SQL QUERY

- You have Emp & Dept table and I want to see the employees which has no Foreign key deptno in Master details relations form
  a) Hint: Use outer join (+)

Display data for those EMP whose last name ends with n

```
SELECT * FROM EMP WHERE substr (ename,-1, 1) ='N'
```

Display data which contain a special symbol like % and _

```
SELECT * FROM EMP WHERE ename LIKE '%SA\_%'ESCAPE '\'
```

CASE

```
SELECT empno, job, sal,
            CASE job WHEN 'SALESMAN' THEN 10*sal
                             WHEN 'MANAGER' THEN 20*sal
                               WHEN 'PRESIDENT' THEN 30*sal
                                ELSE sal
                                END "REVISED_SAL"
                               FROM EMP
```

```
SELECT deptno,
CASE deptno WHEN 10 THEN count(*)
WHEN 20 THEN count(*)
WHEN 30 THEN count (*)
else null
END "count of dept"
FROM emp GROUP BY deptno
```

DECODE

```
SELECT deptno,
DECODE(deptno,10,count(*),
                              20,count(*),
                              30,count(*),
                               null)
                              "count of dept"
FROM emp GROUP BY deptno
```

SELF JOIN
Find out the name of each emp manager

```
SELECT w.ename||'works for'||m.ename
FROM emp w,emp m  WHERE w.mgr=m.empno
```

ALTER TABLE

Add a new column
Modify existing column(datatype,size)
Drop column

```
ALTER TABLE emp ADD (xyz NUMBER (10))
```

```
ALTER TABLE emp MODIFY (xyz VARCHAR2(10))
```

ALTER TABLE emp DROP COLUMN xyz

ALTER TABLE emp SET UNUSED (xyz)

ALTER TABLE emp DROP UNUSED COLUMN

CONSTRAINTS

Prevents deletion of table if there is any dependencies

View cons in USER_CONSTRAINTS ,user_cons_columns
Not null constraint define only at column level
Foreiign key establish relation betn pk or uk in same table or diff table
ON DELETE CASCADE delete the dependent row in child table when parent table deleted
ON DELETE SET NULL convert dependent foregn key value to null
Add or drop cons
Enable/disable
Add not null cons using modify clause

```
CREATE TABLE emp
 (empno NUMBER (10),
   Job VARCHAR2(20) not null,
   Email VARCHAR2(20),
  Hiredate date CONSTRAINT emp_hiredate_nn NOT NULL,
 Deptno number(4) , CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno)
REFERENCES dept (deptno),
CONSTRAINT emp_email_uk UNIQUE (email),
   CONSTRAINT emp_empno_pk PRIMARY KEY (empno))
```

Display emp, sal who earn less than max sal in their dept

```
SELECT a.ename,a.sal,a.deptno,b.maxsal FROM emp a,
(select deptno,max(sal) maxsal FROM emp GROUP BY deptno)b
WHERE a.deptno=b.deptno AND
a.sal<b.maxsal
```

Top n analysis

```
SELECT ename,sal FROM (SELECT ename,sal FROM emp ORDER BY sal desc  ) WHERE rownum<=3
```

ROLLUP

Produce subtotal value

```
SELECT deptno,sum(sal) FROM emp GROUP BY ROLLUP (deptno);
```

CORELATED SUBQUERY

Subquery executed once for every row fetched by main query

Find all emp who earn more than avg sal in their dept;

```
SELECT ename,sal,deptno FROM emp outer WHERE sal >
(SELECT avg(sal) FROM emp WHERE deptno=outer.deptno)
```

Display all emp with their mgr name

```
SELECT w.ename,w.empno,m.ename,m.empno FROM emp w,emp m
```

```
 WHERE w.mgr=m.empno
```

Display all those emp who are manager

```
SELECT DISTINCT ename FROM emp e,(SELECT mgr FROM emp)m
WHERE m.mgr=e.empno

SELECT * FROM emp WHERE empno in (SELECT mgr FROM emp)
```

Display all those emp who are not manager

```
SELECT * FROM emp WHERE empno not in (SELECT mgr FROM emp WHERE mgr is not null)
```

Display top 3 earner of company

```
SELECT ename,empno,sal FROM
(SELECT ename,empno,sal FROM emp ORDER BY sal desc) WHERE
rownum<=3
```

Display top 2 earner from each dept

```
SELECT deptno,sal FROM emp e
 WHERE 2>=(SELECT count(distinct sal) FROM emp
WHEREsal>=e.sal AND deptno=e.deptno)
ORDER BY deptno,sal desc
```

Display 3rd top sal earner of company

```
SELECT DISTINCT sal FROM emp e WHERE
 (&n-1) =(SELECT COUNT (unique sal ) FROM emp b
WHERE b.sal> e.sal)
```

Display all emp who hired on the same month when smith was hired

```
SELECT * FROM emp WHERE to_char(hiredate,'mon') LIKE
(select    to_char(hiredate,'mon') FROM emp WHERE ename='SMITH')
```

Display emp who have the same manager as of scott

```
SELECT * FROM emp WHERE mgr=(SELECT mgr FROM emp
WHERE ename='SCOTT')
 AND ename<>'SCOTT'
```

To view duplicate records

```
SELECT * FROM emp x WHERE 1<(SELECT count (*) fROM emp
 WHERE empno=x.empno);
```

How can u eliminate duplicate values in a table

```
DELETE FROM emp  WHERE rownum not in  (SELECt max (rownum) FROM emp GROUP BY empno);
```

Query to delete duplicate rows but leaving one row undeleted

```
DELETE FROM emp where deptno=10 and rownum not in (SELECT min (rownum) FROM emp WHERE
deptno=x.deptno);
```

Query to find alternative rows

```
SELECT empno,ename FROM emp WHERE (empno,1) in
 (SELECT empno,mod(rownum,2) FROM emp)
```

Query to find other alternative rows

```
SELECT empno,ename FROM emp WHERE (empno,1) not in
 (SELECT empno,mod(rownum,2) FROM emp)
```

Query to delete alternative rows

```
DELETE FROM emp WHERE (empno,1) in
 (SELECT empno,mod(rownum,2) FROM emp);
```

Query to print some text with column values

```
SELECT empno,deptno,DECODE (mod(rownum,5),0,'****')print FROM emp
```

Select blank rows

```
SELECT * FROM emp WHERE empno in null
```

Display total no of emp hire by year wise

```
select count(*) total ,
sum (decode(to_char (hiredate,'yyyy'),1980,1,0))"1981",
sum (decode(to_char (hiredate,'yyyy'),1981,1,0))"1982",
sum (decode(to_char (hiredate,'yyyy'),1982,1,0))"1983",
sum (decode(to_char (hiredate,'yyyy'),1983,1,0))"1984",
sum (decode(to_char (hiredate,'yyyy'),1984,1,0))"1985",
sum (decode(to_char (hiredate,'yyyy'),1985,1,0))"1986",
sum (decode(to_char (hiredate,'yyyy'),1986,1,0))"1987" from emp;
```

o/p

```
TOTAL      1981      1982      1983      1984      1985      1986      1987
----- --------- --------- --------- --------- --------- --------- ---------
   14         1        10         2         1         0         0         0
```

```
select deptno,count(*) total ,
sum (decode(to_char (hiredate,'yyyy'),1980,1,0))"1981",
sum (decode(to_char (hiredate,'yyyy'),1981,1,0))"1982",
sum (decode(to_char (hiredate,'yyyy'),1982,1,0))"1983",
sum (decode(to_char (hiredate,'yyyy'),1983,1,0))"1984",
sum (decode(to_char (hiredate,'yyyy'),1984,1,0))"1985",
sum (decode(to_char (hiredate,'yyyy'),1985,1,0))"1986",
sum (decode(to_char (hiredate,'yyyy'),1986,1,0))"1987" from emp group by deptno;
```

o/p

```
DEPTNO     TOTAL      1981      1982      1983      1984      1985      1986      1987
------ --------- --------- --------- --------- --------- --------- --------- ---------
```

10	3	0	2	1	0	0	0	0
20	5	1	2	1	1	0	0	0
30	6	0	6	0	0	0	0	0

Create a matrix query to display the job and total salary for that job

```
Select job ,
          Sum (decode (deptno,10,sal) “dept 10”,
            Sum (decode (deptno,20,sal) “dept 20”,
            Sum (decode (deptno,30,sal) “dept 30”,
            Sum (decode (deptno,40,sal) “dept 40”
           From emp group by job;
```

Write a SQL query to display the students details whose birthday lies in the months between APR and SEP.

```
SELECT * FROM STUDENT_F03 WHERE
TO_NUMBER(TO_CHAR(DOB,'MM')) BETWEEN 4 AND 9
```

Implicit cursors:
Declared for all DML and PL/SQL SELECT statements
including queries that return only one row.

Drawbacks of Implicit Cursors
* Vulnerability to Data Errors
* Diminished Programmatic Control

Error Handling with Implicit Cursors

The implicit cursor version of the SELECT statement is kind of a black box. You pass the SQL statement to the SQL layer in the database, and it returns a single row of information. You can't get inside the separate operations of the cursor, such as the open, fetch, and close stages. You are also stuck with the fact that Oracle will automatically raise exceptions from within the implicit SELECT for two common outcomes:

* The query does not find any rows matching your criteria. In this case, Oracle raises the NO_DATA_FOUND exception.
* The SELECT statement returns more than one row. In this case, Oracle raises the TOO_ MANY_ROWS exception.

When either of these scenarios occurs (as well as any other exceptions raised when executing a SQL statement), execution of the current block terminates and control is passed to the exception section. You have no control over this process flow; you cannot tell Oracle that with this implicit cursor, you actually expect to not find any rows and it is not an error. Instead, whenever you code an implicit cursor, you should include an exception section that traps and handles these two exceptions (and perhaps others, depending on your application logic).

Explicit cursors:
Declared and named by the Programmer,
For queries that return more than one row
Use explicit cursors to individually process each row returned by a multiple-row SELECT statement.
The set of rows returned by a multiple-row query is called the active set.

With explicit cursors, you have **complete control** over how to access information in the database

Declaring the Cursor

CURSOR *cursor_name* IS
select_statement;

• Do not include the INTO clause in the cursor declaration**.**
• If processing rows in a specific sequence is required, use the ORDER BY clause in the query.

Opening the Cursor

OPEN *cursor_name*;

• Open the cursor to execute the query and identify the active set.
• If the query returns no rows, no exception is raised.
• Use cursor attributes to test the outcome after a Fetch

OPEN is an executable statement that performs the following operations:
1. Dynamically allocates memory for a context area that eventually contains crucial processing information.
2. Parses the SELECT statement.
3. Binds the input variables—sets the value for the input variables by obtaining their memory addresses.
4. Identifies the active set—the set of rows that satisfy the search criteria. Rows in the active set are not retrieved into variables when the OPEN statement is executed. Rather, the FETCH statement retrieves the rows.
5. Positions the pointer just before the first row in the active set.

Fetching Data from the Cursor

FETCH *cursor_name* INTO [*variable1, variable2, ...*]
| *record_name*]*;*

• Retrieve the current row values into variables.
• Include the same number of variables.
• Match each variable to correspond to the columns positionally.
• Test to see whether the cursor contains rows.

Closing the Cursor

CLOSE *cursor_name*;

• Close the cursor after completing the processing of the rows.
• Reopen the cursor, if required.
• Do not attempt to fetch data from a cursor after it has been closed.

Explicit Cursor Attributes

%ISOPEN= Boolean Evaluates to TRUE if the cursor is open
%NOTFOUND= Boolean Evaluates to TRUE if the most recent fetch does not return a row
%FOUND =Boolean Evaluates to TRUE if the most recent fetch returns a row;
complement of %NOTFOUND
%ROWCOUNT = Number Evaluates to the total number of

Cursor FOR Loops

FOR *record_name* IN *cursor_name* LOOP
***statement1*;**
***statement2*;**
. . .
END LOOP;

• The cursor FOR loop is a shortcut to process explicit cursors.
• Implicit open, fetch, exit, and close occur.
• The record is implicitly declared.
Cursor FOR Loops Using Subqueries No need to declare the cursor

```
BEGIN
FOR emp_record IN (SELECT last_name, department_id
FROM employees) LOOP
-- implicit open and implicit fetch occur
IF emp_record.department_id = 80 THEN
...
END LOOP; -- implicit close occurs
End;
```

The FOR UPDATE Clause

SELECT ...
FROM ...
FOR UPDATE [OF *column_reference*][NOWAIT];

• Use explicit locking to deny access for the duration of a transaction.
• Lock the rows *before* the update or delete.

You may want to lock rows before you update or delete rows. Add the FOR UPDATE clause in the cursor query to lock the affected rows when the cursor is opened. Because the Oracle Server releases locks at the end of the transaction, you should not commit across fetches from an explicit cursor if FOR UPDATE is used.

NOWAIT returns an Oracle error if the rows are locked by another session

When querying multiple tables, you can use the FOR UPDATE clause to confine row locking to particular tables. Rows in a table are locked only if the FOR UPDATE clause refers to a column in that table. FOR UPDATE OF col_name(s) locks rows only in tables that contain the col_name(s).

The WHERE CURRENT OF Clause

WHERE CURRENT OF *cursor ;*

• Use cursors to update or delete the current row.
• Include the FOR UPDATE clause in the cursor query to lock the rows first.
• Use the WHERE CURRENT OF clause to reference the current row from an explicit cursor

Instructor Note
Because of the performance impact resulting from locking a large number of rows with the FOR UPDATE clause, some developers choose not to lock the rows, and to include ROWID in the select list of the cursor. Using the ROWID in a nonlocked cursor set provides performance on par with the "WHERE CURRENT OF" clause.

```
DECLARE
CURSOR sal_cursor IS
SELECT e.department_id, employee_id, last_name, salary
FROM employees e, departments d
WHERE d.department_id = e.department_id
and d.department_id = 60
FOR UPDATE OF salary NOWAIT;
BEGIN
FOR emp_record IN sal_cursor
LOOP
IF emp_record.salary < 5000 THEN
UPDATE employees
SET salary = emp_record.salary * 1.10
WHERE CURRENT OF sal_cursor;
END IF;
END LOOP;
End;
```

When you use this clause, the cursor you reference must exist and
must contain the FOR UPDATE clause in the cursor query;

In what order should a open/fetch/loop set of commands in a PL/SQL block be implemented if you use the %NOTFOUND cursor variable in the exit when statement? Why?

OPEN then FETCH then LOOP followed by the exit when. If not specified in this order will result in the final return being done twice because of the way the %NOTFOUND is handled by PL/SQL.

Basic loop, for loop and while loop in cursor
Basic loop need exit when
For loop do not use open, fetch or close and while loop end with condition is no longer true.

three tables are joined in the cursor declaration:

```
DECLARE
   CURSOR joke_feedback_cur
   IS
      SELECT J.name, R.laugh_volume, C.name
        FROM joke J, response R, comedian C
       WHERE J.joke_id = R.joke_id
         AND J.joker_id = C.joker_id;
BEGIN
   ...
```

```
END;
```

In the above example, the cursor does not act as a pointer into any actual table in the database. Instead, the cursor is a pointer into the virtual table represented by the SELECT statement (SELECT is called a virtual table because the data it produces has the same structure as a table—rows and columns—but it exists only for the duration of the execution of the SQL statement).

PL/SQL Variables in a Cursor

Because a cursor must be associated with a SELECT statement, every cursor must reference at least one table from the database and determine from that (and from the WHERE clause) which rows will be returned in the active set. This does not mean, however, that a cursor's SELECT may only return database information. The list of expressions that appears after the SELECT keyword and before the FROM keyword is called the select list.

In native SQL, the select list may contain both columns and expressions (SQL functions on those columns, constants, etc.). In PL/SQL, the select list of a SELECT may contain PL/SQL variables, expressions, and even functions (PL/SQL Release 2.1 and above).

You can reference local PL/SQL program data (PL/SQL variables and constants), as well as host language bind variables in the WHERE, GROUP, and HAVING clauses of the cursor's SELECT statement.

In the following cursor, the SELECT statement retrieves rows based on the employee table, but the information returned in the select list contains a combination of table columns, a PL/SQL variable, and a bind variable from the host environment (such as an Oracle Forms item):

```
DECLARE
   /* A local PL/SQL variable */
   projected_bonus NUMBER := 1000;
   /*
   || Cursor adds $1000 to the salary of each employee
   || hired more than three years ago.
   */
   CURSOR employee_cur
   IS
      SELECT employee_id,
             salary + projected_bonus new_salary, /* Column alias */
            :review.evaluation                   /* Bind variable */
        FROM employee
       WHERE hiredate < ADD_MONTHS (SYSDATE, -36);

BEGIN
   ...
END;
```

Identifier Precedence in a Cursor

Be careful about naming identifiers when you mix PL/SQL variables in with database columns. It is, for instance, common practice to give a variable the same name as the column whose data it is supposed to represent. This makes perfect sense until you want to reference those local variables in a SQL statement along with the column.

In the following example, we want to fetch each employee who was hired more than three years ago and, using a local variable, add $1000 to their salary. The employee table has a column named "salary." Unfortunately, this procedure relies on a local variable of the same name to achieve its ends. Although this code will compile without error, it will not produce the desired result:

```
PROCEDURE improve_QOL
IS
   /* Local variable with same name as column: */
   salary NUMBER := 1000;
   CURSOR double_sal_cur
   IS
      SELECT salary + salary
        FROM employee
       WHERE hiredate < ADD_MONTHS (SYSDATE, -36);
BEGIN
```

Instead of adding $1000 to each person's salary, this code will instead double his or her salary. Inside the SQL statement, any unqualified reference to "salary" is resolved by using the column named "salary."

The desired effect can be achieved by qualifying the PL/SQL variable with the name of the procedure, as follows:

```
CURSOR double_sal_cur
IS
   SELECT salary + improve_QOL.salary
     FROM employee
    WHERE hiredate < ADD_MONTHS (SYSDATE, -36);
```

In this situation, you are informing the compiler that the second reference to salary is that variable "owned" by the improve_QOL procedure. It will then add the current value of that variable to the salary column value.

We do not recommend that you use qualified local variable names in this way. If your local variable names conflict with database column or table names, change the name of your variable. Best of all, avoid this kind of duplication by using a standard naming convention for local variables which represent database information.

