SQL commands plus additional commands for accessing data in ORACLE database. SQL*PLUS is a Structured Query Language supported by Oracle. Through this only, we store, retrieve, edit, enter & run SQL commands and PL/SQL blocks. We can perform calculations , list column definitions, and format query results in the form of a query.
What is PL/SQL ?	It is a Procedural Language extension of SQL. It can contain any no of SQL statements integrated with flow of control statements. Thus it combine the Data Manipulating power of SQL with data processing power of Procedural language.
What are the different types of **SQL** commands ?	**DDL** (Data definition language) **DML** (Data manipulation language) **TCL** (Transact control language) **Session Control Statements. (** ALTER SESSION, ROLE) **System Control Statements. (** ALTER SYSTEM)
What is A DDL statements?	DDL statements are one category of SQL statements. DDL statements define (create) or delete (drop) database objects. Examples are create view, create table, create index, drop table and rename table. The other categories are DML statements and DCL statements.

What is a DML statements ?	DML statements are one category of SQL statements. DML statements, such as select, insert, delete and update, query and update the actual data. The other categories are DDL statements and DCL statements.
What are DCL statements ?	DML statements are one category of SQL statements. DCL statements such as, connect, grant select, grant update and revoke DBA, control access to the data and to the database. The other categories are DDL and DML statements.
What is a Transaction ?	• It can be defined as a logical unit of work. • A transaction is a sequence of SQL statements that ORACLE treats as a single unit. The set of statements is made permanent with the COMMIT statement. Part or all of a transaction can de undone with the ROLLBACK statement. • All changes to the database between successive COMMITS and / or ROLLBACK operations are called a transaction.
What is a Commit ?	• COMMIT commits any changes made to the database since the last COMMIT was executed implicitly or explicitly. WORK is optional and has no effect on usage. • To COMMIT means to make changes to data (inserts, updates and deletes) permanent. before changes are stored both the old and new data exists so that changes can be made, or so that the data can be restored to its prior state.("rollback"). When a user enters the ORACLE SQL Command COMMIT, all changes from that transaction are made permanent. • To end a transaction and make permanent all changes performed in the transaction. This command also **erases all savepoints** in the transaction and **release the transaction locks**
What is a Rollback ?	• A ROLLBACK discards part or all of the work you have done in the current transaction, since the last COMMIT or SAVEPOINT. • To undo work done in current transaction.
What is locking ?	To lock is to temporarily restrict other user's access to data. The restriction is placed on such data is called "a lock". The modes are SHARE, SHARE UPDATE,EXCLUSIVE,ROW SHARE AND ROW EXCLUSIVE. Not all locks can be acquired in all modes. EXCLUSIVE locks permit users to query the locked table but not to do anything else. No other user may lock the table. SHARED locks permit concurrent queries but no updates to the locked table. With a ROW SHARE or SHARE UPDATE lock, no users can lock the whole table for exclusive access, allowing concurrent access for all users to the table. The two types of locks are synonymous, and SHARE UPDATE exists for compatibility with previous versions of ORACLE. ROW EXCLUSIVE locks are similar to ROW SHARE but they prohibit shared locking, so only one user may access the table at the same time.

What is a Savepoint ?	The **Savepoint** is used to identify the point in a transaction to which you can later Rollback.
What is SHARE LOCK ?	A SHARE lock is one that permits other users to query data, but not to change it.
What is SHARE UPDATE LOCK ?	A SHARE UPDATE lock is one that permits other users to both query and lock data.
What is EXCLUSIVE LOCK ?	An EXCLUSIVE LOCK is one that permits other users to query data, but not to change it. It differs from the SHARE lock because it does not permit another user to place any type of lock on the same data; **several users may place SHARE locks** on the same data at the same time.
What is a ROW SHARE LOCK ?	With a ROW SHARE or SHARE UPDATE lock, no users can lock the whole table for exclusive access, allowing concurrent access for all users to the table.
What is a ROW EXCLUSIVE LOCK ?	ROW EXCLUSIVE locks are similar to ROW SHARE but they prohibit shared locking, so only one user may access the table at the same time.
What is a DEAD LOCK ?	A DEAD lock is a rare situation in which two or more user processes of a database cannot complete their transactions. This occurs because each process is holding a resource that the other process requires (such as a row in a table) in order to complete. Although these situations occur rarely, ORACLE detects and resolves deadlocks by rolling back the work of one of the processes.
What are INTEGRITY CONSTRAINTS ?	INTEGRITY CONSTRAINT is a rule that restricts the range of valid values for a column, it is placed on a column when the table is created.
What is REFERENTIAL INTEGRITY ?	REFERENTIAL INTEGRITY is the property that guarantees that values from one column depend on values from another column. This property is enforced through integrity constraints.
What is a PRIMARY KEY ?	The PRIMARY KEY is the column(s) used to uniquely identify each row of a table.
What is a FOREIGN KEY ?	A FOREIGN KEY is one or more columns whose values are based on the PRIMARY or CANDITATE KEY values from the database.
What is a UNIQUE KEY ?	A UNIQUE KEY is one or more columns that must be unique for each row of the table.
What is the difference between UNIQUE and PRIMARY KEY ?	The UNIQUE KEY column restricts entry of duplicate values but entry of NULL value is allowed. In case of PRIMARY KEY columns entry of duplicate as well as NULL value is restricted.

What is a SEQUENCE ?	A SEQUENCE is a database object used to generate UNIQUE INTEGERS for use as PRIMARY KEYS.
What is a VIEW ?	A View is a database object that is a logical representation of a table. It is derived from a table but has no storage space of its own and often may be used in the same manner as a table.
What is a SYNONYM ?	A SYNONYM is a name assigned to a table or view that may thereafter be used to refer it. If you access to another user's table, you may create a synonym for it and refer to it by the synonym alone, without entering the user's name as a qualifier.
What is a ROWID ?	ROWID is the logical address of a row, and it is unique within the database. The ROWID is broken into three sections: left, middle,, and right (corresponding to 00001F20,000C, AND 0001, just shown). The numbering is in hexadecimal notation. The left section is the block in the file, the middle is the row sequence number within the block(numbering starts with 0, not 1), and the right is the file number within the database. Note that the file numbers are unique within the whole database. The tablespace they are in is not relevant to the ROWID. ROWID can be selected, or used in a where clause, but cannot be changed by an insert, update, or delete. However it can change if the table it is in is exported and imported.
What is INDEX ?	INDEX is a general term for an ORACLE / SQL feature used primarily to speed execution an impose UNIQUENESS upon certain data. INDEX provides a faster access method to one table's data than doing a full table scan. There are several types of Indexes : UNIQUE INDEX, COMPRESSED INDEX, CONCATENATED INDEX. An Index has an entry for each value found in the table's Indexed field(s) (except those with a NULL value) and pointer(s) to the rows having that value.
What is an UNIQUE INDEX ?	An UNIQUE INDEX is an index that imposes uniqueness on each value in indexes. The index may be one column or concatenated columns.
What is a COMPRESSED INDEX ?	A COMPRESSED INDEX is an index for which only enough index information is stored to identify unique entries; information that an index stores with the previous or following key is "compressed" (truncated) and not stored to reduce the storage overhead required by an index.
What is CONCATENATED INDEX or KEY?	A CONCATENATED INDEX is one that is created on more than one column of a table. It can be used to guarantee that those columns are unique for every row in the table and to speed access to rows via those columns
What are CLUSTERS ?	A CLUSTER is a means of storing together data from multiple tables, when the data in those tables contains information and is likely to be accessed concurrently.

What is CLUSTER KEY or CLUSTER COLUMNS ?	A CLUSTER KEY is the column or columns that cluster tables have in common, and which is chosen as the storage / access key. For example two tables, **WORKER** and **WORKERSKILL**, might be clustered on the column name. A cluster key is the same thing as a cluster column.
What is CLUSTER INDEX ?	A CLUSTER INDEX is one manually created after a cluster has been created and before any DML (that is SELECT, INSERT, UPDATE AND DELETE)statements can operate on the cluster. This index is created on the CLUSTER KEY columns with the SQL statement CREATE INDEX. In ORACLE 7, you can define a hash cluster to index on the primary key.
What are EXCEPTIONS ?	Exceptions are the error handling routines of PL/SQL. The EXCEPTION section of a PL/SQL block is where program control is transferred whenever an exception flag is raised. Exception flags are either user-defined or system exceptions raised automatically by PL/SQL.
What are CURSORS ?	Cursor has two definitions : • A cursor is a marker such as a blinking square or line that marks your current position on a CRT screen. • Cursor is also a synonym for context area - a work area in memory where ORACLE stores the current SQL statement. For a query , the area in memory also includes column headings and one row retrieved by the SELECT statement.
What is NULL ?	A NULL value is one that is unknown, irrelevant, or not meaningful. Any ORACLE data type can be NULL. NULL in a number data type is not the same as zero. **The default value for a field in ORACLE is NULL**.
What is EXPRESSION ?	An expression is any form of a column. This could be a literal, a variable, a mathematical computation, a function, or virtually any combination of functions and columns whose final result is a single value, such as a string, a number, or a value.
What is a CONDITION ?	A Condition is an expression whose value evaluates to either TRUE or FALSE, such as AGE > 16.
What is a PROFILE ?	A PROFILE is a collection of settings on ORACLE7 that limit database resources.
What are ROLES ?	A ROLE is a set of privileges that an ORACLE7 user can grant to another user or to a role. ORACLE version 6 privileges DBA, CONNECT, AND RESOURCE have become system-supplied roles in ORACLE7, and there are also two new roles for importing and exporting a database. ORACLE has five system-supplied roles : CONNECT,RESOUCE,DBA,EXP_FULL_DATABASE, IMP_FULL_DATABASE.

What is a SEGMENT ?	A SEGMENT is another way to classify the space allocated to a table, index, or cluster. A table has one segment that consists of all of its extents. Every index has one segment similarly defined. A cluster has atleast two segments, one for its data and one for its cluster key index.
What is TABLE SPACE in ORACLE ?	TABLE SPACE is a file or set or files that is used to store ORACLE data. An ORACLE database is composed of the SYSTEM tablespace and possibly other tablespaces.
What are PCTUSED and PCTFREE parameters ?	PCTFREE is a portion of the data block that is not filled by rows as they are inserted into a table. but is reserved for future updates made to the rows in that block. PCTUSED is the percentage of space in a data block, which ORACLE attempts to fill before it allocates another block.
CLIENT	A Client or front-end database application acts as an interface between the user and the Database. It also checks for validation against the data entered by the user. CLIENT is a general term for a user , software application, or computer that requires the services, data, or processing of another application or computer.
SERVER	A Database server or Back End is used to manage the Database tables optimally among multiple clients who concurrently request the server for the same data. It also enforces data integrity across all client applications and controls database access and other security requirements. SERVER system is the configuration of the ORACLE when a remote user accesses ORACLE via SQL*NET.
What is a SESSION ?	A SESSION is a sequence of events that happens between the time a user connects to SQL and the time he or she disconnects.
What is an INSTANCE ?	An INSTANCE is everything required for ORACLE to run: background processes (programs), memory, and so on. An INSTANCE is the means of accessing a database.
What is a BACKROUND PROCESS ?	A BACKROUND process is one of the processes used by an instance of multiple-process ORACLE to perform and coordinate tasks on behalf of concurrent users of the database. The base process are named ARCH (achiever),DBWR (database writer), LGWR (log writer), PMON (process monitor), and SMON (system monitor), and exists as long as an instance does.

What is a BLOCK in ORACLE ?	Basic unit of storage (physical and logical) for all ORACLE data. The number of blocks allocated per ORACLE table depends on the table space in which the table is created. The ORACLE block size varies by operating system and may differ from the block size of the host operating system.. Common block sizes are 512 bytes (characters) and 2048 bytes. A **Block** is a logical container for items. It is also a separate object, with its own set of properties. The properties of the block determine how end users interact with the interface items it contains.
What is the use of ROLLBACK segment ?	A ROLLBACK segment is a storage space within a table space that holds transaction information used to guarantee data integrity during a ROLLBACK and used to provide read consistency across multiple transactions.
What is READ CONSISTENCY ?	READ CONSISTENCY is a state that guarantees that all data encountered by a statement / transaction is a consistent set throughout the duration of the statement / transaction.
What is SGA ?	SGA is a shared storage area in main or virtual memory (depending on your operating system) that is the center of ORACLE activity while the database is running. The size of the SGA (and performance of the system) depends on the values of the variable **init.ora** parameters. The SGA provides communication between the user and the background processes.
What is SYSTEM USERID? What does it have ?	SYSTEM is one of the DBA users that is created when the database system is installed and initialized (the other is SYS). **While SYS owns most of the data dictionary tables, SYSTEM owns the views created on those base tables.**
What is SYS USERID ? What does it have ?	SYS is one of the DBA users that is created when the database system is installed and initialized (the other is SYSTEM). **SYS owns most of the data dictionary tables, SYSTEM owns the views created on those base tables.**
What is a **Datadictionary** in ORACLE ?	The DATA DICTIONARY is a comprehensive set of tables and views owned by the DBA users SYS and SYSTEM, which activates when ORACLE is initially installed, and is a central source of information for the ORACLE RDBMS itself and for all users of ORACLE. The tables are automatically maintained by ORACLE, and holds a set of views and tables containing information about the database objects, users, privileges, events, and use.
What is **Sqldba** ?	SQL * DBA is an ORACLE utility used by DBAs while performing database maintenance and monitoring.
What are **Database files** ?	A DATABASE file is simply any file used in a database. A database is made up of one or more tablespaces, which in turn are made up of one or more database files.

What is a **Control file** ? What is its significance ?	A CONTROL file is a small administrative file required by every database, necessary to start and run a database system. A control file is paired with a database, not with an instance. Multiple identical control files are preferred to a single file, for reasons of data security.
What is an **INIT** file ? What is its significance ?	**init.ora** is a database system parameter file that contains numerous settings and file names used when a system is started using the CREATE DATABASE , START UP, or SHUT DOWN command.
What does a INSERT statement do ?	INSERT adds one or more new rows to the table or view.
What does an UPDATE statement do ?	Updates (changes) the values in the listed columns in the specified table.
What does a DELETE statement do ?	DELETE deletes all rows that satisfy condition from table.
What does a SELECT statement do ?	SELECT retrieves rows from one or more tables (or views or snapshots), either as a command, or as a subquery in another SQL command (with limitations), including SELECT,INSERT,UPDATE and DELETE. ALL means that all rows satisfying the conditions will be returned (this is the default). DISTINCT means that only rows that are unique will be returned: any duplicates will be weeded out first.
What is **Startup** and **Shutdown** ?	STARTUP is the process of starting an instance, presumably with the intent of mounting and opening a database in order to make a database system available for use. To SHUTDOWN is to disconnect an instance from the database and terminate the instance.
What is **Mounting** of database ?	To MOUNT a database is to make it available to the database administrator.
What is **Two Phase - Commit** ?	ORACLE7 manages distributed transactions with a special feature called TWO PHASE - COMMIT. TWO PHASE - COMMIT guarantees that a transaction is valid at all sites by the time it commits or roll back. All sites either commit or rollback together, no matter what errors occur in the network or on the machines tied together by the network. You don't need to do anything special to have your applications use a TWO PHASE - COMMIT.
What are **Snapshots** ?	A SNAPSHOT is a means of creating a local copy of remote data. A snapshot can be used to replicate all or part of a single table, or to replicate the result of a query against multiple tables. The refreshes of the replicated data can be done automatically by the database (at time intervals you specify) or manually.
What are **Triggers** ?	A DATABASE TRIGGER is a stored procedure associated with a table that ORACLE7 automatically executes on one or more specified events (BEFORE or AFTER an INSERT,UPDATE or DELETE) affecting the table. Triggers can execute for the table as a whole or for each affected row in the table.

What are **Packages** ?	A PACKAGE is a PL/SQL object that groups PL/SQL type, variables, SQL cursors, exceptions, procedures, and functions. Each package has a specification and a body. The specification shows the object you can access when you use the package. The body fully defines all the objects and can contain additional objects used only for the internal workings. You can change the body (for example, by adding procedures to the packages) without invalidating any object that uses the package.
What are **Packaged Procedures** ?	A PACKAGED PROCEDURE is a built-in PL/SQL procedure that is available in all forms. Each packaged procedure executes a SQL*FORMS function, such as moving to a field or executing a query.
What are **Restricted Packaged Procedures** ?	Any PACKAGED PROCEDURE that affects the basic functions of SQL*FORMS is a RESRICTED PACKAGED PROCEDURE. You should use restricted packaged procedure only in KEY-TRIGGERS, USER-NAMED TRIGGERS that are invoked by KEY-TRIGGERS, and ON_NEW_FIELD_INSTANCE triggers. You should not use restricted packaged procedures in any of the following types of triggers. • On-error, On-Database-Record, On-delete, On-insert, On-Lock, On-Message, On-New-Record, On-Remove-record, On-Update, On-Validate-Field, and On-validate-Record triggers. • Post-Change triggers. • Pre- and Post- Field, Pre- and Post- Record, Pre- and Post-Block, Pre- and Post-Form triggers. • Pre- and Post-Query triggers. • Pre- and Post-Insert, Pre- and Post-Update, Pre- and Post-Delete, Pre- and Post-Commit triggers. • User-Named triggers that are invoked by any of the above triggers.
What are **Unrestricted Packaged Procedures** ?	Any PACKAGED PROCEDURE that does not interface with the basic functions of SQL*FORMS is an UN- RESRICTED PACKAGED PROCEDURE. You can use unrestricted packaged procedures in any type of trigger. The following list shows the unrestricted packaged procedures: Abort_Query, Anchor_View, Bell, Break, Call, Call_query, Default_Value, Display_Error, Display_field, Display_page, Edit_field, Erase, Execute_Trigger, Help, Hide_Page, Host, Lock_Record, Message, Move_View, Pause, Print, Redisplay, Resize_View, Set_Field, Show_keys, Show_page, Synchronize.
What are **Pseudo Columns** in ORACLE ?	A PSEUDO COLUMN is a “column” that yields a value when selected, but which is not an actual column of the table. An example is ROWID or SYSDATE.
What is a **Schema** ?	A SCHEMA is a collection of objects. SCHEMA objects are logical structures that directly refer to the database’s data. SCHEMA objects include structures such as **tables, views, synonyms, sequences, indexes, clusters, and stored procedures and data links.**

What are the major aspects of the **Relational Database Management System** ?	The Relational model has three major aspects: **Structures** : Structures are well-defined objects that store the data of the database. Structures and the data contained within them can be manipulated by operations. **Operations** : Operations are clearly defined actions that allow the user to manipulate the data and structure of the database. The operation on a database must adhere to a pre-defined set of integrity rules. **Integrity rules** : Integrity rules are the laws that govern which operations are allowed on the data and structure of a database. Integrity rules protect the data and the structures of a database.
What are the benefits of **Relational Database Management System** ?	RDBMS offers benefits such as : 1] Independence of physical data storage and logical database structure. 2] variable and easy access to all data. 3] Complete flexibility in database design. 4] Reduced data storage and redundancy.
What is a **Database Structure** ?	An ORACLE database structure has both a physical and logical structure. **<u>Physical database structure :</u>** An ORACLE database physical structure is determined by the operating system files that constitute the database. Each ORACLE database is comprised of three types of files: one or more data files, two or more redolog files, and one or more control files. The files of a database provide the actual physical storage of the database information. **<u>Logical database structure:</u>** An ORACLE database's logical structure is determined by • One or more tablespaces. • The database's schema objects (e.g. tables, views, indexes, clusters, sequences and stored procedures) The logical storage structures, including tablespaces, segments, and extents, dictate how the physical space of a database is used. the schema objects and the relationships among them form the relational design of the database.
What are the **LOGICAL STRUCTURES** ?	1. **<u>Tablespaces:</u>** A database is divided into logical storage units called tablespaces. A tablespaces used to group related logical structures together. For example , tablespaces commonly group all of an applications objects simplify certain administrative operations. 2. **<u>Databases,Tablespaces and Datafiles:</u>** • Each database is logically divided into one or more tablespaces. • One or more datafiles are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace • The combined size of a tablespace's data files is the total storage capacity of the tablespace (**SYSTEM has 2MB storage capacity while data has 4MB**) • The combined storage capacity of a database's tablespaces is the total storage capacity of the database. (**6MB)**

What is **On-line** and **Off-line** tablespaces ?	An tablespace can be On-line or Off-line. A tablespace is normally On-line so that users can access the information within the tablespace. A tablespace can be Off-line to make a portion of the database unavailable while allowing normal access to the remainder of the database.
What are **Hash clusters** ?	Hash clusters are also cluster table data in a manner similar to normal clusters. However, a row is stored in a hash cluster based on the result of applying a hash function to the row's cluster key value.
What are **Database Links** ?	A database link is a named object that describes a path from one database to the other. Database links are implicitly used when a reference is made to a global object name in a distributed database.
What are **Datablocks** ?	At the finest level of granularity, an ORACLE database data is stored in datablocks . One datablock corresponds to a specific number of bytes of physical database space on the disk. A datablock size is specified for each ORACLE database when the database is created. A database uses and allocates free database space in ORACLE datablocks.
What are **Extents** ?	The next level of logical database space is called an extent. An extent is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information.
What are **Segments** ?	The level of logical database storage above an extent is called a segment. A segment is a set of extents allocated for a certain logical structure. The different types of segments include : **Data segment :** Each non-clustered table has a data segment. All of the table's data is stored in the extents of its data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment. **Index segment :** Each index has an index segment that stores all of its data. **Rollback segment :** In or more rollback segments are created by the database administrator for a database to temporarily store "undo" information. This information is used to generate read-consistent database information, during database recovery and to rollback uncommitted transactions for users. **Temporary segment :** Temporary segments are created by ORACLE when a SQL statement needs a temporary work area to complete execution. when the statement finishes execution the temporary segment's extents are returned to the system for future use.
What is **Application Partitioning** ?	PL/SQL is the language used for both client-side Oracle forms applications and server-side database triggers and stored procedures and there is a PL/SQL engine in both Oracle forms Runform and the Oracle7 Server. This means that you can take advantage of application partitioning to execute application code on either the client or the server. Application partitioning allows you to optimize performance and resource usage by storing and executing procedures either locally or at the server, which makes the most sense for your particular application and configuration.

Explain the **Physical structure** of the **Oracle** database ?	The physical structure of an ORACLE database includes **datafiles, redolog files and control files.** **1. <u>Datafiles:</u>** Every ORACLE database has one or more physical data files. A database's data files contains all the database data. The data of logical database structures such as tables and indexes is physically stored in the data files allocated for a database. The characteristics of data files are : <u>A datafile can be associated with only one database, once created, a data file cannot change in size and one or more data files form a logical unit of database storage called a tablespace.</u> **<u>Note:</u>** Modified or new data is not necessarily written to a data file immediately. To reduce the amount of disk output and increase performance, data is pooled in memory and written to the appropriate data file all at once, as determined by the DBWR background process of ORACLE. **2. <u>Redo log files:</u>** Every ORACLE database has a set of two or more **Redo log files**. The set of redo log files for a database is collectively known as the **Database's redolog.** **The primary function of a redo log is to record changes made to data.** Should a failure prevent modified data to be written to the data files , the changes can be obtained from the redo log and the work is never lost. Thus redo log files are critical in protecting a database against failures. The process of applying the redo log during a recovery operation is called **Rolling forward.** To protect against failures of the redo log itself, ORACLE allows a mirrored redo log so that two or more copies of the redo log can be maintained on different disks. **3. <u>Control files:</u>** Every ORACLE database has a **Control file.** A control file records the physical structure of the database. For example, it contains the following information : **Database name, names and locations of a database's data files and redolog files and the time stamp of database creation.**
	Every time an **instance** of an ORACLE is started, its control file is used to identify database and the redo log files that must be opened for database operation to proceed. If the physical makeup of the database is altered (for example, if a new data file or redo log file is created), the database's control file is automatically modified by the ORACLE to reflect the change. **<u>Note:</u>** **A database's control file is also used if database recovery is necessary.**

Explain the **Memory Structures** of the Oracle database ?	**Oracle** creates and uses memory structures to complete several jobs. For example, memory is used to store program code being executed and data that is shared among users. Several basic memory structures are associated with Oracle; the system global area. (which includes the database and redolog buffers, and the shared pool) and the program global areas. **a) <u>System global area:</u>** The SGA is a shared memory region allocated by Oracle that data and information for one Oracle instance. An SGA and the Oracle background processes constitute an Oracle Instance. The SGA is allocated when an instance starts and deallocated when the instance shuts down. Each instance that is started has its own SGA. The data in the SGA is shared among the users currently connected to the database. For optimal performance , the entire SGA should be as large as possible to store as much data as possible in memory and minimize disks I/O. The information stored within the SGA is divided into several types of memory structures, including the database buffers, redo log buffers and the shared pool. These area have fixed size and are created during instance startup. **<u>1. Database Buffer Cache :</u>** **Database buffers of the SGA store the most recently used blocks of database data**; the set of database buffers in an instance is the database buffer cache. These buffers can contain modified data that has not yet been written to disk. **Because the most recently used data is kept in memory, less disk I/O is necessary and performance is increased.**
	<u>2. Redo log buffer:</u> **The redo log buffer of the SGA stores redo entries - a log of changes made to the database.** The redo entries stored in the redo log buffers are written to an online redo log file, which is used if database recovery is necessary. **Its size is static.** **<u>3. Shared Pool:</u>** **The shared pool is a portion of the SGA that contains shared SQL constructs such as shared SQL areas.** A shared SQL area is required to process every unique SQL statement submitted in a database. **A shared SQL area contains information such as the parse tree and execution plan for the corresponding statement.** A single shared SQL area is used by multiple application that issue the same statement leaving more control over cursors. **<u>4. Cursors:</u>** **A cursor is a handle (a name or pointer) for the memory associated with a specific statement**. Although most Oracle Users rely on the automatic handling of the Oracle Utilities, the programmatic interfaces offer application designers more control over cursors.