The Cursor RETURN Clause

When you group programs together into a package, you can make only the specification, or header information, of those programs available to developers. You can accomplish the same objective with cursors by using the cursor RETURN clause. The RETURN clause allows you to create a specification for a cursor which is separate from its body (the SELECT statement). You may then place cursors in packages and hide the implementation details.

Consider the following cursor declaration with RETURN clause:

```
CURSOR caller_cur (id_in IN NUMBER) RETURN caller%ROWTYPE
IS
   SELECT * FROM caller WHERE caller_id = id_in;
```

The specification of the caller_cur cursor is:

```
CURSOR caller_cur (id_in IN NUMBER) RETURN caller%ROWTYPE
```

while the body of the caller_cur cursor is:

```
SELECT * FROM caller WHERE caller_id = id_in;
```

Everything up to but not including the IS keyword is the specification, while everything following the IS keyword is the body.

You can include a RETURN clause for any cursor you write in PL/SQL Version 2, but it is required only for cursors which are contained in a package specification.

The RETURN clause may be made up of any of the following datatype structures:

w A record defined from a database table, using the %ROWTYPE attribute
w A record defined from a programmer-defined record

Example

First, the package specification provides the name of the cursor and the RETURN datatype (an entire row from the company table):

```
PACKAGE company
IS
   CURSOR company_cur (id_in NUMBER) RETURN company%ROWTYPE;
END company;
```

Then the package body repeats the cursor specification and adds the SQL statement:

```
PACKAGE BODY company
IS
   CURSOR company_cur (id_in NUMBER) RETURN company%ROWTYPE
   IS
      SELECT * FROM company
       WHERE company_id = id_in;

END company;
```

The number of expressions in the cursor's select list must match the number of columns in the record identified by table_name%ROWTYPE or PLSQL_record%ROWTYPE. The datatypes of the elements must also be compatible. If the second element in the select list is type NUMBER, then the second column in the RETURN record cannot be type VARCHAR2 or BOOLEAN.

Placing cursors in a package is advantageous to developers because they never have to code or even see the SELECT statement. They only need to know what records the cursor returns, in what order it returns them, and which columns are in the column list.

When cursor information is limited on this kind of "need to know" basis, it protects developers and the overall application from change. Suppose that a year from now the WHERE clause of a query has to change. If a packaged cursor is not used, then each program that has a hardcoded or local cursor will have to be modified to meet the new specification. On the other hand, if all developers access the same cursor, then changes will only need to be made to that packaged declaration of the cursor. The programs can then be recompiled to automatically support this change.

CURSOR PARAMETER

* A parameter makes the cursor more reusable. Instead of hardcoding a value into the WHERE clause of a query to select particular information, you can use a parameter and then pass different values to the WHERE clause each time a cursor is opened.

* A parameter avoids scoping problems. When you pass parameters instead of hardcoding values, the result set for that cursor is not tied to a specific variable in a program or block. If your program has nested blocks, you can define the cursor at a higher-level (enclosing) block and use it in any of the subblocks with variables defined in those local blocks.

* You can specify as many cursor parameters as you want and need. When you OPEN the parameter, you need to include an argument in the parameter list for each parameter, except for trailing parameters that have default values.

TIP: Use parameters with your cursor if the cursor will be used in more than one place, with different values for the same WHERE clause.

```
DECLARE
   /*
   || Cursor with parameter list consisting of a single
   || string parameter.
   */
   CURSOR joke_cur (category_in VARCHAR2)
   IS
      SELECT name, category, last_used_date
        FROM joke
       WHERE category = UPPER (category_in);

   joke_rec joke_cur%ROWTYPE;

BEGIN
   /* Now when I open the cursor, I also pass the argument */
   OPEN joke_cur (:joke.category);
   FETCH joke_cur INTO joke_rec;
```

The most common place to use a parameter in a cursor is in the WHERE clause, but you can make reference to it anywhere in the SELECT statement, as shown here:

```
DECLARE
   CURSOR joke_cur (category_in VARCHAR2)
   IS
      SELECT name, category_in, last_used_date
        FROM joke
       WHERE category = UPPER (category_in);
```

Instead of returning the category from the table, the category_in parameter is passed back in the select list. The result will be the same either way, because the WHERE clause restricts categories to the parameter value.

Scope of Cursor Parameters

The scope of the cursor parameter is confined to that cursor. You cannot refer to the cursor parameter outside of the SELECT statement associated with the cursor. The following PL/SQL fragment will not compile because the program_name identifier is not a local variable in the block. Instead, it is a formal parameter for the cursor and is defined only inside the cursor:

```
DECLARE
   CURSOR scariness_cur (program_name VARCHAR2)
   IS
      SELECT SUM (scary_level) total_scary_level
        FROM tales_from_the_crypt
       WHERE prog_name = program_name;
BEGIN
   program_name := 'THE BREATHING MUMMY'; /* Illegal reference */
   OPEN scariness_cur (program_name);
END;
```

Cursor Parameter Modes

The syntax for cursor parameters is very similar to that of procedures and functions, with the restriction that a cursor parameter can be an IN parameter only. You cannot specify OUT or IN OUT modes for cursor parameters.

.

Default Values for Parameters

Cursor parameters can be assigned default values. Here is an example of a parameterized cursor with a default value:

```
CURSOR emp_cur (emp_id_in NUMBER := 0)
```

```
IS
   SELECT employee_id, emp_name
     FROM employee
    WHERE employee_id = emp_id_in;
```

REF CURSORS

Limitations of a normal cursors are --
1) A PL/SQL program cannot pass a cursor as a parameter to another program.
2) A PL/SQL program can only open the cursor and process the information within the program itself.

To overcome these limitations there is a concept of REF CURSOR.

Features of REF CURSOR --
1) There can be a TYPE of ref cursor. The variable of this TYPE can be used to pass the parameters to a program and return value from the cursor.
2) The variable of REF CURSOR type returns the same data type as the cursor variable.
3) The cursor variable is passed as a parameter to a procedure.
4) The cursor variable takes all the rows from the specified table.
5) These rows are given to the bind variable.
   So the parameter passed should satisfy two conditions --
    a) The parameter should be taken from the TYPE of Ref Cursor.
    b) It should be of IN OUT mode.
6) Finally the data retrieved by the cursor variable can be seen through the bind variable. For this the data type of the bind variable should be REFCURSOR.
7) While executing the procedure bind variable should be directly given. And then by print statement the data is displayed.
8) The cursor variable's data structure and the procedure block's data structure should be same.

Advantage of REF CURSOR--
Actually we can get the view of the entire data of the table with simplicity using REF CURSOR.
Without using ref cursor if we have to achieve this then, the parameter passed will be of variable type and then the user has to manual loop using cursor to fetch all the records. Here in REF CURSOR there is no need of looping.

Example of REF CURSOR –

Package Specification –

```
create or replace package PRC as

TYPE EmpRC IS REF CURSOR RETURN emp%rowtype;
TYPE DeptRC IS REF CURSOR RETURN dept%rowtype;

Procedure EmpDetails(ve IN OUT EmpRC);
Procedure DeptDetails(vd IN OUT DeptRC);

End PRC;
```

Package Body –

```
create or replace package Body PRC as

   Procedure EmpDetails(ve IN OUT EmpRC)
     is
     Begin
       Open ve FOR select * from emp;
    End EmpDetails;

    Procedure DeptDetails(vd IN OUT DeptRC)
```

```
     is
      Begin
        Open vd FOR select * from dept;

    End DeptDetails;

End PRC;
```

For executing the procdure –

```
1) SQL > variable   E   REFCURSOR
     SQL > variable   D  REFCURSOR

2) To  see the data from the bind variable --
         SQL > Set AutoPrint ON

 3) SQL > Execute PRC.EmpDetails(:E);
      4)   SQL > Execute PRC.DeptDetails(:D);
```

What are SQLCODE and SQLERRM and why are they important for PL/SQL developers?

SQLCODE returns the value of the error number for the last error encountered. The SQLERRM returns the actual error message for the last error encountered. They can be used in exception handling to report, or, store in an error log table, the error that occurred in the code. These are especially useful for the WHEN OTHERS exception.
We can not use it directly into sql statement, instead of that you must assign their value to local variable ,then use this variable in sql statement,

Where the Pre_defined_exceptions are stored ?

In the standard package.
Procedures, Functions & Packages

Introduction to Exceptions

* An error condition is called an Exception
* When an error occurs, an exception is raised i.e. normal execution stops and control transfers to the exception handling part of the PL/SQL block or subprogram
* To handle raised exceptions, separate routines called exception handlers are written

* There are two types of exceptions
– Pre-defined exceptions (Internal Exceptions)
– User-defined exceptions
* You cannot declare an exception twice in the same block, but can declare the same exception in two different blocks
* Exceptions declared in a block are local to that block and global to all its sub-blocks
* Enclosing blocks cannot reference exceptions declared in a sub-block because blocks can only reference local or global exceptions

Predefined Exceptions
* Are IMPLICITLY RAISED whenever a PL/SQL block violates an Oracle rule or exceeds a system-dependent limit
* Every Oracle error has a number, but exceptions must be handled by name
* PL/SQL predefines some common Oracle errors as exceptions
* These predefined exceptions are declared globally by PL/SQL
* Some Pre-defined Exceptions
* CURSOR_ALREADY_OPEN
* NO_DATA_FOUND
* TOO_MANY_ROWS
* VALUE_ERROR
* ZERO_DIVIDE

* More than one exception can be handled in a single exception handler by separating them with the keyword OR

```
EXCEPTION
WHEN NO_DATA_FOUND OR TOO_MANY_ROWS THEN
statements;
WHEN OTHERS THEN
statements;
END;
```

Examples of Exception handling –
1) NO_DATA_FOUND error (Variable is not having any value.)

```
declare
     n emp.ename%type;
    s emp.sal%type;
begin
 select  sal into s
 from emp
 where  ename = '&n';
dbms_output.put_line('Salary is '|| s);

 Exception
        When NO_DATA_FOUND then
          dbms_output.put_line('No record');
end;
```

2) TOO_MANY_ROWS error (Variable is having more than one value)

```
declare
  s emp.sal%type;
 begin
     select sal into s
      from emp;
      dbms_output.put_line('The salary is '|| s );
 Exception
      When TOO_MANY_ROWS then
         dbms_output.put_line('Variable can hold only one value at a time');
          dbms_output.put_line('Please specify the name of person for getting the salary');
end;
```

User-defined Exceptions

* User-defined exceptions need to be defined in the declarative part of a PL/SQL block, subprogram or database trigger
* Declared by naming the exception and defining it as datatype EXCEPTION
* Example

```
DECLARE
past_due EXCEPTION;
zero_error EXCEPTION;
```

* Like variables, user-defined exceptions must be given names
* Unlike variables, user-defined exceptions cannot be assigned values and cannot be used in SQL statements
* They need to be RAISED EXPLICITLY using the RAISE statement

* A block should RAISE an exception only when an error makes it impossible or impractical to finish processing
* RAISE statement for a given expression can be coded anywhere within the scope of that expression

```
IF mrec.ss_fare <= 0 THEN
RAISE zero_error;
END IF;
```

* An exception raised inside a handler immediately propagates to the enclosing block, which is searched to find a handler for the newly raised exception

* From there on, the exception propagates normally
* To re-raise an exception place a RAISE statement in its local handler

Example of Exception variable using Raise key word

```
declare
  p number;
  n number := 6;
  si number;
  r number := 10.5;
 EX exception;

Begin
         p := &p;
         if p < 100 then
           raise EX;
        else
          si := (p * n * r) / 100;
           dbms_output.put_line('The Simple Interest is  '|| si);
          end if;



 Exception
        /*When EX then
               dbms_output.put_line('The principle amt should be greater than or equal to 100.');*/
 end;
```

RAISE_application_error

This can be used to create user defined error meaasge,which can be more descriptive than named exceptions.

Syntax - :

```
Raise_application_error(error number,error message);
```

where error number is any parameter between -20,000 and -20,999.Error message is text that is associated with this error. The message parameter must be less than 512 characters.
Used at two places
Executable
Exception

Example of Raise_application_error

```
declare
   maths number;
    Begin
    maths  := &maths;

           if maths < 35 then
          raise_application_error(-20001,'Failed');
           else
           dbms_output.put_line('Passed');
           end if;
 end;
```

Example of Raise_application_error and error handling together –

```
declare
 x number;
 begin
 x := '&x';
 if x < 0 then
 raise_application_error(-20009,'ty');
 end if;
 exception
 when value_error then
 dbms_output.put_line('ff');
 end;
```

PRAGMA EXCEPTION_init (NON PREDEFINED ORACLE SERVER ERROR)

Declare first
Exc raised implicity

Eg.

```
DEFINE p_deptno=10

DECLARE

E_emp_remain EXCEPTION;
PRAGMA EXCEPTION_INIT(E_emp_remain,-2292);

BEGIN
-
-
-
EXCEPTION

WHEN  E_emp_remain THEN

---
----
End;
```

Question: How can I get the source code for a function or a procedure or a package from the database?
Answer: Query the view ALL_SOURCE. It has a field called TYPE, which says whether the object is a FUNCTION or a PACKAGE or a PACKAGE BODY or a PROCEDURE.
The field TEXT gives the actual source code for that object.

Example:
SELECT TEXT FROM ALL_SOURCE WHERE NAME='FUNCTION_NAME';

What built-in functions/operators are available for manipulating strings?.

The most useful ones are LENGTH, SUBSTR, INSTR, and ||:
* LENGTH(str) returns the length of str in characters.
* SUBSTR(str,m,n) returns a portion of str, beginning at character m, n characters long. If n is omitted, all characters to the end of str will be returned.
* INSTR(str1,str2,n,m) searches str1 beginning with its n-th character for the m-th occurrence of str2 and returns the position of the character in str1 that is the first character of this occurrence.
* str1 || str2 returns the concatenation of str1 and str2.
The example below shows how to convert a string name of the format 'last, first' into the format 'first last':

```
       SUBSTR(name, INSTR(name,',',1,1)+2)
    ||  ' '
    ||  SUBSTR(name, 1, INSTR(name,',',1,1)-1)
```

For case-insensitive comparisons, first convert both strings to all upper case using Oracle's built-in function upper() (or all lower case using lower()).

What is difference between SUBSTR and INSTR?

SUBSTR returns a specified portion of a string eg SUBSTR('BCDEF',4) output BCDE INSTR provides character position in which a pattern is found in a string. eg INSTR('ABC-DC-F','-',2) output 7 (2nd occurence of '-')

Difference between NO DATA FOUND and %NOTFOUND?
NO DATA FOUND is an exception raised only for the SELECT....INTO statements when the where clause of the query does not match any rows. When the where clause of the explicit cursor does not match any rows the %NOTFOUND attribute is set to TRUE instead

Which is more faster - IN or EXISTS?
EXISTS is more faster than IN because EXISTS returns a Boolean value whereas IN returns a value.

Functions

• A function is a named PL/SQL block that returns a value.
• A function can be stored in the database as a schema object for repeated execution.
• A function is called as part of an expression.
* Have a RETURN clause
* Generally take parameters
* Datatype specifier in parameter declaration must be unconstrained
Functions promote reusability and maintainability
You cannot reference host or bind variables in the PL/SQL block of a stored function.

Syntax –

CREATE [OR REPLACE] FUNCTION *function_name*
[(*parameter1* [*mode1*] *datatype1,*
parameter2* [*mode2*] *datatype2,
***. . .*)]**
RETURN *datatype*
IS|AS
PL/SQL Block;

The PL/SQL block must have at least one RETURN statement.

Advantages of User-Defined Functions in SQL Expressions
• Extend SQL where activities are too complex, too awkward, or unavailable with SQL
• Can increase efficiency when used in the WHERE clause to filter data, as opposed to filtering the data in the application
• Can manipulate character strings

Restrictions on Calling Functions from SQL Expressions
To be callable from SQL expressions, a user-defined function must:
• Be a stored function
• Accept only IN parameters
• Accept only valid SQL data types, not PL/SQL specific types, as parameters
• Return data types that are valid SQL data types, not PL/SQL specific types
• Functions called from SQL expressions cannot contain DML statements.
• Functions called from UPDATE/DELETE statements on a table T cannot contain DML on the same tableT.
• Functions called from an UPDATE or a DELETE statement on a table T cannot query the same table.
• Functions called from SQL statements cannot contain statements that end the transactions.
• Calls to subprograms that break the previous restriction are not allowed in the function

Comparing Procedures and Functions
Procedures
Execute as a PL/SQLstatement

Do not contain RETURN clause in the header
Can return none, one, or many values
Can contain a RETURN statement

Functions
Invoke as part of an expression
Must contain a RETURN clause in the header
Must return a single value
Must contain at least one RETURN statement

Calling a Function

* Can call a function as a PL/SQL statement
– Example
chardays := day_fn(3);

* Can call a function as part of an expression
– Example
IF day_fn(3) = 'TUESDAY' THEN
statements;
END IF;

b)
-- Use of a Bind Variable to get the returned value from a function
--For that on SQL prompt declare a bind variable maximum
--variable maximum number
--Now in procedures we could directly give the bind variable on the SQL prompt --since procedure can be called directly by Execute statement.
--But function cannot be called directly by Execute statement.
--So the bind variable cannot get the value of function directly.
--For that assign the value to the bind variable inside a PL/SQL block
--Whenever a bind variable is to be used inside a Pl/Sql block
--then it has to be prefixed by colon
--Advantage of a bind variable is that there is no need for the use
-- to declare a extra variable to hold the return value of the function

```
Begin
		:maximum := maxsal(20);
End;
```

/*After compiling this block to see the value from
the bind variable on SQL prompt directly type --
print maximum */

First declare two bind variables location and deptname
--SQL> variable deptname varchar2(100) (size is imp)
--SQL> variable location varchar2(100)

```
Begin
:location := getdetails(30, :deptname);
End;
```

-- To see both the values
-- print deptname location

Stored Procedures and Functions

A stored procedure or function is a PL/SQL program unit that
has a name
can take parameters, and return values
is stored in the data dictionary
can be invoked by many users

Procedures

* Subprogram that performs specific action
* Stored in database and can be invoked or called by any anonymous block
* Can take parameters
* Datatype specifier in parameter declaration must be unconstrained

CREATE [OR REPLACE] PROCEDURE *procedure_name*
[(*parameter1* [*mode1*] *datatype1,*
parameter2* [*mode2*] *datatype2,
***. . .*)]**
IS|AS
PL/SQL Block;

parameter stands for
variablename [IN|OUT|IN OUT] datatype [{:= | DEFAULT} value]