	b) Program Global Area: The PGA is a memory buffer that contains data and control information for a server process. A PGA is created by Oracle when a server process is started. The information in a PGA depends on the configuration of Oracle.
What is a **Process ?**	**A Process is a "thread of control" or a mechanism in a operating system that can execute a series of steps.** Some operating system use the term **job as task.**
Explain the **types of Processes used by Oracle** ?	An Oracle database system has two types of processes : **1) User Process.** **2) Oracle Process.** **User Process :** A user process is created and maintained to execute the software code of an application program (such as a Pro *C program) or an Oracle tool (such as SQL* DBA). The user processes also manages the communication with the server processes through the program interface. **Oracle Processes:** **Oracle processes are called by other processes to perform functions on behalf of the invoking process.** The different types of Oracle processes and their specific functions are as follows : **Server Process:** Oracle creates server processes to handle requests from connected user processes. A server process is in charge of communicating with the user process and interacting with the Oracle to carry out requests of the associated user process. Oracle can be configured to vary the number of user processes per server process. In a dedicated server configuration, a server process handles requests for a single user process. A multi- threaded server configuration allows many users processes to share a small number of server processes and maximizing the utilization of available system resources.
	Background Processes: Oracle creates a set of background processes for each Oracle Instance. They consolidate functions that would otherwise be handled by multiple Oracle programs running for each user processes. The background processes asynchronously perform I/O and monitor other Oracle processes to provide increased parallelism for better performance and reliability.
	An SGA and the Oracle background processes constitute an Oracle Instance. Each Oracle instance may use several background processes. They are **DBWR,LGWR,CKPT,SMON,PMON, ARCH, RECO, Dnnn AND Lckn.**

What is a **Database Writer ?**	**The DBWR writes modified blocks from the database buffer cache to the datafiles.** Because of the way Oracle performs logging, DBWR does not need to write blocks when a transaction commits. Instead, DBWR is optimized to minimize disk writes. In general, DBWR writes only when more data needs to be read into the SGA and too few database buffers are free. The least recently used data is written to the datafile first.
What is a **Log Writer?**	**The log writer writes redo log entries to disk.** Redo log data is generated in the redo log buffer of the SGA. As transactions commit and the log buffer fills, LGWR writes redo log entries into an on-line redo log file.
What is a **Checkpoint ?**	**At specific times, all modified database buffers in the SGA are written to the data files by DBWR; this event is called a Checkpoint.** The checkpoint process is responsible for signaling DBWR at checkpoints and updating all the data files and control files of the database to indicate the most recent checkpoint. CKPT is optional; if CKPT is not present, LGWR assumes the responsibilities of CKPT.
What is a **System Monitor ?**	**The System monitor performs instance recovery at instance startup.** SMON cleans up temporary segments that are no longer in use and recovers dead transactions skipped during crash and instance recovery because of file read or off-line errors. These transactions are eventually recovered by SMON when the tablespace or file is brought back. SMON also coalesces free extents within the database, to make free space contiguous and easier to allocate.
What is a **Process monitor?**	**The process monitor performs process recovery when a user process fails.** PMON is responsible for cleaning up the cache and freeing resources that the process was using. PMON also checks on the Dispatcher and server processes and restarts them if they have failed.
What is an **Achiever ?**	**The Achiever copies the on-line redo log files to archival storage when they are full.** ARCH is active only when a database's redo log is used in ARCHIEVELOG mode.
What is a **Recoverer?**	**The recoverer is used to resolve distributed transactions that are pending due to a NETWORK or system failure in a distributed database.**
What is a **Dispatcher ?**	**Dispatchers are optional background processes, present only when a <u>Multi-threaded</u> server configuration is used.** Atleast one dispatcher process is created for every communication protocol in use (D000,...Dnnnn). Each dispatcher process is responsible for routing requests from connected user processes to available shared server processes and returning the response back to the appropriate user processes.

What is a **LOCK (LCKn)?**	**Upto ten lock processes (LCK0..LCK9) are used for inter- instance locking when the Oracle parallel server is used.**

1. What is a View? Why is it required to define a View ?

A View is a database object that is a logical representation of a table. It is derived from a table but has no storage space of its own and often may be used in the same manner as a table.

Advantage: 1. Security 2. Complex query can be replaced.

2. Can we create a View without a table?

Yes, Using the FORCE option in the CREATE VIEW syntax.
Ex: CREATE FORCE VIEW view_name as SELECT column name, columnname.. FROM table_name;

3. What is the difference between a SYNONYM and a VIEW ?

A SYNONYM is a name assigned to a table or view that may thereafter be used to refer it. If you access to another user's table, you may create a synonym for it and refer to it by the synonym alone, without entering the user's name as a qualifier.

A View is a database object that is a logical representation of a table. It is derived from a table but has no storage space of its own and often may be used in the same manner as a table.

Difference: A View can be based on MULTIPLE Tables whereas a SYNONYM is based on a single object only.

4. What is SNAPSHOT? What is a SNAPSHOT LOG ?

A SNAPSHOT is a means of creating a local copy of remote data. A snapshot can be used to replicate all or part of a single table, or to replicate the result of a query against multiple tables. The refreshes of the replicated data can be done automatically by the database (at time intervals you specify) or manually. Snapshot Log is the table associated with the Master Table of the Snap shot.

5. What is a DATABASE trigger? What is a DATABASE Procedure?

A DATABASE TRIGGER is a stored procedure associated with a table that ORACLE7 automatically executes on one or more specified events (BEFORE or AFTER an INSERT,UPDATE or DELETE) affecting the table. Triggers can execute for the table as a whole or for each affected row in the table.

A PACKAGED PROCEDURE is a built-in PL/SQL procedure that is available in all forms. Each packaged procedure executes a SQL*FORMS function, such as moving to a field or executing a query.

6. How to show MESSAGES in PROCEDURES for debugging purposes?

DBMS_OUTPUT_PACKAGE allows you to use 3 debugging functions within your package. You must use "SET SERVER OUTPUT ON" before executing the procedure object you will be debugging.

PUT -	Puts multiple O/P's on same line.
PUT_LINE	Puts each O/P on a separate line.
NEW_LINE	Used with PUT; Signals the end of current O/P line.

7. What is the difference between DATABASE trigger and DATABASE procedure?

DATABASE triggers are executed automatically in response to specific events. But the DATABASE procedures are to be explicitly invoked to execute the code contained in them.

8. What is a CURSOR?

A work area in memory where ORACLE stores the current SQL statement. For a query , the area in memory also includes column headings and one row retrieved by the SELECT statement.

9. What are the attributes of IMPLICIT CURSOR?

%ISOPEN, %ROWCOUNT, %FOUND and %NOTFOUND.

Attribute	DML STATEMENT	
	RETURNS ROW	RETURNS NO ROW
%ISOPEN	FALSE	FALSE
%ROWCOUNT	TRUE	FALSE (ZERO)
%FOUND	TRUE	FALSE
%NOTFOUND	FALSE	TRUE

10. Can we pass a PARAMETER to CURSOR ? What is SQL%ROWCOUNT ?

We can pass parameter to CURSOR. E.g.: OPEN CUSOR('VASAN'). SQL%ROWCOUNT is used to count the number of rows returned by an SQL DML statement. It will return zero if the DML statement doesn't return any row.

11. How to write a SQL statement that should have a best RESPONSE TIME ?

Use the _____ in the optimizer hint inorder to obtain a best response time. Use "FIRST_ROW" - Cost based Optimizer Hint.

12. What are OPTIMIZER HINTS ?

Specifies a hint string that Oracle Forms passes on to the RDBMS optimizer when constructing queries. Using the optimizer can improve the performance of database transactions.

13. What is the difference between %TYPE and %rowtype?

%TYPE provides the datatype of a variable, constant or column. It is useful when you declare a variable that refers to a database column in the table.

%ROWTYPE attribute is based on a record variable that has the same structure as a row in a table or view or as a row fetched from a cursor.

14. Can we define structure like objects in PL/SQL?

[If the structure is what we define in 'C' then we can create objects of type structure using RECORD variable available in PL/SQL.]

Yes, Using the PL/SQL tables. PL/SQL tables are temporary array like objects used in a PL/SQL block. PL/SQL tables can have one column and a primary key. The column data type can belong to any scalar data type, but the primary key must only belong to the type binary integer.

Size - UNLIMITED.

15. Can we use a function inside an INSERT statement?

Yes. E.g.: INSERT INTO EMP(COMM) VALUES (SAL*0.05) WHERE DEPTNO = 20;

16. What is TRUNCATE table?

TRUNCATE table is a DDL command used to remove all the rows from the specified table or cluster instantly.
E.g.: TRUNCATE TABLE table_name;

Advantage over DELETING:

- It is a DDL statement and generates NO ROLLBACK information.
- Doesn't fire the tables DELETE TRIGGER.
- Truncating the master table of a snapshot doesn't record any changes in the tables snapshot log.
- It's more convenient than dropping and recreating the table.
- D/R invalidates the table's dependent objects than truncating the object.
- D/R requires you to REGRANT the privileges on the table while truncating doesn't.
- D/R requires you to RECREATE the INDEXES, INTEGRITY CONSTRAINTS, TRIGGERS and STORAGE PARAMETER while truncating doesn't.

17. What is ROWID? What are its components?

ROWID is the logical address of a row, and it is unique within the database. The ROWID is broken into three sections: left, middle and right (corresponding to 00001F20,000C AND 0001 just shown). The numbering is in hexadecimal notation. The left section is the block in the file, the middle is the row sequence number within the block(numbering starts with 0, not 1), and the right is the file number within the database. Note that the file numbers are unique within the whole database. The tablespace they are in is not relevant to the ROWID. ROWID can be selected, or used in a where clause, but cannot be changed by an insert, update, or delete. However it can change if the table it is in is exported and imported.

18. What is the difference between REPLACE and TRASLATE?

Syntax : REPLACE(string, if, then)

REPLACE replaces a character or characters in a string with 0 or more characters, *if* is a character or characters. Everytime it appears in a string, it is by the contents of *then*.

E.g.: REPLACE('ADAH','A','BLAH') - BLAHDBLAHH (Result)

Syntax: TRANSLATE(string, if, then)

TRANSLATE looks at each character in string, and then checks *if* to see if that character is there, if it is, TRANSLATE notes the position in *if* where it found the character, and then looks the same position in *then.* Whatever character it finds there it substitutes the character in *string*

E.g.: TRANSLATE('RAMESH','RAM','SUR') - SURESH(Result)

19. What is a LEVEL?

LEVEL is a pseudo column, used with CONNECT BY. It is equal to 1 for a root, 2 for a child of root, 3 for a child of a child of a root and so on.

20. What is anonymous block in PL/SQL?

The text of an Oracle Forms trigger is an anonymous PL/SQL block. It consists of three sections:

- A declaration of variables, constants, cursors and exceptions which is optional.
- A section of executable statements.
- A section of exception handlers, which is optional.

Syntax:

```
DECLARE     --- declarative statements (optional)
BEGIN       --- executable statements (required)
EXCEPTION   --- exception handlers (optional)
END;
```

21. Name any ORACLE defined EXCEPTION?

- CURSOR_ALREADY_OPEN.
- NO_DATA_FOUND.
- INVALID_NUMBER.

22. Can we define our OWN EXCEPTION? How to raise it?

In the DECLARATION part define a variable of type *exception*. In the execution part call the exception using RAISE *exception_name*. In the exception part handle the exception using WHEN *exception_name*.

23. What is a PRAGMA?

It is a directive to the COMPILER, rather than a piece of executable code. Even though it appears in the program, it is not executable. It gives instructions to the compiler.

24. Difference between CHAR and VARCHAR2?

CHAR (*size*) - It is a fixed length character data, *size* characters long. It is padded with BLANKS ON RIGHT to the full length of *size*. DEFAULT - 1 bytes, MAX - 255 bytes.
VARCHAR2 (*size*) - It is a variable length char string having a maximum of *size* bytes. MAX -2000 bytes.

25. What is a CURSOR FOR LOOP?

The CURSOR FOR LOOP lets you implicitly OPEN a cursor, FETCH each row returned by the query associated with the cursor and CLOSE the cursor when all rows have been processed.

26. What is the possible CONSTRAINTS defined on a TABLE?

NOT NULL, UNIQUE KEY, PRIMARY KEY, FOREIGN KEY and CHECK constraints.

27. What is APPLICATION PARTITIONING?

PL/SQL is the language used for both client-side Oracle forms applications and server-side database triggers and stored procedures and there is a PL/SQL engine in both Oracle forms Runform and the Oracle7 Server. This means that you can take advantage of application partitioning to execute application code on either the client or the server.

Application partitioning allows you to optimize performance and resource usage by storing and executing procedures either locally or at the server, which makes the most sense for your particular application and configuration.