* When a procedure is created, Oracle automatically performs these steps
– Compiles the procedure
– Stores the compiled code
– Stores the procedure in the database
* The PL/SQL compiler used to compile the code
* If an error occurs, the procedure is created but it is invalid

Formal Versus Actual Parameters
• Formal parameters: variables declared in the parameter list of a subprogram specification
Example:
CREATE PROCEDURE raise_sal(p_id NUMBER, p_amount NUMBER)
...
END raise_sal;
• Actual parameters: variables or expressions
referenced in the parameter list of a subprogram call
Example:
raise_sal(v_id, 2000)

Methods for Passing Parameters
• Positional: List actual parameters in the same order as formal parameters.
• Named: List actual parameters in arbitrary order by associating each with its corresponding formal parameter.
• Combination: List some of the actual parameters as positional and some as named.

```
BEGIN
add_dept;
add_dept ('TRAINING', 2500);
add_dept ( p_loc => 2400, p_name =>'EDUCATION');
add_dept ( p_loc => 1200) ;
END;
```

Calling a Stored Procedure

* Can call a procedure in a PL/SQL statement
– Example

branch_sum(‘NYK’);
* Can call a procedure from SQL*Plus
– Example
SQL> EXECUTE branch_sum(‘NYK’);

Parameter Modes for Procedures and Functions
* Used to define the behavior of formal parameters
* Can be used with any subprogram
* Three parameter modes
– IN (Default)
– OUT
– IN OUT
* IN
– allows values to be passed to the subprogram being called
– inside the subprogram it acts like a constant
– actual corresponding parameter can be a constant, literal, initialized variable or expression
– can be initialized to default values

2) --Supplying parameters to a procedure which are by default of of IN type

```
create or replace procedure pr2(En Emp.Empno%type,Name Emp.ename%type,S Emp.Sal%type)
is
Begin
   Insert into Emp(empno,ename,sal)
   Values(En,Name,S);
     dbms_output.put_line('One record inserted through procedure');
End;
```

3) Giving default values to the parameters
Due to default value given the parameter becomes optional also.
But if any other value is given then it takes it.

```
create or replace procedure pr3 (Eno emp.empno%type,N emp.ename%type, S emp.sal%type,dno
emp.deptno%type DEFAULT 10)
is
Begin
Insert into emp (empno,ename,sal,deptno)
values(Eno,N,S,dno);
dbms_output.put_line('Record inserted');
End;

-- While executing
--exec (1,'o',800) -----> (No deptno parameter given!!!)
```

4) --Cannot give size to the parameters

```
create or replace procedure pr4 (name  char, marks number)
is
Begin
         if marks >= 35 then
       dbms_output.put_line('Passed');
     else
      dbms_output.put_line('Failed');
     end if;
    dbms_output.put_line(name);
End;
```

OUT parameter

– allows values to be returned to the caller of a subprogram
– inside the subprogram it acts like an uninitialized variable
– actual corresponding parameter must be a variable; it cannot be a constant or expression
– its value cannot be assigned to another variable or reassigned to itself

```
5) create or replace procedure pr5(Name IN varchar2, Salary OUT number)
Is
Begin
   Select sal into Salary
   from emp
   where ename = Name;
End;
--Steps for displaying the OUT parameter
--1) Compiling the procedure.
--2) Declare the bind variable on SQL prompt as variable payment number
-- Bind variables are of SQL* plus environment which are used to hold the return
--value given by the procedure or function.
--3)Now execute the proc -- exec pr5('SMITH', :payment)
--4)To display the value of payment -- print payment
--5)Bind variables are session specific.Their existence is removed as the session --ends.
```

6) IN OUT parameter
– allows initial values to be passed and returns updated values to the caller
– inside the subprogram it acts like an initialized variable
– actual corresponding parameter must be a variable; it cannot be a constant or expression
– can be assigned a value and its value can be assigned to another variable

```
a) create or replace procedure pr6(x IN OUT number)
Is
Begin

     x := (x * x);

    End;
```

/*pr6 procedure cannot be executed independently on sql prompt.
It has to be called inside a plsql block. It actually gives the square value to the variable of that plsql block.
 In short IN OUT type of paramter makes a procedure similar to function, as the function also returns the value to the calling environment.*/

```
b) declare
a number;
Begin
  a := &a;
  pr6(a);
```

/*When a is given as a parameter , it's status is of IN OUT. So IN means the user input value and OUT means the changes square figure due to the procedure pr6. After the procedure is called with a as parameter then a value gets changed. At this time a acts as a OUT parameter, since the procedure is giving the changed value to a.*/

```
    dbms_output.put_line(a);

End;
```

7) IN OUT example from with respect to database
a)

```
create or replace procedure salrise(salary IN OUT number) is
Begin
salary :=  salary + (salary * 0.20);
```

```
End;
/*Salrise procedure will increment the sal by 20% and give the value to the calling plsql block.*/

b)
 Declare
n emp.ename%type;
s emp.sal%type;
Begin
n := '&n';
select sal into s
from emp
where ename = n;
 dbms_output.put_line('The old salary is  ' || s);
/*Now calling the procdure Salrise and giving s as a IN parameter*/
   Salrise(s);
/*After giving the salary as a parameter the salary value gets incremented by 20%  */
    dbms_output.put_line('The changed salary is  '|| s);
/*updating the table*/
  Update emp
  set sal = s
where ename = n;
    dbms_output.put_line('Salary of  ' || n || '  is updated in the table.');
Exception
  When NO_DATA_FOUND then
        dbms_output.put_line('No such name');
end;

  ---------------------------------------------------------------------------------------------------
```

How can you generate debugging output from PL/SQL?
Use the DBMS_OUTPUT package. Another possible method is to just use the SHOW ERROR command, but this only shows errors. The DBMS_OUTPUT package can be used to show intermediate results from loops and the status of variables as the procedure is executed. The new package UTL_FILE can also be used.

SQL shows a packages source code

```
 select name, type, line, text
 FROM USER_SOURCE
WHERE TYPE IN('PACKAGE')
AND NAME='ENTER_PACKAGE_NAME'
UNION
select name, type, line, text
FROM USER_SOURCE
WHERE TYPE IN('PACKAGE BODY')
AND NAME='ENTER_PACKAGE_NAME';
```

What is an UTL_FILE.What are different procedures and functions associated with it?
UTL_FILE is a package that adds the ability to read and write to operating system files. Procedures associated with it are FCLOSE, FCLOSE_ALL and 5 procedures to output data to a file PUT, PUT_LINE, NEW_LINE, PUTF, FFLUSH.PUT, FFLUSH.PUT_LINE,FFLUSH.NEW_LINE. Functions associated with it are FOPEN, ISOPEN.

Packages
• Group logically related PL/SQL types, items, and subprograms
• Consist of two parts:
– Specification
– Body
• Cannot be invoked, parameterized, or nested
• Allow the Oracle server to read multiple objects

into memory at once

Package Specification
* Declares the types, variables, constants, exceptions, cursors and subprograms available for use
* Holds public declarations, visible to the application
* Scope of the declarations are local to the database schema and global to the package
* Lists the package resources available to applications
* Created using CREATE PACKAGE command

Syntax

```
CREATE [OR REPLACE] PACKAGE <packagename> AS
Global variables declaration;
Procedure specifications;
Function specifications;
Type Definitions;
Cursor Declarations
END [<packagename>];
```

Package Body

* Implements the package specification
* Fully defines cursors and subprograms
* Holds implementation details and private declarations, hidden from the application
* Can be thought of as a 'black body'
* Can be replaced, enhanced or replaced without changing the interface
* Can be changed without recompiling calling programs
* Scope of the declarations are local to the package body
* Declared types and objects are inaccessible except from within the package body
* Initialization part of a package is run only once, the first time the package is referenced

Syntax for Package Body –

```
CREATE [OR REPLACE] PACKAGE BODY <packagename> AS
Procedure Code;
Function Code;
Implementation of Types;
Use of Cursors;
Using Global variables in the members of the package.
END [<packagename>];
```

Referencing Package Objects

* Packaged objects and subprograms must be referenced using the dot notation
packagename.typename
packagename.objectname
packagename.subprogramname

E.g - DBMS_OUTPUT.PUT_LINE

Invoking Package Constructs

Example 1: Invoke a function from a procedure within the same package.

```
CREATE OR REPLACE PACKAGE BODY comm_package IS
. . .
PROCEDURE reset_comm
(p_comm IN NUMBER)
IS
```

```
BEGIN
IF validate_comm(p_comm)
THEN g_comm := p_comm;
ELSE
RAISE_APPLICATION_ERROR
(-20210, 'Invalid commission');
END IF;
END reset_comm;
END comm_package;
```

Example 2: Invoke a package procedure from *i*SQL*Plus.
Example 3: Invoke a package procedure in a different schema.
Example 4: Invoke a package procedure in a remote database.

```
EXECUTE comm_package.reset_comm(0.15)
EXECUTE scott.comm_package.reset_comm(0.15)
EXECUTE comm_package.reset_comm@ny(0.15)
```

Maintaining a Package

* Can drop a package using the DROP command
DROP PACKAGE <packagename>
* Example
DROP PACKAGE airlines;
* To drop just one construct, remove it from the package and then recompile the package

Examples of Packages –

1) Creating a package of 3 procedures –

Package Specification –

```
create or replace package pk1 is
procedure x(a number);
procedure y(b number);
procedure z(c number);
end;
```

Package Body –

```
create or replace package body pk1
is
  procedure x(a number)
    is
        Begin
              dbms_output.put_line('Procedure p1');
     End x;
      procedure y(b number)
         is
        Begin
              dbms_output.put_line('Procedure p2');
        End y;
/*Suppose in the package body if all the procedures are not written then it will give error.*/
   procedure z(c number)
     is
        Begin
              dbms_output.put_line('Procedure p3');
    End z;

End pk1;
```

 Using the Package pk1—
```
 SQL >  Execute PK1.X(4);
```

2) Use of global variable in a function and procedure –

Package Specification –

```
create or replace package pk2
as
g number;
function m(a number) return number;
procedure n;
end pk2;
```

Package Body –

```
create or replace package body pk2
as
function m(a number) return number
is
Begin
 g := a;
return g;
End m;

procedure n
is
 Begin
      if g >= 100 then
     dbms_output.put_line('Discount is 20%');
      else
       dbms_output.put_line('Discount is 10%');
     end if;
end n;
End pk2;
```

Using the package in a PL/SQL block –

```
Declare
   x number;
Begin
  x := pk2.m(700);
   pk2.n;
End;
```

3)
Use of Type in a Procedure –

Package Specification –

```
create or replace package pk3 as
Type t1 is RECORD
(e1 Emp.Empno %Type,
e2 Emp.Ename%Type,
e3 Emp.Sal%Type);
Procedure p1;
end pk3;
```

Package Body –

```
create or replace package body pk3 as
procedure p1
is
 v   t1; /*Using the type of the package directly inside the procedure.*/
Begin
 select empno,ename,sal into v
 from emp
  where ename = 'SMITH';
  dbms_output.put_line(v.e1 || '-' || v.e2 || '-' || v.e3);
End;
End pk3;
```

4) Use of Cursor in Procedure –

Package Specification –

```
create or replace package pk4
as
cursor cf is select * from emp
where job = 'MANAGER';
m cf%rowtype;
procedure CP;
End pk4;
```

Package Body –

```
create or replace package body pk4 as
procedure CP
is
   Begin
        Open cf;
                Loop
                        fetch cf into m;

                         /*Showing the first entry of manager*/
                         if cf%rowcount = 1 then
                            dbms_output.put_line(m.empno || '-' || m.ename || '-' || m.sal);
                           else
                             exit;
                         end if;
                 End Loop;
       Close cf;
    End CP;
End pk4;
```

Using Forward Declarations
You must declare identifiers before referencing them.

```
CREATE OR REPLACE PACKAGE BODY forward_pack
IS
PROCEDURE award_bonus(. . .)
IS
BEGIN
calc_rating(. . .); --illegal reference
END;
PROCEDURE calc_rating(. . .)
IS
BEGIN
```

```
...
END;
END forward_pack;
```

PL/SQL does not allow forward references. You must declare an identifier before using it. Therefore, a subprogram must be declared before calling it.
In the example in the slide, the procedure CALC_RATING cannot be referenced because it has not yet been declared. You can solve the illegal reference problem by reversing the order of the two procedures.
However, this easy solution does not always work. Suppose the procedures call each other or you absolutely want to define them in alphabetical order.
PL/SQL enables for a special subprogram declaration called a forward declaration. It consists of the subprogram specification terminated by a semicolon. You can use forward declarations to do the following:
• Define subprograms in logical or alphabetical order
• Define mutually recursive subprograms
• Group subprograms in a package
Mutually recursive programs are programs that call each other directly or indirectly.
Note: If you receive a compilation error that CALC_RATING is undefined, it is only a problem if CALC_RATING is a private packaged procedure. If CALC_RATING is declared in the package specification, the reference to the public procedure is resolved by the compiler

Using Forward Declarations

```
CREATE OR REPLACE PACKAGE BODY forward_pack
IS
PROCEDURE calc_rating(. . .); -- forward declaration
PROCEDURE award_bonus(. . .)
IS -- subprograms defined
BEGIN -- in alphabetical order
calc_rating(. . .);
. . .
END;
PROCEDURE calc_rating(. . .)
IS
BEGIN
. . .
END;
END forward_pack;
/
```

Using Forward Declarations (continued)
• The formal parameter list must appear in both the forward declaration and the subprogram body.
• The subprogram body can appear anywhere after the forward declaration, but both must appear in the same programunit.
Forward Declarations and Packages
Forward declarations typically let you group related subprograms in a package. The subprogram specifications go in the package specification, and the subprogram bodies go in the package body, where they are invisible to the applications. In this way, packages enable you to hide implementation details.

Restrictions on Package Functions Used in SQL
A function called from:
• A query or DML statement can not end the current transaction, create or roll back to a savepoint, or ALTER the system or session.
• A query statement or a parallelized DML statement can not execute a DML statement or modify the database.
• A DML statement can not read or modify the particular table being modified by that DML statement.

Using the DBMS_DDL Package
The DBMS_DDL Package:
• Provides access to some SQL DDL statements
from stored procedures
• Includes some procedures:
  - ALTER_COMPILE (object_type, owner, object_name)
  - DBMS_DDL.ALTER_COMPILE('PROCEDURE','A_USER','QUERY_EMP'
  - ANALYZE_OBJECT (object_type, owner, name,method)
  - DBMS_DDL.ANALYZE_OBJECT('TABLE','A_USER','JOBS','COMPUTE')

Note: This package runs with the privileges of calling
user, rather than the package owner SYS.

Using DBMS_JOB for Scheduling
DBMS_JOB Enables the scheduling and execution of
PL/SQL programs:
• Submitting jobs
• Executing jobs
• Changing execution parameters of jobs
• Removing jobs
• Suspending Jobs

DBMS_JOB Subprograms
Available subprograms include:
• SUBMIT
• REMOVE
• CHANGE
• WHAT
• NEXT_DATE
• INTERVAL
• BROKEN
• RUN

Using the DBMS_OUTPUT Package
The DBMS_OUTPUT package enables you to output
messages from PL/SQL blocks. Available procedures
include:
• PUT
• NEW_LINE
• PUT_LINE
• GET_LINE
• GET_LINES
• ENABLE/DISABLE

Interacting with Operating System Files
• UTL_FILE Oracle-supplied package:
– Provides text file I/O capabilities
– Is available with version 7.3 and later
• The DBMS_LOB Oracle-supplied package:
– Provides read-only operations on external BFILES
– Is available with version 8 and later
– Enables read and write operations on internal LOBs

What Is the UTL_FILE Package?
• Extends I/O to text files within PL/SQL
• Provides security for directories on the server
through the init.ora file
• Is similar to standard operating system I/O
– Open files
– Get text

– **Put text**
– **Close files**
– **Use the exceptions specific to the UTL_FILE package**

UTL_FILE Procedures and Functions

- **Function FOPEN**
- **Function IS_OPEN**
- **Procedure GET_LINE**
- **Procedure PUT, PUT_LINE, PUTF**
- **Procedure NEW_LINE**
- **Procedure FFLUSH**
- **Procedure FCLOSE, FCLOSE_ALL**

FOPEN A function that opens a file for input or output and returns a file handle used in subsequent I/O operations
IS_OPEN A function that returns a Boolean value whenever a file handle refers to an open file
GET_LINE A procedure that reads a line of text from the opened file and places the text in the output buffer parameter (the maximum size of an input record is 1,023 bytes unless you specify a larger size in the overloaded version of FOPEN)
PUT, PUT_LINE A procedure that writes a text string stored in the buffer parameter to the opened file (no line terminator is appended by put; use new_line to terminate the line, or use PUT_LINE to write a complete line with a terminator)
PUTF A formatted put procedure with two format specifiers: %s and \n (use %s to substitute a value into the output string. \n is a new line character)
NEW_LINE Procedure that terminates a line in an output file
FFLUSH Procedure that writes all data buffered in memory to a file
FCLOSE Procedure that closes an opened file
FCLOSE_ALL Procedure that closes all opened file handles for the session

Exceptions Specific to the UTL_FILE Package

- **INVALID_PATH**
- **INVALID_MODE**
- **INVALID_FILEHANDLE**
- **INVALID_OPERATION**
- **READ_ERROR**
- **WRITE_ERROR**
- **INTERNAL_ERROR**

The UTL_HTTP Package
The UTL_HTTP package:

- **Enables HTTP callouts from PL/SQL and SQL to access data on the Internet**
- **Contains the functions REQUEST and REQUEST_PIECES which take the URL of a site as a parameter, contact that site, and return the data obtained from that site**
- **Requires a proxy parameter to be specified in the above functions, if the client is behind a firewall**
- **Raises INIT_FAILED or REQUEST_FAILED exceptions if HTTP call fails**
- **Reports an HTML error message if specified URL**

is not accessible

Using the UTL_TCP Package
The UTL_TCP Package:
• Enables PL/SQL applications to communicate with external TCP/IP-based servers using TCP/IP
• Contains functions to open and close connections, to read or write binary or text data to or from a service on an open connection
• Requires remote host and port as well as local host and port as arguments to its functions
• Raises exceptions if the buffer size is too small, when no more data is available to read from a connection, when a generic network error occurs, or when bad arguments are passed to a function call