28. Difference between a STORED PROCEDURE and a STORED FUNCTION?

Unlike procedures, FUNCTIONS returns a VALUE to the caller. This value is returned through the RETURN *command/keyword* within the function.

Functions don't use the IN, OUT | IN OUT arguments, which are available for PROCEDURES.

29. How to RUN PROCEDURES from SQL PROMPT?

Use EXECUTE *Procedure_name* command.

30. How to TRAP ERRORS in procedures?

Use SHOW_ERRORS. this will display all the errors associated with the most recently created procedural object. This command will check the VIEW_ERRORS data dictionary for the ERRORS associated with the most recent compilation attempt for that procedural object. SHOW_ERRORS will display the LINE and COLUMN NO. for each error, as well as the text of the error message.

E.g.: SELECT LINE, POSITION,TEXT FROM USER_ERRORS WHERE

```
NAME = 'balance_check' AND
TYPE = PROCEDURE/FUNCTION/PACKAGE
 ORDER BY  SEQUENCE;
```

NOTE: We can use ALL_ERRORS & DBA_ERRORS to view errors.

TRAPPING ERORS:

DBMS_OUTPUT package allows you to use 3 debugging functions within your package. You must set 'SERVER OUTPUT ON' before executing the procedure object you will be debugging.

PUT	-	Puts multiple outputs on same line.
PUT_LINE	-	Puts each o/p on a separate line.
NEW_LINE	-	Used with PUT; Signals the END of current o/p line.

31. When do we get a MUTATING ERROR?

This happens with TRIGGERS. It occurs when a trigger is trying to update a row, which is being used currently. The usual fix involves either use of VIEWS or TEMPORARY TABLES so the database is selecting from one while updating the other.

32. How to DISABLE REFERENTIAL INTEGRITY?

Use the DIABLE option in CREATE TABLE or ALTER TABLE or using

```
DISABLE { { UNIQUE (column) (column)... PRIMARY KEY |
          CONSTRAINT } [CASCADE] | ALL TRIGGERS;
```

NOTE: For disabling REFERENTIAL INTEGRITY we have to include CASCADE option.

33. How to know what are all the CONSTRAINTS present on a table?

- Using the USER_CONSTRAINTS view we can get the type of constraints declared on a table.
- Use ALL_CONSTRAINTS to list the constraints on all of the tables that the user has access.
- DBA_CONSTRAINTS lists all of the constraints in the database.

34. What is MASTER - DETAIL relationship? Can we write a master-detail relationship programs without using the settings at design time. If so how?

It is an association between TWO BASE TABLE blocks - a MASTER block and a DETAIL block. The relationship between the blocks reflects a PRIMARY KEY - FOREIGN KEY relationship between the tables on which the blocks are based.

Yes. Using the SET_RELATION property.

35. What does BUFFER RECORDS option and ARRAY SIZE parameter?

ARRAY SIZE - Specifies the minimum no. of records that get fetched each time forms goes to the database.

BUFFER RECORDS - Specifies the minimum no of records that should be placed in memory when records are fetched from the database. Even if you specify a low value of 3, the minimum per form is slightly over 300.

36. During VALIDATION WHAT CHECKS are done with respective to FIELDS / ITEMS ?

- Data Type
- Maximum Length
- Fixed Length
- Required
- Range Low value / Range High value.

37. What is the difference between PRIMARY KEY and UNIQUE KEY?

- The UNIQUE KEY column restricts entry of duplicate values but entry of NULL value is allowed.
- In case of PRIMARY KEY columns entry of duplicate as well as NULL value is restricted.

38. What is the DIFFERENCE between PRE-QUERY and POST-QUERY ?

- PRE-QUERY fires ONLY ONCE during EXECUTE-QUERY or COUNT-QUERY processing, just before Oracle Forms constructs and issues the SELECT statement to identify rows that match the query criteria.
- POST-QUERY fires each time for records placed on the block list of records.

38. When do you use ON-DATABASE-RECORD trigger?

Use an ON-DATABASE-RECORD to perform an action every time a record is first marked as an INSERT or UPDATE. This trigger fires, as soon as Oracle Forms determines through validation and the record should be processed by the next post or commit as an INSERT or UPDATE

39. What are RESTRICTED PACKAGED PROCEDURES? Why are they restricted from using?

Any PACKAGED PROCEDURE that affects the basic functions of SQL*FORMS is a RESRICTED PACKAGED PROCEDURE. You should use restricted packaged procedure only in KEY-TRIGGERS, USER-NAMED TRIGGERS that are invoked by KEY-TRIGGERS, and ON_NEW_FIELD_INSTANCE triggers. You should not use restricted packaged procedures in any of the following types of triggers.

- On-error, On-Database-Record, On-delete, On-insert, On-Lock,
- On-Message, On-New-Record, On-Remove-record, On-Update,
- On-Validate-Field, and On-validate-Record triggers.
- Post-Change triggers.
- Pre- and Post- Field, Pre- and Post- Record, Pre- and Post-Block, Pre- and Post-Form triggers.
- Pre- and Post-Query triggers.

- Pre- and Post-Insert, Pre- and Post-Update, Pre- and Post-Delete, Pre- and Post-Commit triggers.
- User-Named triggers that are invoked by any of the above triggers.

40. What is the DIFFERENCE between EXPLICIT CURSOR & IMPLICIT CURSOR?

Issuing a SELECT statement automatically opens IMPLICIT CURSORS, but the EXPLICIT cursors are to be opened using OPEN, fetching is done using FETCH and closing using CLOSE.

41. What is the difference between ROWID and ROWNUM?

ROWID is the logical address of the row, whereas ROWNUM returns the sequence no. in which the row was retrieved when first fetched from a table.

42. What is the RESULT of the statement?

SELECT EMPNO, NAME, SAL FROM EMP WHERE ROWNUM >2;
Result: 0, No rows will be selected.

43. How do you evaluate performance?

Using SQL TRACE. It is a utility that can monitor and report on database performance when one or more queries are run against the database. It is used to gather statistics when running the query (i.e.) reports on CPU time spent on the query, the total no. of rows processed and statistics related to parsing and cache performance.

44. What will EXPLAIN PLAN gives?

It is a utility that shows how Oracle will access data for a given query. Use EXPLAIN PLAN to determine the effective way to write queries and decide whether to INDEX CERTAIN COLUMNS or TO USE CLUSTERS.
It shows:

1] The type of query processed; SELECT, INSERT, UPDATE or DELETE.
2] The cost assigned by the COST BASED OPTIMIZER if it is in use.
3] The steps that are necessary to return the data.
4] The internal operations that were performed for each step.
5] The object accessed for each step.

45. How do you analyze TKPROF?

TKPROF filename.tra O/P file EXPLAIN = USR/PWD0

46. What parameter variables to be set to use TKPROF?

SQL PROF

47. How many types of locking are there?

There are 5 types of locks. To lock is to temporarily restrict other user's access to data. The restriction is placed on such data is called "a lock". The modes are SHARE, SHARE UPDATE,EXCLUSIVE,ROW SHARE AND ROW EXCLUSIVE. Not all locks can be acquired in all modes.

48. What is a SHARE LOCK?

A SHARE lock is one that permits other users to query data, but not to change it.

49. What is a SHARE UPDATE LOCK?

A SHARE UPDATE lock is one that permits other users to both query and lock data.

50. What is an EXCLUSIVE LOCK?

An EXCLUSIVE LOCK is one that permits other users to query data, but not to change it. It differs from the SHARE lock because it does not permit another user to place any type of lock on the same data; several users may place SHARE locks on the same data at the same time.

51. What are ROWSHARE, SHAREUPDATE and ROW EXCLUSIVE locks?

With a ROW SHARE or SHARE UPDATE lock, no users can lock the whole table for exclusive access, allowing concurrent access for all users to the table. The two types of locks are synonymous, and SHARE UPDATE exists for compatibility with previous versions of ORACLE. ROW EXCLUSIVE locks are similar to ROW SHARE but they prohibit shared locking, so only one user user may access the table at the same time.

52. What is a DEAD LOCK?

A DEAD lock is a rare situation in which two or more user processes of a database cannot complete their transactions. This occurs because each process is holding a resource that the other process requires (such as a row in a table) in order to complete. Although these situations occur rarely, ORACLE detects and resolves deadlocks by rolling back the work of one of the processes.

53. How do you analyze which resources has locked for what?

Use MONITOR SESSION.

54. How to kill a SESSION?

ALTER SESSION KILL ID, NUMBER FROM SQLDBA;

55. What are USER_EXITS?

It is a utility in SQL*FORMS for making use of HOST 3 GL languages for the purpose like ONLINE PRINTING etc.

56. When will you use the trigger WHEN-NEW-FORM-INSTANCE?

At FORMS STARTUP Oracle navigates to the first navigable item in the first navigable block. This trigger fires after successful completion of any Navigational trigger (i.e.) It will not fire if the control returns to the CALLING FORM from the CALLED FORM.

Usage: For initialization at FORMS STARTUP.

57. What is an INDEX? Why are indexes used in a table?

INDEX is a general term for an ORACLE / SQL feature used primarily to speed execution an impose UNIQUENESS upon certain data. INDEX provides a faster access method to one table's data than doing a full table scan. There are several types of Indexes :

UNIQUE INDEX, COMPRESSED INDEX and CONCATENATED INDEX. An Index has an entry for each value found in the table's Indexed field(s) (except those with a NULL value) and pointer(s) to the rows having that value.

58. What is a UNIQUE INDEX?

An UNIQUE INDEX is an index that imposes uniqueness on each value in indexes. The index may be one column or concatenated columns.

59. What is a COMPRESSED INDEX?

It is an index, for which only enough index information is stored to identify unique entries; information that an index stores with the previous or following key is “compressed” (truncated) and not stored to reduce the storage overhead required by an index.

60. What is a CONCATENATED INDEX?

It is one that is created on more than one column of a table. It can be used to guarantee that those columns are unique for every row in the table and to speed access to rows via those columns

61. What is a UNION, UNION ALL, INTERSECTION and MINUS operator?

- The UNION operator returns ALL DISTINCT ROWS selected by either query.
- The UNION ALL operator returns ALL ROWS selected by either query including duplicates.
- The INTERSECTION operator returns ONLY ROWS that are COMMON to both the queries.
- The MINUS operator returns ALL DISTINCT ROWS selected only by the first query and not by the second.

62. What does ‘GROUP BY’ statement does?

GROUP BY statement causes a SELECT statement to produce ONE SUMMARY ROW for all selected rows that have identical values in one or more specified column or expressions. Each expression in the SELECT clause must be one of the following :

1] A CONSANT
2] A Function without parameters
3] A GROUP function like SUM , AVG.
4] Matched IDENTICALLY to a expression in the ‘GROUP BY’ clause.

63. In 2 SELECT statements
SELECT A FROM DUAL; and
SELECT B FROM DUAL;
What will be the difference in using ‘UNION’ and ‘UNION ALL’?

UNION returns all distinct rows selected by either of the query, whereas UNION ALL returns ALL ROWS selected by either query including duplicates.

64. Give one example where you will use DATABASE TRIGGERS?

For AUDITING purposes we use database triggers.

65. Do you have any idea about ROW-CHAINING? How will you resolve the issue if there is row chaining in a table?

When a row NO LONGER FITS WITHIN THE DATABLOCK, it is stored in more than one database block, and that therefore has several row pieces.

Resolving: Use ANALYZE to identify chained rows and also provides statistics on the chained rows.

E.g.: ANALYZE ledger LIST CHAINED ROWS INTO CHAINED_ROWS:
(CHAINED_ROWS is a user-defined table)

For creating chained_rows run the UTLCHAIN.SQL script.

66. What is an OPTIIMIZER?

OPTIMIZER is a utility used to determine how to access data requested in the query by the USER or APPLICATION PROGRAM. The output of an optimizer is EXECUTION PLAN.

67. How the Oracle in case of a query does OPTIMIZATION?

1] RULE based, and 2] COST based.

68. What is RULE based optimization and COST based optimization?

RULE based optimization USES A FIXED SET OF RULES to determine how to access the data.

COST based optimization USES STASTISTICS STORED IN THE DATA DICTIONARY WITH CERTAIN RULES to determine how to access the data.

Two modes – a] ALL_ROWS, B] FIRST_ROW.

With the help of ALTER SESSION SET OPTIMIZER_GOAL = ALL_ROWS / FIRST_ROW, We can alter the modes of cost based optimizer.

---

69. The KEYWORD comes into the mind immediately when we talk about security in ORACLE 7.0?

GRANT.

Syntax: GRANT privileges (SELECT, INSERT, UPDATE, DELETE, ALTER and INDEX) ON object TO user WITH
GRANT OPTION;

70. What KEWORD is used to withdraw the PRIVILEGE you have granted to other user?

REVOKE

Syntax: REVOKE privileges ON object FROM users;

71. What is SINGLE INSTANCE?

A single instance can run on a single machine.

72. What is MULTIPLE INSTANCES?

A SINGLE MACHINE can run more than one instance at a time. Each instance is connected to its own database.

73. What is DISTRIBUTED PROCESSING?

Different instances on different machines can communicate with each other using DATABASE LINKS and the DISTRIBUTED option. Oracle supports full two-phase commits which means that inserts, updates and deletes can occur on REMOTE database via a network running SQL*Net.

74. What is PARALLEL PROCESSING?

The Oracle parallel server allows multiple instances to share a single database on a shared disk system. The instance can run on a parallel computer or on different computers in a cluster.

75. Difference between SQL and PL/SQL?

SQL	PL/SQL
1. It is flexible, powerful and easy to learn.	1. PL/SQL block can contain any no. of SQL statements combined with the following : A. Flow of control statements such as IF THEN,

	ELSE, EXIT and GOTO. B. Repetition statements such as FOR LOOP and WHILE LOOP. C. Assignment statements such as X := Y + Z
2. It is a non-procedural language. A. It processes set of records rather than just one at a time. B. Provides automatic navigation to the Data.	2. PL/SQL allows you to logically group a set of statements and send them to the RDBMS as a single block.
3. It provides command for a variety of tasks including: A. Querying Data. B. Creating, Updating and Replacing objects C. Inserting, Updating and Deleting.	3. Procedural Capabilities.
4. All RDBMS Supports SQL. Thus one can transfer the skills gained with SQL from one RDBMS to another. Programs written in SQL are portable, they can often be moved from one database to another with little modification.	4. Improved Performance, Enhanced Productivity, portability and Integration with RDBMS.
5. ANSI industry standard language, used to manipulate information in a relational database	5. It is a procedural language extension to Oracle's SQL language

76. How to fetch description of a code in the base table block where code is a base table field and the description is a non-base table field?

Use SELECT with INTO clause to fetch the description value into the NON-BASE table field.

77. What is the purpose of OUTER JOIN?

An OUTER JOIN returns all the rows returned by simple join as well as those rows from one table that do not match any row from the other table. The symbol (+) represents the outer join.

78. Difference between EQUI JOIN and OUTER JOIN?

EQUI JOIN returns rows from both the tables provided they both have the same *column_name* in the where clause. The symbol (=) represents the EQUI JOIN. For OUTER JOIN see previous answer.

79. Define NORMALIZATION?

NORMALIZATION is the process of putting things right, making them normal. It is a part of analysis necessary to understand a business, and build a useful application.

The normalization of data ensures the following:

- Minimization of duplication of data.
- Providing flexibility to support different functional requirements.
- Enabling the model to be translated to database design.

Following are the steps involved in normalization:

- Ensure that all the ENTITIES are uniquely identified by a combination of attributes.
- Remove repeated attributes or group of attributes, to place the entities in the first normal form.

- Remove attributes that are dependent on only part of the identifier.
- Remove attributes that are dependent on attributes, which are not part of the identifier.

80. Define REFERENTIAL INTEGRITY?

REFERENTIAL INTEGRITY is the property that guarantees that values from one column depend on values from another column. This property is enforced through integrity constraints. Referential integrity is the automatic enforcement of referential constraints that exists between a reference table and a referencing table. When referential integrity is enforced , the value of a foreign key exists as a primary key value in the reference table.

81. Explain OUTER JOIN with example?

SELECT DEPT.DEPTNO, DNAME, JOB, ENAME FROM DEPT, EMP WHERE DEPT.DEPTNO = EMP.DEPTNO (+) AND DEPTNO IN (30,40) ORDER BY DEPT.DEPTNO;

82. Explain with example how to use a select statement with GROUP BY HAVING clause? (or) Where and when is the HAVING clause used and what does it have?

The HAVING clause is coded after the GROUP BY clause in the query that is summarizing results by one or more grouping columns. The HAVING clause behaves the same as the WHERE clause except that it is used to specify the conditions each returned group must satisfy. If one row in the group fails the condition of the HAVNG clause, the entire group is not returned as part of the result.

Ex: SELECT MAX (CUSTID), REPID FROM CUSTOMER GROUP BY REPID HAVING COUNT (*) > 2;

83. How do you TUNE SQL statements?

Use OPTIMIZER HINTS for tuning SQL statements.

83. What is the advantage of ENFORCE KEY?

ENFORCE KEY field characteristic indicates the source of the value that SQL*FORMS uses to populate the field

84. What is the Purpose of ERASE command?

ERASE removes an indicated global variable & releases the memory associated with it

85. What do you mean by RI (Referential Integrity)?

A referential integrity constraint designates a column or combination of columns as a foreign key and establishes a relationship between that foreign key and a specified primary or unique key, called the referenced key. In this relationship, the table containing the foreign key is called the child table and the table containing the referenced key is called the parent table.

Note: A foreign key column cannot be of datatype LONG or LONG RAW.

86. What is the difference between a view and a snapshot?

View : Virtual table
Snapshot : A snapshot is a table that contains the results of a query of one or more tables or views, often
Located on a remote database.

87. What is the difference between Constraints and Database triggers?

Constraints : Set of business rules, which are predefined in Oracle.

Database triggers: Set of complex business rules that can be defined & enforced by the programmer.

88. What is the advantage of using packages?

A package is an encapsulated collection of related program objects stored together the database. Program objects in the sense - procedures, functions, variables, constants, cursors & exceptions.

Using packages is an alternative to creating procedures and functions as stand-alone schema objects. Packages have many advantages over stand-alone procedures and functions:

- Packages allow you to organize your application development more efficiently.
- Packages allow you to grant privileges more efficiently.
- Packages allow you to modify package objects without recompiling dependent schema objects.
- Packages allow Oracle7 to read multiple package objects into memory at once.
- Packages can contain global variables and cursors that are available to all procedures and functions in the package.
- Packages allow you to overload procedures or functions.
- Overloading a procedure means creating multiple procedures with the same name in the same package, each taking argument of different number or datatype.

89. How will you declare a private procedure in a package?

In a package
- Public objects (declared in the package), can be referenced outside the package as well as by other objects in the package.
- Private objects (defined in the package body), can only be referenced by other objects in the package. They cannot be referenced outside the package.
- Private procedure or other objects - defined only in package body.
- Public procedure or other objects - declared only in package.

90. Does oracle allows stored procedures overloading?

Yes, only through packages.

91. When the package will be initialized?

Once any objects first called, package get initialized (may be).

92. What do you mean by a parameterized cursor?

93. How will you find out number of rows inserted after an INSERT statement?

Using SQLcursor % ROWCOUNT

94. What do you mean by cluster and what is the advantage of using the cluster?

A cluster is a schema object that contains one or more tables that all have one or more columns in common.

Advantage: Rows of one or more tables that share the same value in these common columns are physically stored together within the database. Clustering provides more control over the physical storage of rows within the database. Clustering can reduce both the time it takes to access clustered tables and the space needed to store the table.

95. What is the use of an index?

An index is a database object that contains an entry for each value that appears in the indexed column(s) of the table or clusters and provides direct, fast access to rows. An index can contain a

maximum of 16 columns. Unlimited indexes can be created for a table provided that the combination of columns differs for each index. You can create more than one index using the same columns provided that you specify distinctly different combinations of the columns. Nulls are not indexed.

96. How will you force the query to look at the index while searching for a record?

Using the indexed column in the where clause.

Example: SELECT * FROM EMP WHERE DEPTNO = 10;

The above query use an index created on the DEPTNO column.

97. How will you avoid using indexes while searching for a record?

In where clause, manipulating the data type of the column will avoid using index.

Example: SELECT * FROM EMP WHERE TO_CHAR (DEPTNO) = 10;

The above query does not use an index created on the DEPTNO column, b'cos the data type of DEPTNO is NUMBER.

Also, if we change the combination of multiple columns indexed in the where clause.

Example: SELECT * FROM EMP WHERE DEPTNO = 10 AND SAL = 5000;

The above query does not use an multiple column index created on the SAL, DEPTNO column.

Also, if we use other functions in where clause.

Example: SELECT * FROM EMP WHERE DEPTNO IN (10,20);

The above query does not use an index created on the DEPTNO column.

98. What do you mean by self-referential integrity constraint?

Foreign key, which references the primary key of the same table, is called self-referential integrity.

99. What kind of locks is allowed in Oracle?

Row, Table & Database level locks are admitted in Oracle. Page level lock is not supported by Oracle.

100. What is the use of SET READONLY?

Specifies that no deletes, inserts, or updates can be performed.

101. What is the use of explain plan command?

This command insert a row describing each step of the execution plans of a particular table into a specified another table.

102. What do you mean by rolling forward?

103. How oracle handles distributed transactions?

104. What is the use of PCTFREE and PCTUSED and they can be used effectively?

- **PCTUSED -** specifies the limit that Oracle7 uses to determine when additional rows can be added to a cluster's data block. The value of this parameter is expressed as a whole number and interpreted as a percentage.
- **PCTFREE -** specifies the space reserved in each of the cluster's data blocks for future expansion. The value of the parameter is expressed as a whole number and interpreted as a percentage.

---

105. What the ROWID column is made of?

ROWID is a unique identification for each row in a table. It is a pseudo column. Other pseudo columns are ROWNUM, LEVEL, CURVAL & NEXTVAL. ROWID column is made of BLOCK.ROW.FILE

Where,
BLOCK: A hexadecimal string identifying the data blocks of the data file containing the row. The length of this
String may vary depending on your operating system.
ROW : A four-digit hexadecimal string identifying the row in the data block.

The first row in the block has the number 0. File is a hexadecimal string identifying the database file containing the row. The first data file has the number 1. The length of this string may vary depending on your operating system.

Example:
Consider this ROWID value:

0000000F.0000.0002

The row corresponding to this ROWID is the first row (0000) in the fifteenth data block (0000000F) of the second data file (0002).

106. What are the logical storage units of oracle database? How are they mapped with physical storage?

Logical storage units: Tables, Views, Indexes, Tablespaces, Segments, Extents, Synonyms, etc.

107. What do you mean by row chaining?

108. After deleting a row from a table, will the rowid used by the deleted row be reused?

No.

109. What do you mean by SGA?

System Global Area

110. What are the background processes of oracle database?

111. What do you mean by an instance?

112. How the uncommitted transactions are recovered after the failure of two-phase commit?

---

113. What is the roll of redo log file and rollback segment?

114. How will you use START WITH and CONNECT BY clauses with a select statement?

To define a hierarchical relationship in a query, you must use the START WITH and CONNECT BY clauses.

115. Is it possible to use a stored function in a select statement? If yes, what needs to be taken care in the stored function?

116. How many triggers are possible against a table?

12 TRIGGERS

117. What do you mean by trigger mutation?

When you query the same table in which the trigger has been defined, it is trigger mutation.

118. What do you mean by a PL/SQL table?

PL/SQL tables are objects of type TABLE, which are modeled as (but not the same as) database tables. PL/SQL tables use a primary key to give you array-like access to rows.

119. Is it advisable to use PL/SQL table?

120. Is deadlock possible in Oracle?

Yes.

121. How will you trap the user defined exception number in a PL/SQL block?

122. What do you mean by cursor variable?

123. What are the available pseudo columns in Oracle?

A pseudocolumn behaves like a table column, but is not actually stored in the table. You can select from pseudocolumns, but you cannot insert, update, or delete their values.

Currval, Nextval, Level, Rowid & Rownum.

124. What are the role of dispatcher and other background processes of oracle database?

125. What is the use of control files?

126. How oracle maintains database recovery incase the case of instance failure?

SQL Commands

The tables in the following sections provide a functional summary of SQL commands and are divided into these categories:

- Data Definition Language commands.
- Data Manipulation Language commands.
- Transaction Control commands.
- Session Control commands.
- System Control commands.
- Embedded SQL commands.

DATA DEFINITION LANGUAGE COMMANDS

Data Definition Language (DDL) commands allow you to perform these tasks:

- CREATE, ALTER and DROP objects.
- GRANT and REVOKE privileges and roles.
- Analyze information on a TABLE, INDEX or CLUSTER.
- Establish auditing options.
- Add comments to the DATA DICTIONARY.

The CREATE, ALTER and DROP command require exclusive access to the object being acted upon. For example, an ALTER TABLE command fails if another user has an open transaction on the specified table.

The GRANT, REVOKE, ANALYZE, AUDIT, and COMMENT commands do not required exclusive access to the object being acted upon. For example, you can analyze a table while other users are updating the table.

Oracle7 implicitly commits the current transaction before and after every Data Definition Language statement.

Many Data Definition Language statements may cause Oracle7 to recompile or reauthorize schema objects. For information on how Oracle7 recompiles and reauthorize schema objects and the circumstances under which a Data Definition Language statement would cause this, see the "Dependencies Among Schema Objects" chapter of Oracle7 Server Concepts.

Data Definition Language commands are not directly supported by PL/SQL, but may be available using packaged procedures supplied by Oracle corporation. For more information, see PL/SQL User's Guide and Reference.

ALTER CLUSTER : To change the storage characteristics of a cluster and to allocate an extent for a cluster.