Package Description
DBMS_ALERT Provides notification of database events
DBMS_APPLICATION_INFO Allows application tools and application developers to inform the database of the high level of actions they are currently performing
DBMS_DESCRIBE Returns a description of the arguments for a stored procedure
DBMS_LOCK Requests, converts, and releases userlocks, which are managed by
the RDBMS lock management services
DBMS_SESSION Provides access to SQL session information
DBMS_SHARED_POOL Keeps objects in shared memory
DBMS_TRANSACTION Controls logical transactions and improves the performance of
short, nondistributed transactions
DBMS_UTILITY Analyzes objects in a particular schema, checks whether the server
is running in parallel mode, and returns the time
CALENDAR Provides calendar maintenance functions
DBMS_ALERT Supports asynchronous notification of database events.
Messages or alerts are sent on a COMMIT command.
Message transmittal is one way, but one sender can alert
several receivers.
DBMS_APPLICATION_INFO Is used to register an application name with the database
for auditing or performance tracking purposes
DBMS_AQ Provides message queuing as part of the Oracle server; is
used to add a message (of a predefined object type) onto a
queue or dequeue a message
DBMS_AQADM Is used to perform administrative functions on a queue or
queue table for messages of a predefined object type
DBMS_DDL Is used to embed the equivalent of the SQL commands
ALTER, COMPILE, and ANALYZE within your PL/SQL
programs
DBMS_DEBUG A PL/SQL API to the PL/SQL debugger layer, Probe, in
the Oracle server
DBSM_DEFER
DBMS_DEFER_QUERY
DBMS_DEFER_SYS
Is used to build and administer deferred remote procedure
calls (use of this feature requires the Replication Option)
DBMS_DESCRIBE Is used to describe the arguments of a stored procedure
DBMS_DISTRIBRUTED_
TRUST_ADMIN
Is used to maintain the Trusted Servers list, which is used
in conjunction with the list at the central authority to

determine whether a privileged database link from a
particular server can be accepted
DBMS_HS Is used to administer heterogeneous services by
registering or dropping distributed external procedures,
remote libraries, and non-Oracle systems (you use
dbms_hs to create or drop some initialization variables
for non-Oracle systems)
DBMS_HS_EXTPROC Enables heterogeneous services to establish security for
distributed external procedures
DBMS_HS_PASSTHROUGH Enables heterogeneous services to send pass-through SQL
statements to non-Oracle systems
DBMS_IOT Is used to schedule administrative procedures that you
want performed at periodic intervals; is also the interface
for the job queue
DBMS_JOB Is used to schedule administrative procedures that you
want performed at periodic intervals
DBMS_LOB Provides general purpose routines for operations on
Oracle large objects (LOBs) data types: BLOB, CLOB
(read only) and BFILES (read-only)
DBMS_LOCK Is used to request, convert, and release locks through
Oracle Lock Management services
DBMS_LOGMNR Provides functions to initialize and run the log reader
DBMS_LOGMNR_D Queries the dictionary tables of the current database, and
creates a text based file containing their contents
DBMS_OFFLINE_OG Provides public APIs for offline instantiation of master
groups
DBMS_OFFLINE_SNAPSH
OT
Provides public APIs for offline instantiation of snapshots
DBMS_OLAP Provides procedures for summaries, dimensions, and
query rewrites
DBMS_ORACLE_TRACE_
AGENT
Provides client callable interfaces to the Oracle TRACE
instrumentation within the Oracle7 server
DBMS_ORACLE_TRACE_
USER
Provides public access to the Oracle7 release server
Oracle TRACE instrumentation for the calling user
DBMS_OUTPUT Accumulates information in a buffer so that it can be
retrieved out later
DBMS_PCLXUTIL Provides intrapartition parallelism for creating partitionwise
local indexes
DBMS_PIPE Provides a DBMS pipe service that enables messages to
be sent between sessions
DBMS_PROFILER Provides a Probe Profiler API to profile existing PL/SQL
applications and identify performance bottlenecks
DBMS_RANDOM Provides a built-in random number generator
DBMS_RECTIFIER_DIFF Provides APIs used to detect and resolve data
inconsistencies between two replicated sites
DBMS_REFRESH Is used to create groups of snapshots that can be refreshed
together to a transactionally consistent point in time;
requires the Distributed option
DBMS_REPAIR Provides data corruption repair procedures
DBMS_REPCAT Provides routines to administer and update the replication
catalog and environment; requires the Replication option
DBMS_REPCAT_ADMIN Is used to create users with the privileges needed by the
symmetric replication facility; requires the Replication
option
DBMS_REPCAT_
INSTATIATE

Instantiates deployment templates; requires the
Replication option
DBMS_REPCAT_RGT Controls the maintenance and definition of refresh group
templates; requires the Replication option
DBMS_REPUTIL Provides routines to generate shadow tables, triggers, and
packages for table replication
DBMS_RESOURCE_
MANAGER
Maintains plans, consumer groups, and plan directives; it
also provides semantics so that you may group together
changes to the plan schema
DBMS_RESOURCE_
MANAGER_PRIVS
Maintains privileges associated with resource consumer
groups
DBMS_RLS Provides row-level security administrative interface
DBMS_ROWID Is used to get information about ROWIDs, including the
data block number, the object number, and other
components
DBMS_SESSION Enables programmatic use of the SQL ALTER SESSION
statement as well as other session-level commands
DBMS_SHARED_POOL Is used to keep objects in shared memory, so that they are
not aged out with the normal LRU mechanism
DBMS_SNAPSHOT Is used to refresh one or more snapshots that are not part
of the same refresh group and purge logs; use of this
feature requires the Distributed option
DBMS_SPACE Provides segment space information not available through
standard views
DBMS_SPACE_ADMIN Provides tablespace and segment space administration not
available through standard SQL
DSMS_SQL Is used to write stored procedure and anonymous PL/SQL
blocks using dynamic SQL; also used to parse any DML
or DDL statement
DBMS_STANDARD Provides language facilities that help your application
interact with the Oracle server
DBMS_STATS Provides a mechanism for users to view and modify
optimizer statistics gathered for database objects
DBMS_TRACE Provides routines to start and stop PL/SQL tracing
DBMS_TRANSACTION Provides procedures for a programmatic interface to
transaction management
DBMS_TTS Checks whether if the transportable set is self-contained
DBMS_UTILITY Provides functionality for managing procedures, reporting
errors, and other information
DEBUG_EXTPROC Is used to debug external procedures on platforms with
debuggers that can attach to a running process
OUTLN_PKG Provides the interface for procedures and functions
associated with management of stored outlines
PLITBLM Handles index-table operations
SDO_ADMIN Provides functions implementing spatial index creation
and maintenance for spatial objects
SDO_GEOM Provides functions implementing geometric operations on
spatial objects
SDO_MIGRATE Provides functions for migrating spatial data from release
7.3.3 and 7.3.4 to 8.1.*x*
SDO_TUNE Provides functions for selecting parameters that determine
the behavior of the spatial indexing scheme used in the
Spatial Cartridge

When to Use Autonomous Transactions

Here are some specific ideas where autonomous transactions are useful:

Logging mechanism

This is the classic example of the need for an autonomous transaction. You need to log error information in a database table, but don't want that log entry to be a part of the logical transaction.

Commits and rollbacks in your database triggers

If you define a trigger as an autonomous transaction, then you can commit and/or roll back in that code.

Retry counter

Autonomous transactions can help you keep track of how many times a user tries to connect to a database or get access to a resource (you'll reject access after a certain number of attempts).

RULES AND EXCEPTIONS -

While it is certainly very easy to add the autonomous transaction pragma to your code, there are some rules and restrictions on the use of this feature. You can only make a top-level anonymous block an autonomous transaction. The following construction will work:

```
DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
  myempno NUMBER;
BEGIN
  INSERT INTO emp VALUES (myempno, ...);
  COMMIT;
END;
/
```

whereas this construction:

```
DECLARE
  myempno NUMBER;
BEGIN
  DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
  BEGIN
    INSERT INTO emp VALUES (myempno, ...);
    COMMIT;
  END;
END;
/
```

results in this error:

```
PLS-00710: PRAGMA AUTONOMOUS_TRANSACTION cannot be declared here
```

You can now use COMMIT and ROLLBACK inside your database triggers. These actions will not affect the transaction that caused the database trigger to fire.

If an autonomous transaction attempts to access a resource held by the main transaction (which has been suspended until the autonomous routine exits), you can cause a deadlock to occur in your program. Here is a simple example to demonstrate the problem. A procedure to perform an update is created, and then called after already having all rows updated:

```
CREATE OR REPLACE PROCEDURE
  update_salary (dept_in IN NUMBER)
IS
  PRAGMA AUTONOMOUS_TRANSACTION;

  CURSOR myemps IS
    SELECT empno FROM emp
```

```
      WHERE deptno = dept_in
        FOR UPDATE NOWAIT;
BEGIN
   FOR rec IN myemps
   LOOP
      UPDATE emp SET sal = sal * 2
       WHERE empno = rec.empno;
   END LOOP;
   COMMIT;
END;
/

BEGIN
   UPDATE emp SET sal = sal * 2;
   update_salary (10);
END;
/
```

This results in the following error:

```
ERROR at line 1:
ORA-00054: resource busy and acquire with NOWAIT specified
```

You cannot mark all subprograms in a package (or all methods in an object type) as autonomous with a single PRAGMA declaration. You must indicate autonomous transactions explicitly in each program. For example, the following package specification is invalid:

```
CREATE PACKAGE warcrimes_pkg
AS
   PRAGMA AUTONOMOUS_TRANSACTION;

   PROCEDURE register (
      culprit IN VARCHAR2, event IN VARCHAR2);
END warcrimes_pkg;
/
```

One consequence of this rule is that you cannot tell by looking at the package specification which, if any, programs will run as autonomous transactions.

To exit without errors from an autonomous transaction program, you must perform an explicit commit or rollback. If the program (or any program called by it) has transactions pending, the runtime engine will raise an exception—and then it will roll back those uncommitted transactions.
Suppose, for example, that your job is to take over failing companies and make them profitable by firing employees. You would then want to use this procedure:

```
CREATE OR REPLACE PROCEDURE fire_em_all
IS
   PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   DELETE FROM emp;
END;
 /
```

You want to make the program an autonomous transaction because you don’t want anyone to back out the changes when you are not looking. If you do not explicitly commit, when you run this procedure, the following error occurs:

```
SQL> exec fire_em_all
*
ERROR at line 1
```

ORA-06519: active autonomous transaction detected and rolled back

The COMMIT and ROLLBACK statements end the active autonomous transaction; they do not, however, force the termination of the autonomous routine. You can have multiple COMMIT and/or ROLLBACK statements inside your autonomous block.

An autonomous block is one in which autonomous transactions are expected. Zero, one, or more autonomous transactions can be executed within an autonomous block.

You can roll back only to savepoints marked in the current transaction. When you are in an autonomous transaction you cannot roll back to a savepoint set in the main transaction. If you try to do so, the runtime engine will raise this exception:

ORA-01086: savepoint 'your savepoint' never established

The TRANSACTIONS parameter in the Oracle initialization file (INIT.ORA) specifies the maximum number of transactions allowed concurrently in a session. If you use autonomous transactions (which run concurrently with the main transaction), you might exceed this number—and raise an exception—unless you raise the TRANSACTIONS value. This is the error you will get if you encounter this problem:

ORA-01574: maximum number of concurrent transactions exceeded

The default value for TRANSACTIONS in Oracle8i is 75.

Using Autonomous Transactions from Within SQL

Ever since Oracle 7.3, you have been able to call your own functions from within SQL—provided that you follow a variety of rules. The main one is this: you are not allowed to update the database. And you can't save or cancel changes from within the function.

With the autonomous transaction feature, however, the picture changes. An autonomous transaction program never violates the two database-related purity levels, RNDS (reads no database state) and WNDS (writes no database state), even if that program actually does read from or write to the database. This is because those purity levels or constraints apply to the SQL statement (which, in this case, is the main transaction), yet an autonomous transaction's DML actions never affect the main transaction.

As long as you define a program to be an autonomous transaction, you can also call it directly or indirectly in a SQL statement. If your program cannot assert another purity level, such as WNPS (writes no package state), you may be restricted from calling that program in certain parts of the SQL statement, such as the WHERE clause.

As an example, suppose that you want to keep a trace of all the rows that have been touched by a query. This table is created:

```
CREATE TABLE query_trace (
  table_name VARCHAR2(30),
  rowid_info ROWID,
  queried_by VARCHAR2(30),
  queried_at DATE
  );
```

This function is then created to perform the audit:

```
CREATE OR REPLACE FUNCTION traceit (
  tab IN VARCHAR2,
  rowid_in IN ROWID)
  RETURN INTEGER
IS
BEGIN
  INSERT INTO query_trace VALUES (tab, rowid_in, USER, SYSDATE);
  RETURN 0;
END;
```

```
/
```

When you try to use this function inside a query, you get the following error:

```
SQL> select ename, traceit ('emp', rowid) from emp;
               *
ERROR at line 1:
ORA-14551: cannot perform a DML operation inside a query
```

However, if traceit is now transformed into an autonomous transaction by adding the pragma (and committing the results before the RETURN statement), the results are very different. The query works, and the query_trace table is filled:

```
SQL> SELECT ename, traceit ('emp', ROWID)
     –    FROM emp;



ENAME          TRACEIT  ('EMP',ROWID)
----------          ----------------
KING                    0
...
MILLER                  0

14 rows selected.

SQL>

SQL> SELECT table_name, rowid_info, queried_by,
  2        TO_CHAR (queried_at, 'HH:MI:SS') queried_at
  3    FROM query_trace;

TABLE_NAME ROWID_INFO        QUERIED_BY QUERIED_AT
---------- ------------------ ---------- ----------
emp        AAADEPAACAAAAg0AAA SCOTT      05:32:54
...
emp        AAADEPAACAAAAg0AAN SCOTT      05:36:50
```

You have other options when it comes to tracing queries: you can write to the screen with the DBMS_OUTPUT built-in package or send information to a pipe with DBMS_PIPE. Now that autonomous transactions are available, if you do want to send information to a database table (or delete rows or update data, etc.), you can take that route instead, but be sure to analyze carefully the overhead of this approach.

Transaction Visibility

The default behavior of autonomous transactions is that once a COMMIT or ROLLBACK occurs in the autonomous transaction, those changes are visible immediately in the main transaction. Oracle offers a SET TRANSACTION statement option to hide those changes from the main transaction:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

The default isolation level is READ COMMITTED, which means that as soon as changes are committed, they are visible to the main transaction.

As is usually the case with the SET TRANSACTION statement, you must call it before you initiate your transactions (i.e., issue any SQL statements); in addition, the setting affects your entire session, not just the current program. The following script demonstrates the SERIALIZABLE isolation level at work.

First, an autonomous transaction procedure is created:

```
CREATE OR REPLACE PROCEDURE fire_em_all
```

```
IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  DELETE FROM emp2;
  COMMIT;
END;
/
```

A script is run that sets the isolation level to SERIALIZABLE, then the number of rows that appear in the emp2 table at the following times are displayed:

- Before I call fire_em_all
- After I call fire_em_all but before the main transaction is committed or rolled back
- After I commit in the main transaction, here is the script that is run:

```
DECLARE
  PROCEDURE showcount (str VARCHAR2) IS
    num INTEGER;
  BEGIN
    SELECT COUNT(*) INTO num FROM emp2;
    DBMS_OUTPUT.PUT_LINE (str || ' ' || num);
  END;
BEGIN
  SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
  showcount ('Before isolated AT delete');
  fire_em_all;
  showcount ('After isolated AT delete');
  COMMIT;
  showcount ('After MT commit');
END;
/
```

Here is the output from running the script:

```
Before isolated AT delete 14
After isolated AT delete 14
After MT commit 0
```

****

EXAMPLE FOR DATA BASE TRIGGER DEFINED AS AN AUTONOMOUS TRANSACTIONS

The benefit of autonomous transactions for database triggers is that inside those triggers you can now issue **COMMITs** and **ROLLBACKs**, statements that are otherwise not allowed in database triggers. The changes you commit and roll back will not affect the main transaction that caused the database trigger to fire. They will only apply to DML activity taking place inside the trigger itself (or through stored program units called within the trigger).

You may want to take an action in the database trigger that is not affected by the ultimate disposition of the transaction that caused the trigger to fire. For example, suppose that you want to keep track of each action against a table, whether or not the action is completed. You might even want to be able to detect which actions failed. You can use autonomous transactions to do this.

First, construct a simple autonomous transaction trigger on the ceo_compensation table that writes a simple message to the following ceo_comp_history table. Here are the two table definitions:

```
CREATE TABLE ceo_compensation (
  company VARCHAR2(100),
  name VARCHAR2(100),
  compensation NUMBER,
  layoffs NUMBER);
CREATE TABLE ceo_comp_history (
  name VARCHAR2(100),
  description VARCHAR2(255),
  occurred_on DATE);
```

Here is the before-insert trigger to run all the elements in the script:

```
CREATE OR REPLACE TRIGGER bef_ins_ceo_comp
BEFORE INSERT ON ceo_compensation FOR EACH ROW
DECLARE
   PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   INSERT INTO ceo_comp_history VALUES (
      :new.name, 'BEFORE INSERT', SYSDATE);
   COMMIT;
END;
/
```

With this trigger in place, every insert attempt can be tracked, as shown in the steps below:

```
BEGIN
   INSERT INTO ceo_compensation VALUES (
      'Mattel', 'Jill Barad', 9100000, 2700);

   INSERT INTO ceo_compensation VALUES (
      'American Express Company',
      'Harvey Golub', 33200000, 3300);

   INSERT INTO ceo_compensation VALUES (
      'Eastman Kodak', 'George Fisher', 10700000, 20100);

   ROLLBACK; --I wish!
END;
/
SELECT name,
       description,
       TO_CHAR (occurred_on,
          'MM/DD/YYYY HH:MI:SS') occurred_on
  FROM ceo_comp_history;
NAME                 DESCRIPTION          OCCURRED_ON
-------------------- -------------------- ---------------
Jill Barad           BEFORE INSERT        03/17/1999 04:00:56
Harvey Golub         BEFORE INSERT        03/17/1999 04:00:56
George Fisher        BEFORE INSERT        03/17/1999 04:00:56
```

Fine-tuning the database trigger

But there is something of a problem with the trigger just defined. The trigger was defined as an autonomous transaction because the alert was performed in the body of the trigger. Suppose you want to perform some additional DML for the main transaction here in the trigger. It won't be rolled back with the rest of the transaction (if a rollback occurs).