ALTER DATABASE : To open/mount the database. To convert an Oracle Version 6 data dictionary when migrating to Oracle7.To prepare to downgrade to an earlier release of Oracle7.To choose archive log / no archive log mode. To perform media recovery. To add/drop/clear redo log file groups members. To rename a data file/redo log file member. To backup the current control file. To backup SQL commands (that can be used to re-create the database) to the trace file. To create a new data file. To resize one or more datafiles. To create a new datafile in place of an old one for recovery purposes. To enable/disable autoextending the size of datafiles. To take a data file online/offline. To enable/disable a thread of redo log file groups. To change the database's global name. To change the MAC mode. To set the DBHIGH or DBLOW labels.

ALTER FUNCTION : To recompile a stored function.

ALTER INDEX : To redefine index’s future storage allocation.

ALTER PACKAGE : To recompile a stored package.

ALTER PROCEDURE : To recompile a stored procedure.

ALTER PROFILE : To add or remove a resource limit to or from a profile.

ALTER RESOURCE COST : To specify a formula to calculate the total cost of resources used by a session.

ALTER ROLE : To change the authorization needed to access a role.

ALTER ROLLBACK SEGMENT : To change a rollback segment's storage characteristics. To bring a rollback segment online/offline. To shrink a rollback segment to an optimal or given size.

ALTER SEQUENCE : To redefine value generation for a sequence.

ALTER SNAPSHOT : To change a snapshot's storage characteristics, automatic refresh time, or automatic refresh mode.

ALTER SHAPSHOT LOG : To change a snapshot log's storage characteristics.

ALTER TABLE : To add a column/integrity constraint to a table. To redefine a column, to change a table's storage characteristics. To enable/disable/drop an integrity constraint. To enable/disable tables locks on a table. To enable/disable all triggers on a table. To allocate an extent for the table. To allow/disallow writing to a table. To modify the degree of parallelism for a table.

ALTER TABLESPACE : To add/rename data files. To change storage characteristics. To take a tablespace online/offline. To begin/end a backup. To allow/disallow writing to a tablespace.

ALTER TRIGGER : To enable/disable a database trigger.

ALTER USER : To change a user's password, default tablespace, temporary tablespace, tablespace quotas, profile, or default roles.

ALTER VIEW : To recompile a view.

ANALYZE : To collect performance statistics, validate structure, or identify chained rows for a table, cluster, or index.

AUDIT : To choose auditing for specified SQL commands or operations on schema objects.

COMMENT : To add a comment about a table, view, snapshot or column to the data dictionary.

CREATE CLUSTER : To create a cluster that can contain one or more tables.

CREATE CONTROLFILE : To recreate a control file.

CREATE DATABASE : To create a database.

CREATE DATABASE LINK : To create a link to a remote database.

CREATE FUNCTION : To create a stored function.

CREATE INDEX : To create a index for a table or cluster.

CREATE PACKAGE : To create the specification of a stored package.

CREATE PACKAGE BODY : To create the body of a stored package

CREATE PROCEDURE : To create a stored procedure.

CREATE PROFILE : To create a profile and specify its resource limits.

CREATE ROLE : To create a role.

CREATE ROLLBACK SEGMENT : To create a rollback segment.

CREATE SCHEMA : To issue multiple CREATE TABLE, CREATE VIEW, and GRANT statements in a single transaction.

CREATE SEQUENCE : To create a sequence for generating sequential values.

CREATE SHAPSHOT : To create a snapshot of data from a remote master tables.

CREATE SNAPSHOT LOG : To create a snapshot log-containing changes made to the master table of a snapshot.

CREATE SYNONYM : To create a synonym for a schema object.

CREATE TABLE : To create a table, defining its columns, integrity constraints, and storage allocation.

CREATE TABLESPACE : To create a place in the database for storage of schema objects, rollback segments, and temporary segments, naming the data files to comprise the tablespace.

CREATE TRIGGER : To create a database trigger.

CREATE USER : To create a database user.

CREATE VIEW : To define a view of one or more tables or views.

DROP CLUSTER : To remove a cluster from the database.

DROP DATABASE LINK : To remove a database link.

DROP FUNCTION : To remove a stored function from the database.

DROP INDEX : To remove index from the database.

DROP PACKAGE : To remove a stored package from the database.

DROP PROCEDURE : To remove a stored procedure from the database.

DROP PROFILE : To remove a profile from the database.

DROP ROLE : To remove a role from the database.

DROP ROLLBACK SEGMENT : To remove a rollback segment from the database.

DROP SEQUENCE : To remove a sequence from the database.

DROP SNAPSHOT : To remove a snapshot from the database.

DROP SNAPSHOT LOG : To remove a snapshot log from the database.

DROP SYNONYM : To remove a synonym from the database.

DROP TABLE : To remove a table from the database.

DROP TABLESPACE : To remove a tablespace from the database.

DROP TRIGGER : To remove a trigger from the database.

DROP USER : To remove a user and the objects in the user's schema from the database.

DROP VIEW : To remove a view from the database.

GRANT : To grant system privileges roles and object privileges to users and roles.

NOAUDIT : To disable auditing by reversing, partially or completely, the effect of a prior AUDIT statement.

RENAME : To change the name of a schema object.

REVOKE : To revoke system privileges, roles, and object privileges from users and roles.

TRUNCATE : To remove all rows from a table or cluster and free the space that the rows used.

DATA MANIPULATION LANGUAGE COMMANDS

Data Manipulation Language (DML) commands query and manipulates data in schema objects. These commands do not implicitly commit the current transaction.

DELETE : To remove rows from a table.

EXPLAIN PLAN : To return the execution plan for a SQL statement.

INSERT : To add new rows to a table.

LOCK TABLE : To lock a table or view, limiting access to it by other users.

SELECT : To select data in rows and columns from one or more tables.

UPDATE : To change data in a table.

All Data Manipulation Language commands except the EXPLAIN PLAN command are supported in PL/SQL.

TRANSACTION CONTROL COMMANDS

Transaction Control commands manage changes made by Data Manipulation Language commands.

COMMIT : To make permanent the changes made by statements issued and the beginning of a transaction.

ROLLBACK : To undo all changes since the beginning of a transaction or since a savepoint.

SAVEPOINT : To establish a point back to which you may roll.

SET TRANSACTION : To establish properties for the current transaction.

All Transaction Control commands except certain forms of the COMMIT and ROLLBACK commands are supported in PL/SQL. For information on the restrictions, see COMMIT and ROLLBACK

SESSION CONTROL COMMANDS

Session Control commands dynamically manage the properties of a user session. These commands do not implicitly commit the current transaction. PL/SQL does not support session control commands.

ALTER SESSION : To enable/disable the SQL trace facility. To enable/disable global name resolution. To change the values of the session's NLS parameters. For Trusted Oracle7, to change the session label. To change the default label format. In a parallel server, to indicate that the session must access database files as if the session was connected to another instance. To close a database link. To send advice to remote databases for forcing an in-doubt distributed transaction. To permit or prohibit procedures and stored procedures from issuing COMMIT and ROLLBACK statements. To change the goal of the cost-based optimization approach.

SET ROLE : To enable/disable roles for the current session.

SYSTEM CONTROL COMMAND

The single System Control command dynamically manages the properties of an Oracle7 instance. This command does not implicitly commit the current transaction.

ALTER SYSTEM is not supported in PL/SQL.

ALTER SYSTEM : To alter the Oracle7 instance by performing a specialized function.

EMBEDDED SQL COMMANDS

Embedded SQL commands place Data Definition Language, Data Manipulation Language, and Transaction Control statements within a procedural language program. Embedded SQL is supported by the Oracle Precompilers.

ALLOCATE : To allocate a cursor variable.

CLOSE : To disable a cursor, releasing the resources it holds.

CONNECT : To log on to an Oracle7 instance.

DECLARE CURSOR : To declare a cursor, associating it with a query.

DECLARE DATABASE : To declare the name of a remote database.

DECLARE STATEMENT : To assign a SQL variable name to a SQL statement.

DECLARE TABLE : To declare the structure of a table for semantic checking of embedded SQL statements by the Oracle pre-compiler.

DESCRIBE : To initialize a descriptor, a structure holding host variable descriptions.

EXECUTE : To execute a prepared SQL statement or PL/SQL block or to execute an anonymous PL/SQL block.

EXECUTE IMMEDIATE : To prepare and execute a SQL statement containing no host variables.

FETCH : To retrieve rows selected by a query.

OPEN : To execute the query associated with a cursor.

PREPARE : To parse a SQL statement.

TYPE : To perform user-defined equivalencing.

VAR : To perform host variable equivalencing.

WHENEVER : To specify handling for error and warning conditions.

What Is an Oracle Precompiler?
An Oracle Precompiler is a programming tool that allows you to embed SQL statements in a high-level source program. As Figure 1-1 shows, the precompiler accepts the source program as input, translates the embedded SQL statements into standard Oracle runtime library calls, and generates a modified source program that you can compile, link, and execute in the usual way.

Editor

Host
Program ←---------- With embedded SQL statements

Oracle
Precompiler

Source
Program ←---------- With all SQL statements replaced by library calls

Compiler
Object
Program

Linker **-< -- -- - - - -- - - - - - OracleRuntimeLibrary(SQLLIB)**
To resolve calls

Executable
Program

Why Use the Oracle Pro*C/C++ Precompiler?
The Oracle Pro*C/C++ Precompiler lets you use the power and flexibility of SQL in your application programs. A convenient, easy to use interface lets your application access Oracle directly.
Unlike many application development tools, the Pro*C/C++ Precompiler lets you create highly customized applications. For example, you can create user interfaces that incorporate the latest windowing and mouse technology. You can also create applications that run in the background without the need for user interaction.
Furthermore, Pro*C/C++ helps you fine-tune your applications. It allows close monitoring of resource use, SQL statement execution, and various runtime indicators. With this information, you can tweak program parameters for maximum performance.
Although precompiling adds a step to the application development process, it saves time because the precompiler, not you, translates each embedded SQL statement into several calls to the Oracle runtime library (SQLLIB).

Why Use the Oracle Pro*C/C++ Precompiler?
The Oracle Pro*C/C++ Precompiler lets you use the power and flexibility of SQL in your application programs. A convenient, easy to use interface lets your application access Oracle directly.
Unlike many application development tools, the Pro*C/C++ Precompiler lets you create highly customized applications. For example, you can create user interfaces that incorporate the latest windowing and mouse technology. You can also create applications that run in the background without the need for user interaction.
Furthermore, Pro*C/C++ helps you fine-tune your applications. It allows close monitoring of resource use, SQL statement execution, and various runtime indicators. With this information, you can tweak program parameters for maximum performance.
Although precompiling adds a step to the application development process, it saves time because the precompiler, not you, translates each embedded SQL statement into several calls to the Oracle runtime library (SQLLIB).

For example, Pro*C/C++ allows you to
• write your application in C or C++
• follow the ANSI/ISO standards for embedding SQL statements in a

high-level language
• take advantage of dynamic SQL, an advanced programming technique that lets your program accept or build any valid SQL statement at runtime
• design and develop highly customized applications
• write multi-threaded applications
• automatically convert between Oracle internal datatypes and high-level language datatypes
• improve performance by embedding PL/SQL transaction processing blocks in your application program
• specify useful precompiler options inline and on the command line and change their values during precompilation
• use datatype equivalencing to control the way Oracle interprets input data and formats output data
• separately precompile several program modules, then link them into one executable program
• completely check the syntax and semantics of embedded SQL data manipulation statements and PL/SQL blocks
• concurrently access Oracle databases on multiple nodes using SQL*Net
• use arrays as input and output program variables
• conditionally precompile sections of code in your host program so that it can run in different environments
• directly interface with SQL*Forms via user exits written in a high-level language
• handle errors and warnings with the SQL Communications Area (SQLCA) and the WHENEVER or DO statement
• use an enhanced set of diagnostics provided by the Oracle Communications Area (ORACA)
• work with object types in the database
• use National Character Set data stored in the database
• use Oracle Call Interface functions in your program
To sum it up, the Pro*C/C++ Precompiler is a full-featured tool that supports a professional approach to embedded SQL programming.

Does the Oracle Pro*C/C++ Precompiler Meet Industry Standards?
SQL has become the standard language for relational database management systems. This section describes how the Pro*C/C++ Precompiler conforms to SQL standards established by the following organizations:
• American National Standards Institute (ANSI)
• International Standards Organization (ISO)
• U.S. National Institute of Standards and Technology (NIST)
These organizations have adopted SQL as defined in the following publications:
• ANSI standard X3.135-1992, *Database Language SQL*
• ISO/IEC standard 9075:1992, *Database Language SQL*
• ANSI standard X3.135-1989, *Database Language SQL with Integrity Enhancement*
• ANSI standard X3.168-1989, *Database Language Embedded SQL*
• ISO standard 9075-1989, *Database Language SQL with Integrity Enhancement*
• NIST standard FIPS PUB 127-1, *Database Language SQL* (FIPS is an acronym for Federal Information Processing Standards)

Requirements
ANSI standard X3.135-1992 (known informally as SQL92) provides three levels of compliance:
• Full SQL
• Intermediate SQL (a subset of Full SQL)

• Entry SQL (a subset of Intermediate SQL)
ANSI standard X3.168-1992 specifies the syntax and semantics for embedding SQL statements in application programs written in a standard programming language such as Ada, C, COBOL, FORTRAN, Pascal, or PL/I.
A conforming SQL implementation must support at least Entry SQL. The Oracle Pro*C/C++ Precompiler does conform to Entry SQL92.
NIST standard FIPS PUB 127-1, which applies to RDBMS software acquired for federal use, also adopts the ANSI standards. In addition, it specifies minimum sizing parameters for database constructs and requires a “FIPS Flagger” to identify ANSI extensions.

Compliance
Under Oracle8, the Pro*C/C++ Precompiler complies 100% with current ANSI/ISO standards.
The Pro*C/C++ Precompiler also complies 100% with the NIST standard. It provides a FIPS Flagger and an option named FIPS, which enables the FIPS Flagger. For more information, see “FIPS Flagger” on page 1-8.
Certification
NIST tested the Pro*C/C++ Precompiler for ANSI SQL92 compliance using the *SQL Test Suit*e, which consists of nearly 300 test programs. Specifically, the programs tested for conformance to the C embedded SQL standard. The result: the Oracle Pro*C/C++ Precompiler was certified 100% ANSI-compliant for Entry SQL92.

Migrating an Application from Earlier Releases
There are several semantic changes in database operations between Oracle7 and Oracle8. For information on how this affects Pro*C/C++ applications, see the section “Migrating From Earlier Pro*C/C++ Releases” on page 3-10, and the discussion of the DBMS precompiler option on page 9-13.

Frequently Asked Questions
This section presents some questions that are frequently asked about Pro*C/C++, and about Oracle8 in relation to Pro*C/C++. The answers are more informal than the documentation in the rest of this Guide, but do provide references to places where you can find the reference material.
Answer: Here’s a short description of VARCHARs:
Question: Does Pro*C/C++ generate calls to the Oracle Call Interface (OCI)?
Answer: No. Pro*C/C++ generates data structures, and calls to its runtime library: SQLLIB *(libsql.a* in UNIX). SQLLIB, in turn, calls the UPI to communicate with the database.
Question: Then why not just code using SQLLIB calls, and not use Pro*C/C++?
Answer: SQLLIB is not externally documented, is unsupported, and might change from release to release. Also, Pro*C/C++ is an ANSI/ISO compliant product, that follows the standard requirements for embedded SQL.
If you need to do low-level coding, use the OCI. It is supported, and Oracle is committed to supporting it.
You can also mix OCI and Pro*C/C++. See “SQLLIB Extensions for OCI Release 8 Interoperability” on page 4-50.
Question: Can I call a PL/SQL stored procedure from a Pro*C/C++ program?
Answer: Certainly. See Chapter 6, “Using Embedded PL/SQL”. There’s a demo program starting on page 6-21.
Question: Can I write *C++* code, and precompile it using Pro*C/C++?
Answer: Yes. Starting with Pro*C/C++ release 2.1, you can precompile C++ applications. See Chapter 7, “Using C++”.
Question: Can I use bind variables anyplace in a SQL statement? For example, VARCHAR2 A kind of column in the database that contains variable-length character data, up to 2000 bytes. This is what Oracle

calls an “internal datatype”, because it’s a possible column type. See page 3-20.
VARCHAR An Oracle “external datatype” (datatype code 9). You use this only if you’re doing dynamic SQL Method 4, or datatype equivalencing. See page 3-23 for datatype equivalencing, and Chapter 14, “Using Dynamic SQL: Advanced Concepts”.
VARCHAR[n
]
varchar[n]
This is a Pro*C/C++ “pseudotype” that you can declare as a host variable in your Pro*C/C++ program. It’s actually generated by Pro*C/C++ as a **struc**t, with a 2-byte length element, and a [n]-byte character array. See page 3-43.

Question: Does Pro*C/C++ generate calls to the Oracle Call Interface (OCI)?
Answer: No. Pro*C/C++ generates data structures, and calls to its runtime library: SQLLIB *(libsql.a* in UNIX). SQLLIB, in turn, calls the UPI to communicate with the database.
Question: Then why not just code using SQLLIB calls, and not use Pro*C/C++?
Answer: SQLLIB is not externally documented, is unsupported, and might change from release to release. Also, Pro*C/C++ is an ANSI/ISO compliant product, that follows the standard requirements for embedded SQL.
If you need to do low-level coding, use the OCI. It is supported, and Oracle is committed to supporting it.
You can also mix OCI and Pro*C/C++. See “SQLLIB Extensions for OCI Release 8 Interoperability” on page 4-50.
Question: Can I call a PL/SQL stored procedure from a Pro*C/C++ program?
Answer: Certainly. See Chapter 6, “Using Embedded PL/SQL”. There’s a demo program starting on page 6-21.
Question: Can I write *C++* code, and precompile it using Pro*C/C++?
Answer: Yes. Starting with Pro*C/C++ release 2.1, you can precompile C++ applications. See Chapter 7, “Using C++”.
Question: Can I use bind variables anyplace in a SQL statement? For example, I’d like to be able to input the name of a table in my SQL statements at runtime. But when I use host variables, I get precompiler errors.

Answer: In general, you can use host variables at any place in a SQL, or PL/SQL, statement where expressions are allowed. See page 3-32. The following SQL statement, where *table_name* is a host variable, is *illega*l:
EXEC SQL SELECT ename,sal INTO :name, :salary FROM :table_name;
To solve your problem, you need to use dynamic SQL. See Chapter 13, “Using Dynamic SQL”. There is a demo program that you can adapt to do this starting on page 13-9.
Question: I am confused by character handling in Pro*C/C++. It seems that there are many options. Can you help?
Answer: There are many options, but we can simplify. First of all, if you need compatibility with previous V1.x precompilers, and with both Oracle V6 and Oracle7, the safest thing to do is use VARCHAR[n] host variables. See page 3-43.
The default datatype for all other character variables in Pro*C/C++ is CHARZ; see page 3-27. Briefly, this means that you must null-terminate the string on input, and it is both blank-padded and null-terminated on output.
In release 8.0, the CHAR_MAP precompiler option was introduced to specify the default mapping of char variables. See “Precompiler Option CHAR_MAP” on page 3-50.
If neither VARCHAR nor CHARZ works for your application, and you need total C-like behavior (null termination, absolutely no blank-padding), use the

TYPE command and the C *typedef* statement, and use datatype equivalencing to convert your character host variables to STRING. See page 3-59. There is a sample program that shows how to use the TYPE command starting on page 3-38.

Question: What about character pointers? Is there anything special about them?

Answer: Yes. When Pro*C/C++ binds an input or output host variable, it must know the length. When you use VARCHAR[n], or declare a host variable of type *char[n*], Pro*C/C++ knows the length from your declaration. But when you use a character pointer as a host variable, and use *malloc()* to define the buffer in your program, Pro*C/C++ has no way of knowing the length.

What you must do on output is not only allocate the buffer, but pad it out with some non-null characters, then null-terminate it. On input or output, Pro*C/C++ calls *strlen()* for the buffer to get the length. See page 3-41.

Question: Where can I find the on-line versions of the sample programs?

Answer: Each Oracle installation should have a *demo* directory. On UNIX systems, for example, it is located in *$ORACLE_HOME/proc/demo*. If the directory is not there, or it does not contain the sample programs, see your system or database administrator.

Question: How can I compile and link my application?

Answer: Compiling and linking are very platform specific. Your system-specific Oracle documentation has instructions on how to link a Pro*C/C++ application. On UNIX systems, there is a makefile called *proc.mk* in the *demo* directory. To link, say, the demo program sample1.pc, you would enter the command line

```
make -f proc.mk sample1
```

If you need to use special precompiler options, you can run Pro*C/C++ separately, then do the make. Or, you can create your own custom makefile. For example, if your program contains embedded PL/SQL code, you can enter

```
proc cv_demo userid=scott/tiger sqlcheck=semantics
make -f proc.mk cv_demo
```

On VMS systems, there is a script called LNPROC that you use to link your Pro*C/C++ applications.

Question: I have been told that Pro*C/C++ now supports using structures as host variables. How does this work with the array interface?

Answer: You can use arrays inside a single structure, or an array of structures with the array interface. See page 3-35 and page 3-41.

Question: Is it possible to have recursive functions in Pro*C/C++, if I use embedded SQL in the function?

Answer: Yes. With release 2.1 of Pro*C/C++, you can also use cursor variables in recursive functions.

Question: Can I use any release of the Oracle Pro*C or Pro*C/C++ Precompiler with any version of the Oracle Server?

Answer: No. You can use an older version of Pro*C or Pro*C/C++ with a newer version of the server, but you cannot use a newer version of Pro*C/C++ with an older version of the server.

For example, you can use release 2.2 of Pro*C/C++ with Oracle8, but you cannot use Pro*C/C++ release8.0 with the Oracle7 server.

Question: When my application runs under Oracle8, I keep getting an ORA-1405 error (fetched column value is null). It worked fine under Oracle V6. What is happening?

Answer: You are selecting a null into a host variable that does not have an associated indicator variable. This is not in compliance with the ANSI/ISO standards, and was changed beginning with Oracle7.

If possible, rewrite your program using indicator variables, and use indicators in future development. Indicator variables are described on page 3-33.

Alternatively, if precompiling with MODE=ORACLE and DBMS=V7 or V8, specify UNSAFE_NULL=YES the command line (see “UNSAFE_NULL” on page 9-36 for more information) to disable the ORA-01405 message, or precompile with DBMS=V6.

Question: Are all SQLLIB functions private?
Answer: No. There are some SQLLIB functions that you can call to get information about your program, or its data. The SQLLIB public functions are shown here:
SQLSQLDAAlloc() Used to allocate a SQL descriptor array (SQLDA) for dynamic SQL Method 4.See "How is the SQLDA Referenced?" on page 14-3.
SQLCDAFromResultSetCursor() Used to convert a Pro*C/C++ cursor variable to an OCI cursor data area. See "SQLLIB Public Functions -- New Names" on page 4-34.
SQLSQLDAFree() Used to free a SQLDA allocated using *SQLSQLDAAlloc()*. See "SQLLIB Public Functions -- New Names" on page 4-34 .
SQLCDAToResultSetCursor() Used to convert an OCI cursor data area to a Pro*C/C++ cursor variable. See "SQLLIB Public Functions -- New Names" on page 4-34.
SQLErrorGetText() Returns a long error message. See "sqlerrm" on page 11-20.
SQLStmtGetText) Used to return the text of the most recently executed SQL statement. See "Obtaining the Text of SQL Statements" on page 11-30.
In the preceding list, the functions are thread-safe SQLLIB public functions. Use these functions in multi-threaded applications. The names of the functions have been changed for release 8.0, but the old names are still supported in Pro*C/C++. For more information about these thread-safe public functions (including their old names), see the table "SQLLIB Public Functions -- New Names" on page 4-34.
Question: How does Oracle8 support the new Object Types?
Answer: See the chapters Chapter 8, "Object Support in Pro*C/C++" and Chapter 16, "Using the Object Type Translator" for how to use Object types in Pro*C/C++ applications.
SQLLDAGetNamed() Used to obtain a valid Logon Data Area for a named connection, when OCI calls are used in a Pro*C/C++ program. See "SQLLIB Public Functions -- New Names" on page 4-34.
SQLLDAGetCurrent() Used to obtain a valid Logon Data Area for the most recent connection, when OCI calls are used in a Pro*C/C++ program. See "SQLLIB Public Functions -- New Names" on page 4-34.
SQLColumnNullCheck() Returns an indication of null status for dynamic SQL Method 4. See page 14-16.
SQLNumberPrecV6() Returns precision and scale of numbers. See page 14-15.
SQLNumberPrecV7() A variant of *SQLNumberPrecV6*(). See page 14-15.
SQLVarcharGetLength() Used for obtaining the padded size of a VARCHAR[n]. See page 3-45.

C H A P T E R

2 Learning the Basics

This chapter explains how embedded SQL programs do their work. You examine the special environment in which they operate and the impact of this environment on the design of your applications.

After covering the key concepts of embedded SQL programming and the steps you take in developing an application, this chapter uses a simple program to illustrate the main points.

Topics are:

• Key Concepts of Embedded SQL Programming
• Steps in Developing an Embedded SQL Application
• Sample Tables
• Sample Program: A Simple Query

Key Concepts of Embedded SQL Programming

This section lays the conceptual foundation on which later chapters build. It discusses the following subjects:

• Embedded SQL Statements
• Embedded SQL Syntax
• Static Versus Dynamic SQL Statements
• Embedded PL/SQL Blocks
• Host and Indicator Variables
• Oracle Datatypes
• Arrays
• Datatype Equivalencing
• Private SQL Areas, Cursors, and Active Sets
• Transactions
• Errors and Warnings

Embedded SQL Statements

The term *embedded SQL* refers to SQL statements placed within an application program. Because it houses the SQL statements, the application program is called a *host progra*m, and the language in which it is written is called the *host languag*e. For example, the Pro*C/C++ Precompiler allows you to embed certain SQL statements in a C or C++ host program.