Generally, it is recommended that you not make a database trigger itself the autonomous transaction. Instead, push all of the independent DML activity (such as writing to the audit or history table) into its own procedure. Make that procedure the autonomous transaction. Have the trigger call the procedure.

First, create the audit procedure:

```
CREATE OR REPLACE PROCEDURE audit_ceo_comp (
   name IN VARCHAR2,
   description IN VARCHAR2,
   occurred_on IN DATE
   )
IS
   PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   INSERT INTO ceo_comp_history VALUES (
      audit_ceo_comp.name,
      audit_ceo_comp.description,
      audit_ceo_comp.occurred_on
      );
   COMMIT;
END;
/
```

Then change the trigger to the following.

```
CREATE OR REPLACE TRIGGER aft_ins_ceo_comp
AFTER INSERT ON ceo_compensation FOR EACH ROW
DECLARE
   ok BOOLEAN := is_valid_comp_info (:NEW.name);
BEGIN
   IF ok
   THEN
      audit_ceo_comp (
         :new.name, 'AFTER INSERT', SYSDATE);
   ELSE
      RAISE VALUE_ERROR;
   END IF;
END;
/
```

Note the following differences:

- The trigger is now an after-insert trigger, rather than a before-insert trigger. Wait until after the INSERT to the compensation table takes place, then perform the audit.

- When the is_valid_comp_info function returns FALSE, do not even perform an audit. Instead, stop the transaction by raising an error. This demonstrates the other reason you don't want the trigger itself to be autonomous. In some situations, you will always want to perform an audit. Under other circumstances, however, you may want to stop the main transaction by raising an exception. Both of those events can't happen if the exception is raised in the same block and transaction as the audit DML.

As you take advantage of the new autonomous transaction pragma, plan out how you will be using these new code elements. You will almost always be better off hiding the details of your new, independent transactions behind a procedural interface.

1. **What is the difference between REF Cursor & PL/SQL Table.**

Ans: REF Cursor is like Pointer whereas PL/SQL Table is like ARRAY.
REF Cursor can pass to a procedure/function as a parameter directly whereas in PL/SQL table one record has to be passed each time.

Composite Data Types
• Are of two types:
– PL/SQL RECORDs
– PL/SQL Collections
  – INDEX BY Table
  – Nested Table

– **VARRAY**

PL/SQL Records

- **Must contain one or more components of any scalar,RECORD, or INDEX BY table data type, called fields**
- **Are similar in structure to records in a third generation language (3GL)**
- **Are not the same as rows in a database table**
- **Treat a collection of fields as a logical unit**
- **Are convenient for fetching a row of data from a table for processing**

Creating a PL/SQL Record
Syntax:

TYPE *type_name* IS RECORD
(*field_declaration[, field_declaration]…);*
identifier type_name;

Where *field_declaration* is:
field_name {field_type* | *variable%TYPE
| *table.column%TYPE* | *table%ROWTYPE}*
[[NOT NULL] {:= | DEFAULT} *expr*]

```
TYPE emp_record_type IS RECORD
(last_name VARCHAR2(25),
job_id VARCHAR2(10),
salary NUMBER(8,2));
emp_record emp_record_type;
```

.PL/SQL Tables

Features of PL/SQL tables are as follows –

1) It is a composite data type.
2) They are modeled as similar to database tables, but they are not permanent tables. So they can be created and manipulated only in a PL SQL block.
3) They can have only one column but any data type
4) It will have a primary key which is compulsory for the reference of values
5) There is no name to the column and primary key
6) The data type of the primary key is BINARY_INTEGER.
 BINARY_INTEGER is a special data type which can be given only to the column of PL/SQL table for it's indexing purpose to store and retrieve values.
7) Size is unconstrained (Table size grows as the rows are added to the table).
8) Can visualize a Pl/SQL table as a **single dimensional vertical array**, which can hold unlimited elements.
9) **Suitable for storing and displaying the values of one column of a table given by a cursor.**

Example of PL SQL Table –

Each name from the emp table is given to the vname plsql table by using cursor.   Then those names from vname table are displayed .

```
Declare
  Type nametable IS TABLE OF CHAR(10)
INDEX BY BINARY_INTEGER;
/*Creating variable vname of nametable type.*/
   vname nametable;
    Cursor cf is select ename from emp;
    i number;
    vrows number;
    /*i is for the loop and vrows is for displaying the total names from the vname table*/
Begin
```

```
    Open cf;
      i := 1;
             Loop
                  Fetch cf into vname(i);    /*Transferring each ename into vname table*/
                  Exit when cf%NotFound;

                  vrows := cf%rowcount;
                   i := i + 1;
              End Loop;
    Close cf;


/*Now retrieving the names from the vname plsql table using for loop.*/
        For n in 1..vrows
                 Loop
                        dbms_output.put_line('Name is  '||vname(n));
                  End Loop;
End;
```

VARRAY

u Also known as varying arrays
u It is an aggregation of values stored in a single column
u A single column in the database would point to a group of elements
u Can associate a single identifier with an entire collection
u Can reference the entire collection as a whole or access the elements individually
u To access individual elements, use standard subscripting syntax i.e. array_name(subscript)
u Maximum size must be specified at time of definition
u <u>Index has a fixed lower bound of 1 and an extensible upper bound</u>

Steps for Creating VARRAYs

u Create a type which will hold information for a single line item
u Create an array of type known as Varray type, holding multiple values of a particular type
u Create a table having one of the columns, known as varray column, based on the Varray type

The SYNTAX for creating a VARRAY

CREATE TYPE <varray_type_name>
AS VARRAY(limit) OF <type>;

Example – In a factory table there are workers using different tools. Each worker is using say 3 numbers of tools. Thus if a single worker is taking 3 tools then that worker's name will come thrice in the table. So there will be data redundancy. Varrays will repeat values that change for a particular column which will save the storage space.

A set of rows is repeated in one column in varray.

Step 1— Create the type

```
create type tool_ty
as object
 (toolname varchar2(25))
```

Step 2 – Now creating array of this type to hold multiple (3) values

```
create type  tool_va as
 Varray(3) of tool_ty
```

Step 3 - Finally create a table having one of the column as the varray column based on the varray type.

```
create table factory
(name varchar2(20),
 tools tool_va)
```

The SYNTAX for inserting records in a table having a column of a VARRAY

INSERT INTO <table_name> VALUES
(column_values, <varray_type_name> (<type_name>(attribute1_value,...),
<typename>(attribute2_value,...)));

```
insert into factory
  values ('John', Tool_va(
                                          Tool_ty('Hammer'),
                                          Tool_ty('Drill'),
                                          Tool_ty('Shaft')
                                          )
               );
insert into factory
  values ('Smith', Tool_va(
                                          Tool_ty('Hammer'),
                                          Tool_ty('Keys'),
                                          Tool_ty('Sledge')
                                          )
               );
insert into factory
  values ('Roger', Tool_va(
                                          Tool_ty('Vc'),
                                          Tool_ty('Bolts'),
                                          Tool_ty('Shaft')
                                           )
               );
```

Updating VARRAYs

- Individual elements cannot be updated
- Entire VARRAY has to be updated if one of the elements has to be changed

SYNTAX

UPDATE <table_name>
SET <varray_column> = <varray_type_name>
(<typename>(attribute1_value,...),
<typename>(attribute2_value,...))
WHERE <condition>;

To change the name of the tool of John from Drill to Drill-2.5mm

```
update factory
set tools = Tool_va
(tool_ty('Hammer'), tool_ty('Drill-2.5mm'),  tool_ty('Shaft'))
where name = 'John';
```

Displaying data from the varray column –

```
declare
             cursor cf is
                        select * from factory;
                        vcf cf%rowtype;
begin
            for vcf in cf
                  loop
                  /*This loop is for the normal column name.*/
                 dbms_output.put_line('Contact Name '|| vcf.name);
                          for i in 1..vcf.tools.count
                                loop
```

```
                                    /*This loop is for the number of tools for the current                                    row's
name*/
                                    dbms_output.put_line(vcf.tools(i).toolname);
                              end loop;
                dbms_output.put_line('----------------------------------');
                end loop;
end;
-------------------------------------------------------------------------------------------------
```

Displaying status for each record whether Hammer is given or not—

```
declare
            cursor cf is
                      select * from factory;
                      vcf cf%rowtype;
            x  number;
begin
           for vcf in cf
                 loop
                       x := 0;
                        dbms_output.put_line('Contact Name --  '|| vcf.name);
                         for i in 1..vcf.tools.count
                              loop
                                       if vcf.tools(i).toolname = 'Hammer' then
                                        x := x + 1;
                                       end if;
                              end loop;
                             if x > 0 then
                              dbms_output.put_line('Hammer is supplied');
                             else
                             dbms_output.put_line('Hammer is not supplied');
                           end if;
                           dbms_output.put_line('--------------------------- ');
                end loop;
end;
```

Displaying only count of workers to whom Hammer is supplied.

```
declare
            cursor cf is
                      select * from factory;
                      vcf cf%rowtype;
            countofHammers number;
begin
           /*Count of Hammers has to be initialised out side the outer for loop.*/
          countofHammers := 0;
           for vcf in cf
                 loop
                      for i in 1..vcf.tools.count
                              loop
                                      if vcf.tools(i).toolname='Hammer' then
                                        countofHammers :=  countofHammers + 1;
                                      end if;
                              end loop;
                    end loop;

      dbms_output.put_line('The total number of workers who have got hammer is  '|| countofHammers);
        end;
```

Displaying the names of workers who have been given shafts.

```
declare
            cursor cf is
                      select * from factory;
                      vcf cf%rowtype;
                      countOfShaftHolders number;
begin
```

```
            dbms_output.put_line('The Shaft holders are --> ');
              countOfShaftHolders := 0;
           for vcf in cf
                 loop
                        for i in 1..vcf.tools.count
                                loop
                                      if vcf.tools(i).toolname='Shaft' then
                                        dbms_output.put_line(vcf.name);
                                          countOfShaftHolders :=   countOfShaftHolders + 1;
                                      end if;
                               end loop;
                    end loop;
                        if   countOfShaftHolders = 0 then
                               dbms_output.put_line('Sorry, there are no shaft holders');
                        end if;

end;
```

Nested Table

--- Table within a table
--- A table is represented as a column within another table
--- There is no limit to the number of rows in the nested table for each row in the main table.
--- Basically used for mapping master-detail relationships between tables.
i.e. In the parent table there would be one column, which will store the location of the nested table.

Steps for Creating Nested Tables

- u Create a type which will hold information for a single line item (this represents the nested table details)
- u Create Type of Table to hold details for the multiple rows of the nested table (this will be the nested details - nested table type)
- u Create the master table with one of the columns which will hold the nested table type

Creating a Nested Table

SYNTAX

CREATE TYPE <nested_table_type_name>
AS TABLE OF <typename>;

Example --
There is main table Fare_Tab1.
It has columns such as Route_Coe,Route_Desc,Origin,Destination, Firts_Fare, Bus_Fare,Eco_Fare and journey_hrs.
There is a type as table Fs_Nst_Type.
This type table holds column such as Flightno, Airbusno,Deprt_Time, Flight_Day1 and Flight_Day2.

Now we are trying to establish the relation between Fare_Tab1 table and this type table Fs_Nst_Type.

One route_code from Fare_Tab1 table can hold multiple Flightno, Airbusno,Deprt_Time, Flight_Day1 and Flight_Day2. For this purpose we will create a column in Fare_Tab1 table which will show details of the Fs_Nst_Type for a particular route_code. So that column actually represents the nested table.

Step 1 -- Create a type Flight_Sch_type to hold details for a single flight.

Create or replace type Flight_Sch_Type as OBJECT

```
(Flightno char(4),
 Airbusno char(4),
 Deprt_time char(6),
Flight_day1 number(1),
Flight_day2 number(1));
```

Step 2 -- Create a type of table which will hold the multiple rows of the nested table.

```
Create type Fs_Nst_Type
 as TABLE of    Flight_Sch_Type;
```

Step3 -- Create the main table which will have the nested table through a column of the main table

```
  Create table Fare_Tab1
(
  Route_Code char(7),
  Route_Desc varchar2(30),
 Origin varchar2(15),
 Destination Varchar2(15),
 First_fare number(5),
 Bus_Fare number(5),
 Eco_Fare number(5),
 Journey_Hrs char(5),
 Flight_Sch_Det  Fs_Nst_Type
)
  Nested Table Flight_Sch_Det store as Fs_Store_Tab;
```

Fs_Store_Tab is system generated table for oracle's reference.

Inserting records in the nested table through the main table

- **SYNTAX**

```
INSERT INTO <master_table_name>
VALUES(<master_table_column1_value>,…
                                    <master_table_columnN_value,
nested_table_type(nested_column(attribute1_value>,...<attributeN_value>),
nested_column(attribute1_value>,…
                        <attributeN_value>)));
```

```
Insert into Fare_Tab1
Values ('Goa-Ban', 'Goa - Bangalore', 'Goa', 'Bangalore', 450, 300, 200, 3,
 Fs_Nst_Type  ( Flight_Sch_Type('F2', 'AB01',' 9:00',1,3),
                          Flight_Sch_Type('F3','AB02', '11:00', 3, 5) ) );
```

Here in this insert statement for one route_code 'Goa-Ban' 2 nested rows of F2 and F3 are inserted.

Displaying the data of the nested table columns --

- **SYNTAX in Oracle 8i**

```
SELECT <master_table_columns>,
<nested_alias>.<attributes_from_type>
FROM THE (SELECT <nested_column>
FROM <master_table>
WHERE <master_table_condition>) <nested_alias>, <master_table>
WHERE <nested_table_condition>;
```

To display the nested row for the route code 'Goa-Ban' for flighno F1

```
Select NT.flight_day1, NT.flight_day2 ,Nt.Deprt_Time, route_code
  from THE (Select Flight_Sch_Det from
                    Fare_Tab1
                    where route_code = 'Goa-Ban')
                NT, Fare_Tab1
  where  NT.Flightno = 'F2';
```

In Oracle 9i Table function is used instead of THE clause which makes the query very easy to understand

```
select route_code, NT.flight_day1, Nt.flight_day2,NT.Deprt_Time,NT.flightno
 from Fare_Tab1, Table(Fare_Tab1.Flight_Sch_Det) NT
 where route_code = 'Goa-Ban' And  NT.flightno = 'F3'
```

So Syntax in 9i is
Select   master_table_cols, Alias.Neste_Table_Cols
From Master_Table, Table(Master_Table.Nested_Table) Alias

<u>To see all attribute columns of the Nested Table we can use * with alias name of the nested table –</u>

```
select route_code, NT.*
from Fare_Tab1, Table(Fare_Tab1.Flight_Sch_Det) NT
where route_code = 'Goa-Ban' And  NT.flightno = 'F3'
```

Updating the nested row in 8i

- **SYNTAX**

```
UPDATE THE (SELECT <nested_column>
FROM <master_table>
WHERE <master_table_condition>) <nested_alias>
SET <nested_alias>.<attribute1_name> = <value>,...,
WHERE <nested_table_condition>;
```

To change the value of flight_day1 from 3 to 5 for the Flightno F3 on the route_code 'Goa-Ban'

```
Update THE
      (Select Flight_Sch_Det
        from Fare_Tab1 F
         where F.route_code = 'Goa-Ban')
         NT
 Set
         NT.Flight_day1 = 5
         where NT.Flightno = 'F3';
```

F is the alias name for the master table Fare_Tab
NT is the alias name for the nested table.

In 9i the same Update statement will be as follows –

To change the Flightno of the nested table to F11

```
update TABLE(select FLIGHT_SCH_DET
             from Fare_Tab1
             where route_code = 'Goa-Ban') NT
Set NT.Flightno = 'F11'
where NT.Flight_Day2 = 5;
```

Syntax in 9i –

```
 Update TABLE(select nested_table_name)
                         From Master_Table
                          Where master_table condition) Alias for Nested_Table
 Set Alis.Nested_Col = <new value>
Where Alias.Nested_Col Condition;
```

Deleting a row from the nested table

- **SYNTAX in 9i**

```
DELETE FROM THE
(SELECT <nested_column>
FROM <master_table>
WHERE <master_table_condition>) <nested_alias>
WHERE <nested_table_condition>;
```

To delete the details of flightno 'F2' for the route_code 'Goa-Ban'

```
Delete From THE
    (Select Flight_Sch_Det
       from Fare_Tab1 F
       where F.Route_Code  = 'Goa-Ban')
    NT
      where NT.Flightno = 'F2';
```

Deleting a row in 9i –
To delete a nested row of Flight_day2 having value 5 for a route_code 'Goa-Ban'

```
Delete Table(select  FLIGHT_SCH_DET
             from Fare_Tab1
             where route_code = 'Goa-Ban') NT
 where NT.Flight_Day2 = 5;
```

Inserting a nested row to route_code 'Goa-Ban' – (9i)

```
Insert into TABLE (Select FLIGHT_SCH_DET
                   from Fare_Tab1
                   where route_code = 'Goa-Ban')
Values
(Flight_Sch_Type ('F22','AB02','3',3,5))
```

Difference Between VARRAYs and Nested Tables

VARRAYs	**Nested Tables**
u Maximum size specified at time of definition	u No maximum size
u Data is stored in-line	u Data is stored out-of-line in a store table
• Retain their ordering and subscripts when stored in database	u Do not retain ordering and subscripts when stored in database

When to use Nested Tables instead of VARRAYs

- When the order of the data elements is not important
- When indexing is required on an attribute of the nested table type
- When there is no set limit to the number of entries for the data elements
- When the data elements need to be queried

Collections

- u Collections are similar to arrays available in most 3GLs
- u A collection is an object that contains other objects of similar type
- u It is an ordered group of elements of the same type
- u Each element is identified by a unique subscript that determines its position in the collection

A *collection* is an ordered group of elements, all of the same type (for example, the grades for a class of students). Each element has a unique subscript that determines its position in the collection. PL/SQL offers two collection types. Items of type TABLE are either *index-by tables* (Version 2 *PL/SQL tables*) or *nested tables* (which extend the functionality of index-by tables). Items of type VARRAY are *varrays* (short for variable-size arrays). Collections work like the arrays found in most third-generation programming languages. However, collections can have only one dimension and must be indexed by integers. (In some languages such as Ada and Pascal, arrays can have multiple dimensions and can be indexed by enumeration types.) You can define collection types in a package, then use them programmatically in your applications. Also, you can pass collections as parameters. So, you can use them to move columns of data into and out of database tables or between client-side applications and stored subprograms. In addition, collections can store instances of
an object type and (except for index-by tables) can be attributes of an object type

Step 1 – Creating the collection type.

```
CREATE TYPE Color_tab_t AS TABLE OF VARCHAR2(30);
```

Step 2 -
Type of a TABLE used as a data type of a column. This column is having nested table.

```
CREATE TABLE cars(
  name VARCHAR2(12),
  colors Color_tab_t)
NESTED TABLE colors STORE AS color_model_colors_tab;
```

Colors is the column having Color_tab_t type. The colors column is having the nested table.

Step3 – Inserting records….

```
insert into cars
  values ('Maruti', Color_tab_t('RED','GREEN','BLUE'));

insert into cars
  values ('Indica',Color_tab_t('CYAN','YELLOW','MAGENTA','BLACK'));
```

Step 4 --
To retrieve color names through a PL/SQL block create a variable of Color_tab_t type. Then fetch the nested table column into that variable for that row. Then through loop show each color with the help of the variable.

To see all the colors available to Maruti car --

```
DECLARE
  colorlist Color_tab_t;
BEGIN
  SELECT colors INTO colorlist  FROM cars
    WHERE name = 'Maruti';
    FOR i IN 1..colorlist.COUNT
    LOOP
```

```
        dbms_output.put_line( colorlist(i) );
      END LOOP;
END;
```

INDEX BY Tables
• Are composed of two components:
– Primary key of data type BINARY_INTEGER
– Column of scalar or record data type
• Can increase in size dynamically because they are Unconstrained

Using INDEX BY Table Methods
The following methods make INDEX BY tables easier to use:
– NEXT
– TRIM
– DELETE
– EXISTS
– COUNT
– FIRST and LAST
– PRIOR

Basic Trigger Syntax

```
CREATE [OR REPLACE] TRIGGER <trigger_name>

    {BEFORE|AFTER} {INSERT|DELETE|UPDATE} ON <table_name>

    [REFERENCING [NEW AS <new_row_name>] [OLD AS <old_row_name>]]

    [FOR EACH ROW [WHEN (<trigger_condition>)]]

    <trigger_body>
```

Some important points to note:
• **BEFORE**: Execute the trigger body before the triggering DML event on a table.
• **AFTER**: Execute the trigger body after the triggering DML event on a table.
• **INSTEAD OF**: Execute the trigger body instead of the triggering statement. This is used for views that are not otherwise modifiable

* You may specify up to three triggering events using the keyword OR.,

* UPDATE can be optionally followed by the keyword OF and a list of attribute(s) in <table_name>. If present, the OF clause defines the event to be only an update of the attribute(s) listed after OF. Here are some examples:

```
    ... INSERT ON R ...

    ... INSERT OR DELETE OR UPDATE ON R ...

    ... UPDATE OF A, B OR INSERT ON R ...
```

* If FOR EACH ROW option is specified, the trigger is row-level; otherwise, the trigger is statement-level.

* Only for row-level triggers:
The special variables NEW and OLD are available to refer to new and old tuples respectively. Note: In the trigger body, NEW and OLD must be preceded by a colon (":"), but in the WHEN clause, they do not have a preceding colon!
* The REFERENCING clause can be used to assign aliases to the variables NEW and OLD.
* A trigger restriction can be specified in the WHEN clause, enclosed by parentheses. The trigger restriction is a SQL condition that must be satisfied in order for Oracle to fire the trigger. This condition cannot contain subqueries. Without the WHEN clause, the trigger is fired for each row.
* <trigger_body> is a PL/SQL block, rather than sequence of SQL statements. Oracle has placed certain restrictions on what you can do in <trigger_body>, in order to avoid situations where one trigger performs an action that triggers a second trigger, which then triggers a third, and so on, which could potentially create an infinite loop. The restrictions on <trigger_body> include:
* You cannot modify the same relation whose modification is the event triggering the trigger.
* You cannot modify a relation connected to the triggering relation by another constraint such as a foreign-key constraint.

Trigger Example

We illustrate Oracle's syntax for creating a trigger through an example based on the following two tables:

```
CREATE TABLE T4 (a INTEGER, b CHAR(10));

CREATE TABLE T5 (c CHAR(10), d INTEGER);
```

We create a trigger that may insert a tuple into T5 when a tuple is inserted into T4. Specifically, the trigger checks whether the new tuple has a first component 10 or less, and if so inserts the reverse tuple into T5:

```
CREATE TRIGGER trig1
   AFTER INSERT ON T4
   REFERENCING NEW AS newRow
   FOR EACH ROW
   WHEN (newRow.a <= 10)
   BEGIN
      INSERT INTO T5 VALUES(:newRow.b, :newRow.a);
   END trig1;
.
run;
```

Notice that we end the CREATE TRIGGER statement with a dot and run, as for all PL/SQL statements in general. Running the CREATE TRIGGER statement only creates the trigger; it does not execute the trigger. Only a triggering event, such as an insertion into T4 in this example, causes the trigger to execute.

Displaying Trigger Definition Errors

```
show errors trigger <trigger_name>;
.
```

Viewing Defined Triggers

To view a list of all defined triggers, use:

```
select trigger_type, triggering_event, table_name, referencing_names, trigger_body
from user_triggers
where trigger_name = '<trigger_name>';
```

Dropping Triggers

To drop a trigger:

```
drop trigger <trigger_name>;
```

Disabling Triggers

alter trigger <trigger_name> {disable|enable};

Rename Trigger

ALTER TRIGGER <trigger_name> RENAME TO <new_name>;

CALL Statements

CREATE [OR REPLACE] TRIGGER *trigger_name*
timing
***event1* [OR *event2* OR *event3*]**
ON *table_name*
[REFERENCING OLD AS *old* | NEW AS *new*]
[FOR EACH ROW]
[WHEN *condition*]
CALL *procedure_name*

CREATE OR REPLACE TRIGGER log_employee
BEFORE INSERT ON EMPLOYEES
CALL log_execution

A CALL statement enables you to call a stored procedure, rather than coding the PL/SQL body in the trigger itself. The procedure can be implemented in PL/SQL, C, or Java.
The call can reference the trigger attributes :NEW and :OLD as parameters as in the following example:

```
CREATE TRIGGER salary_check
BEFORE UPDATE OF salary, job_id ON employees
FOR EACH ROW
WHEN (NEW.job_id <> 'AD_PRES')
CALL check_sal(:NEW.job_id, :NEW.salary)
```

Aborting Triggers with Error
Triggers can often be used to enforce contraints. The WHEN clause or body of the trigger can check for the violation of certain conditions and signal an error accordingly using the Oracle built-in function RAISE_APPLICATION_ERROR. The action that activated the trigger (insert, update, or delete) would be aborted. For example, the following trigger enforces the constraint Person.age >= 0:

```
create table Person (age int);
CREATE TRIGGER PersonCheckAge
AFTER INSERT OR UPDATE OF age ON Person
FOR EACH ROW
BEGIN
IF (:new.age < 0) THEN
RAISE_APPLICATION_ERROR(-20000, 'no negative age allowed');
END IF;
END;
.
RUN;
```

If we attempted to execute the insertion:

```
insert into Person values (-3);
```

we would get the error message:

```
ERROR at line 1:
ORA-20000: no negative age allowed
ORA-06512: at "MYNAME.PERSONCHECKAGE", line 3
ORA-04088: error during execution of trigger 'MYNAME.PERSONCHECKAGE'
```

and nothing would be inserted. In general, the effects of both the trigger and the triggering statement are rolled back.

Mutating Table Errors

Sometimes you may find that Oracle reports a "mutating table error" when your trigger executes. This happens when the trigger is querying or modifying a "mutating table", which is either the table whose modification activated the trigger, or a table that might need to be updated because of a foreign key constraint with a CASCADE policy. To avoid mutating table errors:

* A row-level trigger must not query or modify a mutating table. (Of course, NEW and OLD still can be accessed by the trigger.)

* A statement-level trigger must not query or modify a mutating table if the trigger is fired as the result of a CASCADE delete.

WHAT IS A MUTATING TABLE ERROR AND HOW CAN YOU GET AROUND IT?

This happens with triggers. It occurs because the trigger is trying to update a row it is currently using. The usual fix involves either use of views or temporary tables so the database is selecting from one while updating the other.

What is a database trigger ? Name some usages of database trigger ?

Database trigger is stored PL/SQL program unit associated with a specific database table.

Usages

Audit data modifications,

Log events transparently,

Enforce complex business rules Derive column values automatically, Implement complex security authorizations.

Maintain replicate tables.

Can you use a commit statement within a database trigger?

No

Can some body give me some example of INSTEAD OF trigger?

Here INSTEAD-OF triggers provide a transparent way of modifying views that cannot be modified directly through SQL DML statements (INSERT, UPDATE, and DELETE). These triggers are called INSTEAD-OF triggers because, unlike other types of triggers, Oracle fires the trigger instead of executing the triggering statement.

```
CREATE TABLE s_dept (
deptno NUMBER PRIMARY KEY,
deptname VARCHAR2(20),
manager_num NUMBER
);

CREATE TABLE e_emp (
empno NUMBER PRIMARY KEY,
empname VARCHAR2(20),
deptno NUMBER REFERENCES dept(deptno),
startdate DATE
);

CREATE VIEW manager_info AS
SELECT d.deptno, d.deptname, e.empno, e.empname
FROM emp e, dept d
WHERE e.empno = d.manager_num;

CREATE TRIGGER manager_info_insert
INSTEAD OF INSERT ON manager_info
REFERENCING NEW AS n -- new manager information
FOR EACH ROW
```

```
DECLARE
empCount NUMBER;
BEGIN

/* First check to make sure that the number of employees
* in the department is greater than one */
SELECT COUNT(*) INTO empCount
FROM emp e
WHERE e.deptno = :n.deptno;

/* If there are enough employees then make him or her the manager */
IF empCount >= 1 THEN

UPDATE dept d
SET manager_num = :n.empno
WHERE d.deptno = :n.deptno;

END IF;
END;
/
```

Chapter 16, Solutions
Triggers
Beginner
1. A trigger is a block of code (whether in PL/SQL, Java, or C) that fires, or executes, in response to a specific event.
2. The statements are:
a. True. You can create a trigger in response to most Data Manipulation Language (DML) operations.
b. False. There is no AFTER EXECUTION.
c. True. Oracle8i allows you to trap "user" level events such as logons.
d. True. You can create triggers that fire when Data Definition Language (DDL) statements are executed.
3. A trigger executes implicitly in response to an event, such as an update to a table. A procedure executes explicitly at the request of a user (or a call from another program).
4. A trigger can have one of two modes: ENABLED (meaning that it fires normally) and DISABLED (meaning that it does not fire, even if its triggering event occurs).
5. (b). A trigger that causes other triggers to fire is called a cascade.
6. Statement-level triggers fire only one time per statement, and row-level triggers fire for each row that is affected by the DML statement.
7. The WHEN clause causes a trigger to fire only when a specific set of user-defined conditions are met.
8. Trigger 2 correctly populates the employee_id column. Trigger 1, which attempts to set the sequence number using an INSERT statement, illustrates a fairly common mistake. While Trigger 1 compiles successfully, it will probably produce the following error when it's executed:
ORA-00036: Maximum number of recursive sql levels (50)
This error is generated because each execution of the trigger results in another execution of the same trigger, eventually exceeding the number of allowed recursive SQL levels. This limit was introduced to prevent such mistaken constructions from resulting in an infinite loop.
9. The pseudo-columns shown are:
a. Valid.
b. Invalid.
c. Invalid.
d. Valid.
e. Valid.
f. Invalid.
g. Invalid.

The OLD, NEW, and PARENT pseudo-columns can be used only in row-level triggers. PARENT, introduced in Oracle 8i, refers to the current row of the parent table for a trigger defined on a nested table. OLD and NEW refer to the following:
* Old and new values of the current row of the relational table.
* Old and new values of the row of the nested table if the trigger is defined on a nested table (Oracle8i).
* For a trigger defined on an object table or view, OLD and NEW always refer to the object instances.
TIP:
OLD and NEW are the default names; specifying the REFERENCING clause of the trigger can change these names.
10. You must have one of the following privileges:
CREATE TRIGGER Allows you to create a trigger in your own schema for a table or view owned by the same schema
CREATE ANY TRIGGER Allows you to create a trigger in any user's schema on a table owned by any schema
ADMINISTER DATABASE TRIGGER Allows you to create database level triggers (e.g., SERVERERROR, LOGON, LOGOFF, etc.)
11. The ALTER ANY TRIGGER privilege allows you only to enable/disable a trigger. The CREATE ANY TRIGGER privilege allows you to create a new trigger in any schema or recreate an existing trigger without having to explicitly drop it and then create it again.
When a trigger is created, it is compiled and stored in the database. If you want to change the text of the trigger, you cannot just "edit" a piece of code; you have to change the definition in the database. To do this, you have to either use the DROP TRIGGER and CREATE TRIGGER commands or the CREATE OR REPLACE TRIGGER command.
12. The error occurs in the WHEN clause; when referring to the new or old values in a WHEN clause, you must omit the colon:

```
CREATE OR REPLACE TRIGGER emp_before_ins_t
  BEFORE INSERT
  ON employee
  FOR EACH ROW
  WHEN (NEW.mgr is null)
BEGIN
  IF (:NEW.sal > 800)
  THEN
    :NEW.sal := 850;
  END IF;
END;
```

Also, note that the WHEN condition must be a SQL condition, rather than a PL/SQL condition. PL/SQL functions or methods cannot be invoked from the trigger WHEN clause.
13. When Oracle compiles the trigger definition, it parses the entire code and reports all the errors encountered during the compilation phase; you can see these errors by querying the trusty USER_ERRORS data dictionary view:

```
SELECT line, position, text
  FROM user_errors
 WHERE name = 'MY_TRIGGER'
   AND TYPE = 'TRIGGER'
 ORDER BY sequence
```

In SQL*Plus, you can also use the following shortcut:
SQL> **SHOW ERRORS TRIGGER MY_TRIGGER**
<CLASS="NOTE"Remember that if a trigger compiles with errors, it still gets created but fails during the execution. This means that all triggering DML statements are blocked until the trigger is temporarily disabled, dropped, or replaced with the trigger version that compiles without errors.
14. There is only one way to explicitly recompile a trigger: execute the ALTER command with the COMPILE option:
ALTER TRIGGER my_trigger_name COMPILE
You might have to use this command because triggers, as part of the Oracle object dependency, may become invalid if an object that the trigger depends upon changes.

15. You might want to omit the OR REPLACE option when you first create a trigger to save yourself a headache if a trigger by that name already exists in that schema. Since you're not using the OR REPLACE option, if a trigger already exists by that name, you get the following error:
ORA-04081: trigger name already exists
16. The statements are:
a. Incorrect. Oracle explicitly disallows developers from putting triggers on SYS data dictionary objects, because this could inadvertently modify the behavior of the database. If an attempt is made to create a trigger on a SYS object, Oracle generates the error "ORA-04089: cannot create triggers on objects owned by SYS."
b. Correct. See (a).
c. Incorrect. Prior to Oracle8i, DML events were the only events that could be trapped by triggers. Oracle8i introduced the ability to trap other events as well, such as database-level events (STARTUP, SHUTDOWN, etc.) and DDL events (CREATE, DROP, etc.).
17. The statements are:
a. False. You can create triggers only for certain types of schema objects.
b. True. Oracle7 allowed the triggers to be created at the TABLE level.
c. False. Oracle8 added the ability to define triggers for VIEWs, which opened the road to fully updateable views in Oracle.
d. Oracle8i introduced the triggers created on NESTED TABLE level.
e. True. As of Oracle8i, you can also use the ON clause to define triggers for database or schema-level events. The DATABASE keyword specifies that the trigger is defined for the entire database, and the SCHEMA keyword specifies that the trigger is defined for the current schema.
18. (c). Here is the required syntax:
```
CREATE OR REPLACE TRIGGER upd_employee_commision
  AFTER UPDATE OF comm
  ON emp
  FOR EACH ROW
BEGIN
  <<Trigger logic>>
END;
```
19. (d). The trigger already fires just once for each INSERT operation on the emp table.
20. The trigger fails because the statement-level trigger cannot reference a pseudo-column name such as :NEW or :OLD. This is only available for row-level triggers.
21. A mutating table is a table that is currently being modified by an UPDATE, DELETE, or INSERT statement, or it is a table that might need to be updated by the effects of a declarative DELETE CASCADE referential integrity constraint. The SQL statements of a trigger cannot read from (query) or modify a mutating table of the triggering statement.
Intermediate
22. Executing the procedure results in an error because DBMS_SQL treats :OLD and :NEW as ordinary bind variables; consequently, it complains that not all the binds have been assigned. In short, the :OLD and :NEW predicates can't be directly referenced in statements executed via dynamic SQL.
23. The following template shows how to raise an error when a DML statement violates a business rule:
```
CREATE OR REPLACE TRIGGER secure_del_trigger
  BEFORE DELETE
  ON emp
  FOR EACH ROW
DECLARE
  unauthorized_deletion   EXCEPTION;
BEGIN
  IF <your business rule is violated> THEN
    RAISE unauthorized_deletion;
  END IF;
EXCEPTION
  WHEN unauthorized_deletion
  THEN
    raise_application_error (-20500,
     'This record cannot be deleted');
END;
/
```

24. (b). Object privileges must be granted directly by the owner and cannot be acquired through database roles.
25. The triggers are:
a. Invalid. The trigger explicitly calls SQL COMMIT, which is forbidden in Oracle.
b. Invalid. The trigger dynamically creates a sequence via the DDL statement CREATE SEQUENCE that is issuing an implicit commit. SQL DDL statements are not allowed in triggers because DDL statements issue implicit COMMITS upon completion.
c. Valid. A system trigger is the only type of trigger that allows CREATE/ALTER/DROP TABLE statements and ALTER...COMPILE in the trigger body, despite the fact it issues an implicit COMMIT.
26. No. The restriction still applies because a stored procedure, even though it's called by a trigger, still runs within the trigger's transaction context.
27. You can use Oracle8i 's autonomous transaction pragma to start a new context. The following trigger illustrates how to use dynamic SQL to execute a DDL statement, which requires an implicit commit, inside a trigger:

```
CREATE OR REPLACE TRIGGER format_table_trig
  AFTER INSERT
  ON format_table
  FOR EACH ROW
  WHEN (new.tablecode = 3334)
DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
  seq_sql        VARCHAR(200);
  cursor_handle  INTEGER;
  execute_ddl    INTEGER;
BEGIN
  seq_sql := 'CREATE SEQUENCE ' ||
         SUBSTR (:new.table_id, 1, 21) ||
         '_SEQ START WITH 0 INCREMENT BY 1 MINVALUE 0';
  cursor_handle := DBMS_SQL.open_cursor;
  DBMS_SQL.parse (cursor_handle, seq_sql, DBMS_SQL.native);
  execute_ddl := DBMS_SQL.execute (cursor_handle);
  DBMS_SQL.close_cursor (cursor_handle);
END;
```