To manipulate and query Oracle data, you use the INSERT, UPDATE, DELETE, and SELECT statements. INSERT adds rows of data to database tables, UPDATE modifies rows, DELETE removes unwanted rows, and SELECT retrieves rows that meet your search condition.

The powerful SET ROLE statement lets you dynamically manage database privileges. A *role* is a named group of related system and/or object privileges granted to users or other roles. Role definitions are stored in the Oracle data dictionary. Your applications can use the SET ROLE statement to enable and disable roles as needed.

Only SQL statements—not SQL*Plus statements—are valid in an application program. (SQL*Plus has additional statements for setting environment parameters, editing, and report formatting.)

Embedded SQL Syntax

In your application program, you can freely intermix complete SQL statements with complete C statements and use C variables or structures in SQL statements. The only special requirement for building SQL statements into your host program is that you begin them with the keywords EXEC SQL and end them with a semicolon. Pro*C/C++ translates all EXEC SQL statements into calls to the runtime library SQLLIB.

Many embedded SQL statements differ from their interactive counterparts only through the addition of a new clause or the use of program variables. The following example compares interactive and embedded ROLLBACK statements:

ROLLBACK WORK: -- interactive
EXEC SQL ROLLBACK WORK; -- embedded

These statements have the same effect, but you would use the first in an interactive SQL environment (such as when running SQL*Plus), and the second in a Pro*C/C++ program.

Static Versus Dynamic SQL Statements

Most application programs are designed to process static SQL statements and fixed transactions. In this case, you know the makeup of each SQL statement and transaction before runtime; that is, you know which SQL commands will be issued, which database tables might be changed, which columns will be updated, and so on.

However, some applications might be required to accept and process any valid SQL statement at runtime. So, you might not know until runtime all the SQL commands, database tables, and columns involved.

Dynamic SQL is an advanced programming technique that lets your program accept or build SQL statements at run time and take explicit control over datatype conversion.

Embedded PL/SQL Blocks

The Pro*C/C++ Precompiler treats a PL/SQL block like a single embedded SQL statement. So, you can place a PL/SQL block anywhere in an application program that you can place a SQL statement. To embed PL/SQL in your host program, you simply declare the variables to be shared with PL/SQL and bracket the PL/SQL block with the keywords EXEC SQL EXECUTE and END-EXEC. From embedded PL/SQL blocks, you can manipulate Oracle data flexibly and safely because PL/SQL supports all SQL data manipulation and transaction processing commands. For more information about PL/SQL, see Chapter 6, “Using Embedded PL/SQL”.

Oracle Datatypes

Typically, a host program inputs data to Oracle, and Oracle outputs data to the program. Oracle stores input data in database tables and stores output data in program host variables. To store a data item, Oracle must know its *datatyp*e, which specifies a storage format and valid range of values.

Oracle recognizes two kinds of datatypes: *internal* and *externa*l. Internal datatypes specify how Oracle stores data in database columns. Oracle also uses internal datatypes to represent database pseudocolumns, which return specific data items but are not actual columns in a table.

External datatypes specify how data is stored in host variables. When your host program inputs data to Oracle, if necessary, Oracle converts between the external datatype of the input host variable and the internal datatype of the target database column. When Oracle outputs data to your host program, if necessary, Oracle converts between the internal datatype of the source database column and the external datatype of the output host variable.

Arrays

Pro*C/C++ lets you define array host variables called *host arrays* and operate on them with a single SQL statement. Using the array SELECT, FETCH, DELETE, INSERT, and UPDATE statements, you can query and manipulate large volumes of data with ease. You can also use host arrays inside a host variable **struc**t.

Datatype Equivalencing

Pro*C/C++ adds flexibility to your applications by letting you *equivalence* datatypes. That means you can customize the way Oracle interprets input data and formats output data.

On a variable-by-variable basis, you can equivalence supported C datatypes to

the Oracle external datatypes. You can also equivalence user-defined datatypes to Oracle external datatypes.

Private SQL Areas, Cursors, and Active Sets

To process a SQL statement, Oracle opens a work area called a *private SQL area*. The private SQL area stores information needed to execute the SQL statement. An identifier called a *cursor* lets you name a SQL statement, access the information in its private SQL area, and, to some extent, control its processing. For static SQL statements, there are two types of cursors: *implicit* and *explicit*. Oracle implicitly declares a cursor for all data definition and data manipulation statements, including SELECT statements (queries) that return only one row. However, for queries that return more than one row, to process beyond the first row, you must explicitly declare a cursor (or use host arrays). The set of rows returned is called the *active set*; its size depends on how many rows meet the query search condition. You use an explicit cursor to identify the row currently being processed, called the *current row*.

Imagine the set of rows being returned to a terminal screen. A screen cursor can point to the first row to be processed, then the next row, and so on. In the same way, an explicit cursor “points” to the current row in the active set. This allows your program to process the rows one at a time.

Transactions

A transaction is a series of logically related SQL statements (two UPDATEs that credit one bank account and debit another, for example) that Oracle treats as a unit, so that all changes brought about by the statements are made permanent or undone at the same time.

All the data manipulation statements executed since the last data definition, COMMIT, or ROLLBACK statement was executed make up the current transaction.

To help ensure the consistency of your database, Pro*C/C++ lets you define transactions using the COMMIT, ROLLBACK, and SAVEPOINT statements. COMMIT makes permanent any changes made during the current transaction. ROLLBACK ends the current transaction and undoes any changes made since the transaction began. SAVEPOINT marks the current point in the processing of a transaction; used with ROLLBACK, it undoes part of a transaction.

Errors and Warnings

When you execute an embedded SQL statement, it either succeeds or fails, and might result in an error or warning. You need a way to handle these results. Pro*C/C++ provides two error handling mechanisms: the SQL Communications Area (SQLCA) and the WHENEVER statement.

The SQLCA is a data structure that you include (or hard code) in your host program. It defines program variables used by Oracle to pass runtime status information to the program. With the SQLCA, you can take different actions based on feedback from Oracle about work just attempted. For example, you can check to see if a DELETE statement succeeded and, if so, how many rows were deleted.

With the WHENEVER statement, you can specify actions to be taken automatically when Oracle detects an error or warning condition. These actions are: continuing with the next statement, calling a function, branching to a labeled statement, or stopping.

Steps in Developing an Embedded SQL Application

Figure 2-1 shows the embedded SQL application development process.

Figure 2-1: Embedded SQL Application Development Process

See note for fig.2.1

Sample Tables

Most programming examples in this guide use two sample database tables: DEPT and EMP. Their definitions follow:

CREATE TABLE DEPT

(DEPTNO NUMBER(2) NOT NULL,
DNAME VARCHAR2(14),
LOC VARCHAR2(13))
CREATE TABLE EMP
(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10),
JOB VARCHAR2(9),
MGR NUMBER(4),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2))

Sample Data

Respectively, the DEPT and EMP tables contain the following rows of data:

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

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO
----- ------- --------- ------ --------- ------ ------ -------
7369 SMITH CLERK 7902 17-DEC-80 800 20
7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30
7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30
7566 JONES MANAGER 7839 02-APR-81 2975 20
7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30
7698 BLAKE MANAGER 7839 01-MAY-81 2850 30
7782 CLARK MANAGER 7839 09-JUN-81 2450 10
7788 SCOTT ANALYST 7566 19-APR-87 3000 20
7839 KING PRESIDENT 17-NOV-81 5000 10
7844 TURNER SALESMAN 7698 08-SEP-81 1500 30

Sample Tables

Most programming examples in this guide use two sample database tables: DEPT and EMP. Their definitions follow:

CREATE TABLE DEPT
(DEPTNO NUMBER(2) NOT NULL,
DNAME VARCHAR2(14),
LOC VARCHAR2(13))
CREATE TABLE EMP
(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10),
JOB VARCHAR2(9),
MGR NUMBER(4),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2))

Sample Data

Respectively, the DEPT and EMP tables contain the following rows of data:

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

```
EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO
----- ------- --------- ------ --------- ------ ------ -------
7369 SMITH CLERK 7902 17-DEC-80 800 20
7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30
7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30
7566 JONES MANAGER 7839 02-APR-81 2975 20
7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30
7698 BLAKE MANAGER 7839 01-MAY-81 2850 30
7782 CLARK MANAGER 7839 09-JUN-81 2450 10
7788 SCOTT ANALYST 7566 19-APR-87 3000 20
7839 KING PRESIDENT 17-NOV-81 5000 10
7844 TURNER SALESMAN 7698 08-SEP-81 1500 30
```

Sample Program: A Simple Query

One way to get acquainted with Pro*C/C++ and embedded SQL is to study a program example. The program listed below is also available on-line in the file *sample1.pc* in your Pro*C/C++ *demo* directory.

The program connects to Oracle, then loops, prompting the user for an employee number. It queries the database for the employee's name, salary, and commission, displays the information, and then continues the loop. The information is returned to a host structure. There is also a parallel indicator structure to signal whether any of the output values SELECTed might be null. You should precompile sample programs using the precompiler option MODE=ORACLE.

```
/*
* sample1.pc
*
* Prompts the user for an employee number,
* then queries the emp table for the employee's
* name, salary and commission. Uses indicator
* variables (in an indicator struct) to determine
* if the commission is NULL.
*
*/
#include <stdio.h>
#include <string.h>
/* Define constants for VARCHAR lengths. */
#define UNAME_LEN 20
#define PWD_LEN 40
/* Declare variables.No declare section is needed if MODE=ORACLE.*/
VARCHAR username[UNAME_LEN];
/* VARCHAR is an Oracle-supplied struct */
varchar password[PWD_LEN];
/* varchar can be in lower case also. */
/*
Define a host structure for the output values of a SELECT statement.
```

Sample Program: A Simple Query

One way to get acquainted with Pro*C/C++ and embedded SQL is to study a program example. The program listed below is also available on-line in the file *sample1.pc* in your Pro*C/C++ *demo* directory.

The program connects to Oracle, then loops, prompting the user for an employee number. It queries the database for the employee's name, salary, and commission, displays the information, and then continues the loop. The information is returned to a host structure. There is also a parallel indicator structure to signal whether any of the output values SELECTed might be null.

You should precompile sample programs using the precompiler option
MODE=ORACLE.

```
/*
* sample1.pc
*
* Prompts the user for an employee number,
* then queries the emp table for the employee's
* name, salary and commission. Uses indicator
* variables (in an indicator struct) to determine
* if the commission is NULL.
*
*/
#include <stdio.h>
#include <string.h>
/* Define constants for VARCHAR lengths. */
#define UNAME_LEN 20
#define PWD_LEN 40
/* Declare variables.No declare section is needed if MODE=ORACLE.*/
VARCHAR username[UNAME_LEN];
/* VARCHAR is an Oracle-supplied struct */
varchar password[PWD_LEN];
/* varchar can be in lower case also. */
/*
Define a host structure for the output values of a SELECT statement.

/* Break out of the inner loop when a
* 1403 (”No data found”) condition occurs.
*/
EXEC SQL WHENEVER NOT FOUND DO break;
for (;;)
{
emp_number = 0;
printf(”\nEnter employee number (0 to quit): ”);
gets(temp_char);
emp_number = atoi(temp_char);
if (emp_number == 0)
break;
EXEC SQL SELECT ename, sal, comm
INTO :emprec INDICATOR :emprec_ind
FROM EMP
WHERE EMPNO = :emp_number;
/* Print data. */
printf(”\n\nEmployee\tSalary\t\tCommission\n”);
printf(”--------\t------\t\t----------\n”);
/* Null-terminate the output string data. */
emprec.emp_name.arr[emprec.emp_name.len] = ’\0’;
printf(”%-8s\t%6.2f\t\t”,
emprec.emp_name.arr, emprec.salary);
if (emprec_ind.comm_ind == -1)
printf(”NULL\n”);
else
printf(”%6.2f\n”, emprec.commission);
total_queried++;
} /* end inner for (;;) */
if (emp_number == 0) break;
printf(”\nNot a valid employee number - try again.\n”);
} /* end outer for(;;) */
printf(”\n\nTotal rows returned was %d.\n”, total_queried);
printf(”\nG’day.\n\n\n”);
/* Disconnect from ORACLE. */
```

```
EXEC SQL COMMIT WORK RELEASE;
exit(0);
}
void sql_error(msg)
char *msg;
{
char err_msg[128];
int buf_len, msg_len;
EXEC SQL WHENEVER SQLERROR CONTINUE;
printf("\n%s\n", msg);
buf_len = sizeof (err_msg);
sqlglm(err_msg, &buf_len, &msg_len);
printf("%.*s\n", msg_len, err_msg);
EXEC SQL ROLLBACK RELEASE;
exit(1);
}
```