Prior to Oracle8i, you needed to use database pipes (via the DBMS_PIPE package) to achieve this sort of transaction isolation. You first wrote a program to create the sequence (or create a record into a logging table), started it in another process/session, and then used DBMS_PIPE to send it requests from inside your trigger. Since the procedure in the other session was completely isolated from the main transaction, you circumvented the restrictions on transactions inside a trigger. Of course, this method was a lot more work!
28. The statements are:
a. False. A trigger can contain more than 32 lines of code.
b. True. A trigger's size is limited to 32K.
c. True. Only 32 triggers can cascade at one time.
d. True. The maximum size of a string that can hold a RAW or LONG RAW is 32K.
29. The following SQL*Plus script enables or disables all the triggers in the current user's schema:

```
/* Filename on web page: set_trigger_status.sql */
SET VERIFY OFF;
SET SERVEROUTPUT ON;

PROMPT  Program to enable/disable user triggers
ACCEPT op PROMPT '(E - enable, D - disable): '


DECLARE
```

```
   cur INTEGER;
   done EXCEPTION;
   cnt NUMBER := 0;
BEGIN
   FOR user_trg IN (SELECT trigger_name
                      FROM user_triggers)
   LOOP
      BEGIN
         cnt := cnt + 1;
         cur := DBMS_SQL.open_cursor;

         IF UPPER ('&&op') = 'E'
         THEN
            DBMS_SQL.parse (
               cur,
               'ALTER TRIGGER  ' ||
               user_trg.trigger_name ||
               ' ENABLE',
               DBMS_SQL.native
            );
         ELSIF UPPER ('&&op') = 'D'
         THEN
            DBMS_SQL.parse (
               cur,
               'ALTER TRIGGER ' ||
               user_trg.trigger_name ||
               ' DISABLE',
               DBMS_SQL.native
            );
         ELSE
            DBMS_OUTPUT.put_line (
               'Invalid input argument passed'
            );
            DBMS_SQL.close_cursor (cur);
            RETURN;
         END IF;

         DBMS_SQL.close_cursor (cur);
      EXCEPTION
         WHEN OTHERS
         THEN
            DBMS_OUTPUT.put_line (
               SQLCODE || '-' || SQLERRM
            );
            DBMS_SQL.close_cursor (cur);
      END;
   END LOOP;

   IF UPPER ('&&op') = 'E'
   THEN
      DBMS_OUTPUT.put_line (
         cnt || ' triggers enabled'
      );
   ELSIF UPPER ('&&op') = 'D'
   THEN
      DBMS_OUTPUT.put_line (
         cnt || ' triggers disabled'
      );
   END IF;
END;
/
```
Expert

30. Triggers of the same type fire in the order of their respective object identifiers (OIDs). OIDs, which are assigned by Oracle when the trigger is created, are beyond the designer's control. Consequently, the order of firing triggers is not guaranteed and cannot be controlled. The best approach is to make sure that the trigger design is independent of the order of trigger firing.

31. The trigger is clearly designed to fire for DML update events and only when the new salary doesn't equal the old salary. The way the trigger is written at present, it fires unnecessarily across a wide range of DML events that occurs on the employee table. You can use the WHEN clause to eliminate these unnecessary executions:
```
CREATE OR REPLACE TRIGGER employee_upd_t1
   AFTER UPDATE OF salary
   ON employee
   FOR EACH ROW
   WHEN (old.salary <> new.salary)
BEGIN
   employee_pkg.update_emp (:new.employee_id, :new.salary);
END;
```
32. At first, you might be tempted to try something like this:
```
CREATE OR REPLACE TRIGGER employee_t1
   BEFORE DELETE
   ON emp
   FOR EACH ROW
BEGIN
  UPDATE emp
    SET mgr = null
  WHERE mgr = :new.empno;
END;
/
```
Unfortunately, this trigger results in the mutating trigger error "ORA-04091 table name is mutating, trigger/function may not see it." You can use a combination of packaged variables and different types of triggers to solve this problem.

The first step is to create a package containing an index-by table to hold the ids of the managers who have been deleted:
```
CREATE OR REPLACE PACKAGE mutating_table_pkg
IS
   TYPE array IS TABLE OF emp%ROWTYPE
      INDEX BY BINARY_INTEGER;

   emp_values   array;
   empty        array;
END;
/
```
The second step is to create a statement-level BEFORE DELETE trigger that fires at the beginning of the transaction; its only purpose is to initialize the emp_values table to make sure it is empty:
```
CREATE OR REPLACE TRIGGER mutating_trig_1
   BEFORE DELETE
   ON emp
BEGIN
   mutating_table_pkg.emp_values := mutating_table_pkg.empty;
END;
/
```

The third step is to create a row level BEFORE UPDATE trigger to populate the emp_values tables with the employee numbers of the rows that are being deleted. This is the only type of processing implemented in this trigger; the UPDATE statement is intentionally removed from this trigger because it caused the "mutating table" problem in the first place:

```
CREATE OR REPLACE TRIGGER mutating_trig_2
  BEFORE DELETE
  ON emp
  FOR EACH ROW
  WHEN (old.job = 'MANAGER')
DECLARE
  i   NUMBER := mutating_table_pkg.emp_values.COUNT + 1;
BEGIN
  mutating_table_pkg.emp_values (i).empno := :old.empno;
END;
/
```

The final step is to create a statement-level AFTER DELETE that uses the array of managers to modify the employee records. At this point, the employee table is no longer a mutating table (undergoing changes), so you're free to make update statements :

```
CREATE OR REPLACE TRIGGER mut_trg_3
  AFTER DELETE
  ON emp
BEGIN
  FOR i IN 1 .. mutating_table_pkg.emp_values.COUNT
  LOOP
    UPDATE emp
      SET mgr = NULL
     WHERE mgr = mutating_table_pkg.emp_values (i).empno;
  END LOOP;
END;
/
```

33. As of Oracle8iOracle8i, the only way to log in to a database that has an invalid AFTER LOGON trigger is to use a DBA utility that can CONNECT INTERNAL (e.g., Server Manager). Here's a trace of a SVRMGR session to disable the trigger:

```
SVRMGR> connect internal
Connected.
SVRMGR> ALTER TRIGGER on_logon DISABLE;
Statement processed.
```

34. Because the trigger is running as an autonomous transaction, the aggregate query on the emp tables doesn't see the rows inserted by the calling transaction. Consequently, the trigger doesn't correctly record the department's salary history.

35. The first trigger, which is governed by the local object dependency mechanism provided by Oracle, is invalidated immediately after the UPDATE_BONUS procedure is recompiled. Since it's recompiled and revalidated automatically, the next execution of the trigger fires successfully.

The second trigger, which refers to a remote procedure, is not immediately invalidated and revalidated after UPDATE_BONUS@RDB is recompiled. Consequently, the trigger produces the following stack of errors:

```
ORA-04068: existing state of packages has been discarded
ORA-04062: timestamp of procedure "UPDATE_BONUS" has been changed
ORA-06512: at "REM_PROC_TRIGGER", line 2
ORA-04088: error during execution of trigger 'REM_PROC_TRIGGER'
```

Constraints and Triggers

Constraints are declaractions of conditions about the database that must remain true. These include attributed-based, tuple-based, key, and referential integrity constraints. The system checks for the violation of the constraints on actions that may cause a violation, and aborts the action accordingly. Information on SQL constraints can be found in the textbook. The Oracle implementation of constraints differs from the SQL standard, as documented in Oracle 9i SQL versus Standard SQL.

Triggers are a special PL/SQL construct similar to procedures. However, a procedure is executed explicitly from another block via a procedure call, while a trigger is executed implicitly whenever the triggering event happens. The triggering event is either a INSERT, DELETE, or UPDATE command. The timing can be either BEFORE or AFTER. The trigger can be either row-level or statement-level, where the former fires once for each row affected by the triggering statement and the latter fires once for the whole statement.

* Constraints:
* Deferring Constraint Checking
* Constraint Violations
* Triggers:
* Basic Trigger Syntax
* Trigger Example
* Displaying Trigger Definition Errors
* Viewing Defined Triggers
* Dropping Triggers
* Disabling Triggers
* Aborting Triggers with Error
* Mutating Table Errors

Deferring Constraint Checking
Sometimes it is necessary to defer the checking of certain constraints, most commonly in the "chicken-and-egg" problem. Suppose we want to say:

```
CREATE TABLE chicken (cID INT PRIMARY KEY,
                  eID INT REFERENCES egg(eID));
CREATE TABLE egg(eID INT PRIMARY KEY,
              cID INT REFERENCES chicken(cID));
```

But if we simply type the above statements into Oracle, we'll get an error. The reason is that the CREATE TABLE statement for chicken refers to table egg, which hasn't been created yet! Creating egg won't help either, because egg refers to chicken.
To work around this problem, we need SQL schema modification commands. First, create chicken and egg without foreign key declarations:

```
CREATE TABLE chicken(cID INT PRIMARY KEY,
                  eID INT);
CREATE TABLE egg(eID INT PRIMARY KEY,
              cID INT);
```

Then, we add foreign key constraints:

```
ALTER TABLE chicken ADD CONSTRAINT chickenREFegg
   FOREIGN KEY (eID) REFERENCES egg(eID)
   INITIALLY DEFERRED DEFERRABLE;
ALTER TABLE egg ADD CONSTRAINT eggREFchicken
   FOREIGN KEY (cID) REFERENCES chicken(cID)
   INITIALLY DEFERRED DEFERRABLE;
```

INITIALLY DEFERRED DEFERRABLE tells Oracle to do deferred constraint checking. For example, to insert (1, 2) into chicken and (2, 1) into egg, we use:

```
INSERT INTO chicken VALUES(1, 2);
INSERT INTO egg VALUES(2, 1);
COMMIT;
```

Because we've declared the foreign key constraints as "deferred", they are only checked at the commit point. (Without deferred constraint checking, we cannot insert anything into chicken and egg, because the first INSERT would always be a constraint violation.)
Finally, to get rid of the tables, we have to drop the constraints first, because Oracle won't allow us to drop a table that's referenced by another table.

```
ALTER TABLE egg DROP CONSTRAINT eggREFchicken;
ALTER TABLE chicken DROP CONSTRAINT chickenREFegg;
DROP TABLE egg;
DROP TABLE chicken;
```

Constraint Violations
In general, Oracle returns an error message when a constraint is violated. Specifically for users of JDBC, this means an SQLException gets thrown, whereas for Pro*C users the SQLCA struct gets updated to reflect the error. Programmers must use the WHENEVER statement and/or check the SQLCA contents (Pro*C users) or catch the exception SQLException (JDBC users) in order to get the error code returned by Oracle.
Some vendor specific error code numbers are 1 for primary key constraint violations, 2291 for foreign key violations, 2290 for attribute and tuple CHECK constraint violations. Oracle also provides simple error message strings that have a format similar to the following:
ORA-02290: check constraint (YFUNG.GR_GR) violated
or
ORA-02291: integrity constraint (HONDROUL.SYS_C0067174) violated - parent key not found
For more details on how to do error handling, please take a look at Pro*C Error handling or at the Retrieving Exceptions section of JDBC Error handling.

System Privileges Related To Table Triggers
create trigger
create any trigger
administer database trigger
alter any trigger
drop any trigger

Table Trigger Firing Options
-- before constraints are applied
BEFORE INSERT
BEFORE UPDATE
BEFORE DELETE

-- after constraints are applied
AFTER INSERT
AFTER UPDATE
AFTER DELETE

Maximum trigger size
32K - but you can call procedures and function in triggers to perform processing
Create Row Level Triggers
Note: After row triggers create less UNDO than Before row triggers so use After when possible.

Trigger With Autonomous Transaction
The same simple trigger as an autonomous transaction
```
CREATE OR REPLACE TRIGGER t_autonomous_tx
AFTER INSERT
ON t2

DECLARE

PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN
UPDATE t3
SET cnt = cnt + 1;
COMMIT;

END t_autonomous_tx;
/
```

Mutating Trigger Fix With Autonomous Transaction

```
CREATE TABLE t1 (x int);
CREATE TABLE t2 (x int);

INSERT INTO t1 VALUES (1);

SELECT * FROM t1;
SELECT * FROM t2;

CREATE OR REPLACE TRIGGER t_trigger
AFTER INSERT
ON t1
FOR EACH ROW

DECLARE

i PLS_INTEGER;

BEGIN
SELECT COUNT(*)
INTO i
FROM t1;

INSERT INTO t2
VALUES
(i);
END;
/

INSERT INTO t1 VALUES (1);

SELECT COUNT(*) FROM t1;
SELECT COUNT(*) FROM t2;

CREATE OR REPLACE TRIGGER t_trigger
AFTER INSERT ON t1 FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS_TRANSACTION;
i PLS_INTEGER;

BEGIN
SELECT COUNT(*)
INTO i
FROM t1;

INSERT INTO t2
VALUES
(i);
COMMIT;
END;
/

INSERT INTO t1 VALUES (1);
```

```
SELECT COUNT(*) FROM t1;
SELECT COUNT(*) FROM t1;
```

TRIGGERS

* It is a stored PL/SQL program unit associated with a specific database table
* Oracle executes (fires) the trigger automatically whenever a given SQL operation affects the table
* They are invoked implicitly
* They are useful for customizing a database
* They should be used only when necessary
* Can automatically generate derived column values
* Can prevent invalid transactions
* Can enforce complex security authorizations
* Can enforce referential integrity across nodes in a distributed database
* Can enforce complex business rules
* Can provide transparent event logging
* Can provide sophisticated auditing
* Can maintain synchronous table replicates
* Can gather statistics on table access
* Can derive column values automatically
* Can restrict DML operations to regular business hours

DDL EVENT TRIGGERS (Can be created by DBA)

1) To prevent a Scott from dropping table TT

```
create or replace trigger prevent_drop
Before   Drop on Scott.Schema
Begin
     If ora_dict_obj_owner = 'SCOTT'
         and
         ora_dict_obj_name  = 'TT'
         and
          ora_dict_obj_type = 'TABLE'
     Then
                raise_application_error(-20020,'Cannot drop table TT');
     End If;
End;
```

```
        2) --To keep a track of Scott new objects
            --The new objects should get added in a table ScottObjects
             --ScottObjects is owned by DBA

create table ScottObjects
(Object_name varchar2(30),
  Date_of_Creation date);

create or replace trigger Put_New_Objs
After CREATE ON Scott.Schema
Begin
   Insert into ScottObjects
   Values(ora_dict_obj_name,Sysdate);
End;
```

```
3) --To keep a track of Scott's dropped objects
    --The dropped objects should get added in a table ScottDrop

create table ScottDrop
(Object_name varchar2(30),
  Date_of_Creation date);

create or replace trigger Put_Drop_Objs
After DROP ON Scott.Schema
Begin
   Insert into Scottdrop
   Values(ora_dict_obj_name,Sysdate);
End;
4) --Preventing Scott to drop column h2 of table hh
 --Step 1 Login as scott
--create table hh
--(h1 number,
--h2 number,
--h3 number);

-- Step 2 Log in as sysdba

Create or Replace Trigger Prev_Drop_Col
Before ALTER on Scott.Schema
Begin
   If ora_dict_obj_name = 'HH'
      and
       ora_is_drop_column('H2')
     Then
    raise_application_error(-20067,'Cannot drop column h2');
   End If;
End;


5) --To prevent Scott from modifying the data type of column h3 of hh table
     --Log in as DBA

create or replace trigger Prev_Modify
Before ALTER on Scott.Schema
Begin
  If ora_dict_obj_name = 'HH'
      and
     ora_is_alter_column('H3')
   Then
    raise_application_error(-20045,'Cannot modify column H3');
  End If;
End;
```

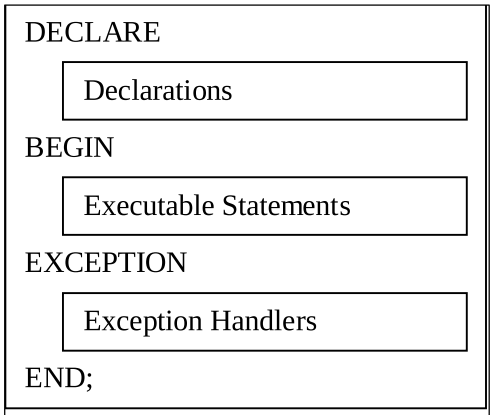
PL/SQL features ---

- PL/SQL is an extension of SQL
- It is an application development language containing procedural statements and commands along with SQL commands
- It bridges the gap between database technology and procedural programming languages
- It allows you to process data using flow control statements like iterative loops and conditional branching

u Uses procedural techniques of control, looping and branching
u Supports SQL i.e. cursor operations, exceptions, functions and transactional commands
u Variables and constants, robust error handling and functions
u Adds functionality to non-procedural tools such as SQL*Forms
u Developers using SQL*Forms can enter an entire PL/SQL block using a single trigger

Structure of PL/SQL

u **Standard PL/SQL code segment is called a Block**
u **A block consists of three parts or sections**
    u **Declaration Part**
    u **Executable Part**
    u **Exception Handling Part**



Sections of a PL/SQL Block

u Dec
fined
u Executable Part
    u mandatory part which consists of executable statements
u Exception Handling Part
    u optional part which consists of code for handling errors (runtime)

Pl/SQL Files -

u PL/SQL programs can be written in any editor and saved as files with .sql extension
u Can also use "ED" command in SQL*Plus to create a PL/SQL program file
u Use the "@ <filename>" command to execute a PL/SQL program file
    u **Variables --**
u Used to store results of a query for later processing, or to calculate values to be inserted into database tables
u Can be used anywhere in an expression, either in SQL or PL/SQL statements
u Must be declared before referencing it in other statements, including other declarative statements
u Are declared by specifying the name along with the datatype
u Can be declared to be of any datatype native to Oracle
u Examples

```
oldfare NUMBER(5);
 m_name VARCHAR(15);
```

(Note – Set Serveroutput On has to be given when a session starts for displaying the output statements)

```
declare
      x number;
 begin
        x := 67;
      dbms_output.put_line(x);
      dbms_output.put_line('The value of x is '|| x);
 end;
```

Declaring variable in declare block.
Assigning value in in begin block using := .
Output statement is dbms_output.put_line
Concatenation operator is ||
Command terminator is ; after end
********Declaring and initialising variables together**

```
declare
          y number := 100;
         begin
            dbms_output.put_line('The value of y is '|| y);
end;
```

Taking value from the user using &

```
declare
        z number;
        a varchar2(10);
 begin
         z := &z;
         a := '&a';
            dbms_output.put_line('Z is  '|| z);
            dbms_output.put_line('A is '|| a);
 end;
```

/*Cannot declare or initialise more than one variable simulataneously*/

```
declare
         a number;
         b number;
         c number;
begin
         a := 67; b := 90;  c := 87;
        dbms_output.put_line(a);
        dbms_output.put_line(b);
end;
```

A constant number has to declared and initialised in the declare block only using CONSTANT keyword.   Value cannot be changed

```
declare
      r CONSTANT number :=100;
 begin
   /*   r := r + 100; Not possible*/
             dbms_output.put_line(r);
end;
```

/*Assigning value to variable from a cloumn of a table using select into clause*/

```
declare
        x number;
begin

            Select sal Into x from emp
              where ename = 'SMITH';
               dbms_output.put_line('Salary of Smith is '|| x);
 end;
```

/* Selecting ename,sal from emp
Use of more than one columns value with Into clause*/

```
declare
        n varchar2(50);
        s number;
begin
        select ename,sal Into n,s
        from emp
        where ename = 'SMITH';
        dbms_output.put_line(n);
        dbms_output.put_line(s);
end;
```

% Type Attribute –

- **Provides datatype of a variable or column**
- **Useful when declaring a variable that refers to a column in a database**
  - **exact datatype of column need not be known**
  - **if column definition changes, variable datatype changes accordingly at runtime**
- **Example**

oldfare fare.first_fare%TYPE;
newfare oldfare%TYPE;

```
declare
      a emp.ename%type;
      b emp.sal%type;
      c emp.deptno%type;
   /*Using %TYPE attribute for variable data type*/
begin
       select ename,sal,deptno
       into a,b,c
       from emp
       where ename = 'KING';
       dbms_output.put_line(a ||'-'||  b ||'-' || c);
  end;
```

%RowType Attribute –

- **Useful when declaring a record variable having same structure as a row in a table or view, or as a row fetched from a cursor**
- **Fields in the record have same names and datatypes as the columns in the table/view**
- **Example**

emp_rec employee%ROWTYPE;

u **A specific field can be referenced using**

emp_rec.emp_num;

```
declare
       E emp%rowtype;
       /*rowtype attribute holds the datatype of the columns of the
        entire row*/
begin
       select ename,sal,deptno INTO  E.ename,E.sal,E.deptno
       from emp
       where ename = 'MARTIN';
        dbms_output.put_line(E.sal);
      dbms_output.put_line(E.ename);
      dbms_output.put_line(e.deptno);
```

```
 end;
```

Conditional Statements – IF

- u The selection structure tests a condition, then executes one sequence of statements instead of another, depending on the condition
- u There are three forms of statements
  - – IF-THEN
  - – IF-THEN-ELSE
  - – IF-THEN-ELSIF
- u Sequence of statements is executed only if the condition evaluates to TRUE
- u If condition evaluates to FALSE or NULL, it does nothing
- u In either case control passes to next statement after the IF-THEN structure

```
IF <condition> THEN
statements;
END IF;
```

- u Sequence of statements in the ELSE clause is executed only if the condition evaluates to FALSE or NULL

```
IF <condition> THEN
statements;
ELSE
statements;
END IF;
--------------------------------------------------------------------------------
declare
   /*Simple if condition */
    x number;
  begin
        x := &x;
        if x >= 35 then
        dbms_output.put_line('Passed');
         else
          dbms_output.put_line('Failed');
         end if;
end;
-----------------------------------------------------
```

IF-THEN-ELSIF Structure

- u This construct allows selection of action from several mutually exclusive alternatives
- u The IF statement can have any number of ELSIF clauses
- u The final ELSE is optional
- u Conditions are evaluated one by one from top to bottom

Syntax

```
IF <condition1> THEN
statements;
ELSIF <condition2> THEN
statements;
ELSIF <condition3> THEN
statements;
ELSE
statements;
END IF;
```

Example 1 –

```
Declare
```

```
     y number;
     /*Multiple ifs */
 Begin

           y := &y;

             if y >= 70 then
              dbms_output.put_line('Distinction');
             elsif y >= 60 then
              dbms_output.put_line('First class');
              elsif y >= 50 then
              dbms_output.put_line('Second class');
              elsif y >= 35 then
              dbms_output.put_line('Passed');
              else
              dbms_output.put_line('Failed');
              end if;
end;
```

Example 2

```
 create table adm
(Name varchar2(30),
Marks number(3),
 College varchar2(30),
 Fees number(5));
```

/*Use of multiple if's
Accept name and marks from user.
Depending upon marks entered the college and fees should be decided
and the record should be entered in the adm table.*/

```
Declare
     n adm.name%type;
     m adm.marks%type;
     c adm.college%type;
     f adm.fees%type;

Begin
         n := '&n';
         m := &m;

            if m >= 95 then
                c  :=  'COEP';
                 f :=   10000;
           elsif  m >= 90 then
                c :=  'MIT';
                f := 15000;
            elsif  m >= 85 then
                c := 'VIT';
                f := 22000;
            elsif   m >= 80 then
                c :=  'D Y Patil';
                f := 27000;
            elsif   m >= 75 then
                c := 'Pune Vidyarthi';
                f := 33000;
              else
                   dbms_output.put_line('Cannot get admission');
         end if;

            if c is not null and f is not null then
            dbms_output.put_line('Your College is  '|| c || '  and fees are ' || f);
              Insert into adm
                values(n,m,c,f);
              end if;

end;
```

LOOPS

- u The ability to repeat or skip sections of a block can be achieved with the usage of LOOP or GOTO statements
- u There are three forms of the LOOP statement
    - u LOOP
    - u WHILE-LOOP
    - u FOR-LOOP

LOOP Statement

- u LOOP repeats a sequence of statements
- u Statements to be repeated are placed between keyword LOOP and END LOOP
- u With each iteration of the loop, the sequence of statements is executed, then control resumes at the top of the loop

```
LOOP
statements;
END LOOP;
```

EXIT Statement

10) Used to complete a loop if further processing in a loop is undesirable or impossible
11) There are two forms of the EXIT statement
    12) EXIT
    13) EXIT-WHEN
14) Forces a loop to complete unconditionally
15) Must be placed inside a loop

```
LOOP
statements;
IF <condition> THEN
EXIT;   -- exit loop immediately
END IF;
END LOOP;
-- control resumes here
```

Example of Loop –

/*To show 1 to 10 on screen*/

```
Declare
   x number;
Begin
   x :=  1;
    Loop
          dbms_output.put_line(x);
           x := x + 1;
           exit when x > 10;
```

```
     End Loop;
 End;
```

```
create table five
(no number);
```

/*Inserting mutliples of five in table five*/

```
Declare
    x number;
Begin
        x := 5;
         Loop
               Insert into five
                values(x);
                x := x + 5;
                exit when x > 50;
          End Loop;
End;
```

<u>FOR LOOP</u>

Advantages --
1) No need of declaring loop variable
2) No need of giving condition
3) No need of updation statement (increment or decrement)
4)Code becomes small and compact

Disadvantage --
Updation can be done by only one.

Syntax –

```
FOR <counter> IN [REVERSE]
lower_bound .. higher_bound LOOP
statements;
END LOOP
```

Example 1 of for loop
/*To show 1 to 10 on screen*/

```
begin
    for x in 1..10
    Loop
              dbms_output.put_line(x);
   End Loop;
end;
```

Example 2
/*Reverse for loop 10,9,8 … 1*/

```
Begin
          for  i in REVERSE 1 ..10
           Loop
                  dbms_output.put_line(i);
```

```
            End Loop;
end;
```

Example 3 –
Calculating compound interest for a principal of Rs.100 @10% for each year.
Values will be shown of the CI after each year.

```
create table CI_100
(year number(2),
total number(4));
----------------------------------
Declare
     p number := 100;
     tot number;
/*Calculation of compound interest.
   Rs.100 is principal.
   Rate of interest is 10%.
    Period is 5 years.
 */
Begin
 for y in 1..5
    Loop
    /* Tot variable is getting 10% more than p */
    tot := p + p * 0.10;
    Insert into CI_100
    values(y,tot);
    /*Since the next interest is based on the current interest
      so the tot will be considered as p for the next year*/
    p := tot;
    End Loop;
end;
```

WHILE-LOOP Statement

- u Associates a condition with a sequence of statements enclosed within LOOP-END LOOP
- u Condition evaluated before each iteration
- u If condition evaluates to TRUE, sequence of statements is executed and control resumes at the top of the loop
- u If condition evaluates to FALSE or NULL, loop is bypassed and control passes to next statement
- u Number of iterations depends on the condition and is unknown until the loop completes

```
WHILE <condition>
LOOP
statements;
END LOOP;
```

Example 1 of while loop to show 1 to 15

```
declare
```

```
  x number;
Begin
   x := 1;
   while x <=15
    Loop
            dbms_output.put_line(x);
            x := x + 1;
    End Loop;
end;
```

Example 2 Forces a loop to complete unconditionally

```
declare
 z number;
/*Using break after z reaches to 8*/
Begin
   z := 1;
   while z <=15
    Loop
            dbms_output.put_line(z);
            z := z + 1;
             exit when z = 8;
    End Loop;
end;
```

UNIX COMMANDS

UNIX commands are case sensitive. Type commands exactly as shown; most UNIX commands are lower case. File and directory names can be lower, upper, or mixed case but must be typed exactly as listed.

mkdir
directory-name -
create a directory called
directory-name
more
filename
- display the file contents one screen at a time
head
filename
-
display first few lines of a file
tail
filename
-
display last few lines of a file
ls
directory-name
- list contents of directory
options
:
-a
list all files including invisible files
-l long list - shows ownership, permission, and links
-t list files chronologically
-f

append "*" to executable file name, "/" to directory name
-d
list directories
-u
list files using time of last access instead of time of last modification
-F
mark directories with forward slash, mark executables with an @-sign.
-g
lists ownership of each file or directory
pwd
-display the name of present working directory
CHANGE WORKING DIRECTORY
cd
to change to your home directory
cd
directory-name
to change to another directory
examples:
cd ~
change to home directory
cd test
change to directory test
mv
present-filename new-filename
to rename a file
mv
source-filename destination-directory
to move a file
into another directory
options:
-i
interactive mode. Must confirm file overwrites.
COPY FILES
cp
source-filename destination-filename
to copy a file into another
filename
cp
source-filename destination-directory
to copy a file into another
directory
options:
-i
interactive mode. Must confirm overwrites.

CHANGE FILE ACCESS PERMISSIONS
chmod [
who op permission
]
filename
who
can be any combination of:
u
(user)

g
(group)
o
(other)
a
(all)
(i.e.
ugo
)
op
adds or takes away permission, and can be:
+ (add permission),
- (remove permission), or
= (set to exactly this permission).
permission
can be any combination of
r
(read)
w
(write)
x
(execute)
Example:
chmod a+x
filename
- makes
filename
executable by
everyone
~
home directory (tilde)
~
username
another user's home directory
.
current or working directory
..
parent of working directory
SHELL TOOLS
Wild Cards
single character wild card
arbitrary number of characters
HISTORY: COMMAND REPETITION
history
display list of most recent commands
!
repeat the entire last command line at any point
in the current command line
$
repeat the last word of previous command
line at any point in current command line
:
n
repeat the
n
th argument from previous

line at any point in current command line
^
repeat first argument from previous command
line at any point in the current command line
n
repeat command line
n
!:p
display previous command
string
command beginning with
string
*****
repeat all arguments to previous command
COMMAND I/O
>
command output redirection (create new)
>>
command output redirection (append)
<
command input redirection (from file)
<<
command input (from script or standard input)

PROCESS CONTROL
PROCESS STATUS
ps -
display the status of the current processes
options:
-a
include information about processes owned by other users
-g
display all processes
-u
display user-oriented processes
-x
include processes with no controlling terminals
-gx
display all of your local processes
RUN COMMAND IN BACKGROUND: JOB CONTROL
To run a command in the background, as opposed to the more common method of running commands in the foreground, append an & to the end of a command string. Then, you can type more commands to the command prompt, or even run more commands in the background for simultaneous command execution. (PID-Process ID) can be found by first using the ps command.

Control-Z
stop (interrupt) foreground job
jobs
list of background jobs
bg
run a stopped job in the background
fg
resume stopped job in the background
FILE OPERATIONS
SEARCH FOR PATTERNS IN FILES

grep
search-string filename
[
filename...
] to find and type out lines containing
the string in a file
options:
-v
type out lines that don't contain the string (invert the search)
wc
filename(s)
counts the number of words, lines, and characters in a file
COMPARE FILES
diff
filename1
filename2
compares contents of
filename1
and
filename2
on a
line-by-line basis
mail
address
sends mail to user at address
address
format is
user
@
host.domain
ftp
host.domain
use file transfer protocol to connect to remote host computer
Type ? for commands
compress
filename
compress file and rename it
filename.Z
uncompress
filename Z
uncompress file and rename
filename
cc
filename.c
C compiler
env
lists your environment settings
who
lists users on the local system
finger
username
@
host.domain
looks up information on an
other user
whois

username
display real name of user
clear
clears screen
stty
display terminal characteristics
date
displays current time and date
cal
year
for yearly calendar
cal
month-year
for monthly calendar
rlogin
host.domain
log into remote host computer
telnet
host.domain
user interface to a remote system
whereis
command
locate a command; display its pathnam
spell
filename
report spelling errors
ispell
filename
interactive spell-checker
echo $path
inspect your search path
bc basic calculator
(Control-D to exit)
du display the number of disk blocks use
per directory or file
du -s display your total disk usage

display a one-line summary about
command
man
command

vi Editor(Text Editor)
Typing the ESC (escape) character takes you out of
input mode and into command mode. A partial command may be
ancelled by typing ESC. To enter the vi editor type:
vi
-options
filename
OPTIONS
l
lisp mode: indents appropriately for lisp code,
the () {} [[and]] commands in vi are modified
to have meaning for lisp
r
filename

recover filename after an editor or system crash.
If file is not specified a list of all saved files will
be printed.
R
read only mode
+
/
string
search for string
i
filenamelist
enter vi, read in first file in filenamelist
n
edit next file in filenamelist
n
filenamelist
specify new filenamelist
e
filename
e#
return to original file
e!
filename
edit filename, discard previous buffer
r
filename
place copy of filename below the current line
CONTROL-g
display the current file name and line number of
current line
Modes of Operation
Command normal and initial mode; other modes return to
command mode upon completion.
ESC
(escape) is
used to cancel partial command.
nput To enter the Input mode, type one of the following:
a, i, A, I, o O, c, C, s, S, or R.
Text may then be
entered. Input mode is terminated with
ESC
character.
ESC
cancel unexecuted vi command
Control-C
stop in-progress command
u
counteract last command that changed the buffer
U
counteract changes to current line
REPEATING A COMMAND
.
repeat the last command that changed the buffer
SAVING WORK/EDITING
:w
write file under original name

ZZ
or
:wq
write file under original name, exit vi
:q
exit vi, no changes are saved
:w
filename
write the file under
filename

Wildcard and Multiplier Characters
.
match any one character
*****
match zero or more occurrences of the preceding
character
.*
match any number of characters

VIEWING FILES
cat
file1
Display
file1
to the screen.
head
file1
Display first few lines of
file1
to the screen.
tail
file1
Display the last few lines of
file1
to the
screen.
more
file1
Interactively displays file on screen. Type q
to quit more. Type f to move forward one
screen, b to move backward one screen.
Type h for help using more.
EDITING FILES
pico
file1
A simple text editor.

what is difference between ATTRIBUTE column & GLOBAL_ATTRIBUTE column in a table?
If i want to define more DFF than ATTRIBUTE column in a table then what we should do ?

1. Global_AttributesNN are used by Oracle for localizations. Hence you must not use this at customer site for defining new Flexfields.

2. You have two options
a. Create a flexfield within Flexfield Segment. I think there is a note on metalink.
b. Create a custom screen that is based on custom table. In the custom table you will have a column that references the primary key of the main table upon which standard screen is based. This custom screen can then be integrated using forms personalization or CUSTOM.pll

Is there any solution if I need to add another ATTRIBUTE?

- You can create a FlexField within Flexfield, i.e. reference a Key Flex within one of the segments of DFF
  The unique id of KFF will eb stored in attribute column

  2. You can create a custom screen on custom table, and zoom to that screen by enabling Tools Menu.
  This can be done using forms personalization. the custom table will contain reference to unique key of main screen [i.e. order header id]

How can you define more then one table to table type?

ANS- Specify more than one name in the table name field,
No value req in the table application field
Use join statement in where clause

Descriptive Flexfield

Question: What does DFF mean?
Answer: DFF is a mechanism that lets us **create new fields in screens** that are delivered by Oracle.

Question: Oh good, but can these new fields be added without modifying/customization of the screen?.
Answer: Yes, certainly. Only some setup is needed, but no programmatic change is needed to setup DFF.

Question: Are these DFF's flexible?
Answer: A little flexible, for example, depending upon the value in a field, we can make either Field1 or Field2 to appear in DFF.

Question: So we create new fields in existing screen, but why the need of doing so?
Answer: Oracle delivers a standard set of fields for each screen, but different customers have different needs, hence Oracle lets us create new fields to the screen.

Question: Are these new fields that get created as a result of DFF free text?
I mean, can end user enter any junk into the new fields that are added via DFF?
Answer: If you attach a value set to the field(at time of setup of dff), then field will no longer be free text. The entered value in the field will be validated, also a list of valid values will be provided in LOV.

Question : Will the values that get entered by the user in dff fields be updated to database?
Answer: Indeed, this happens because for each field that you create using DFF will be mapped to a column in Oracle Applications.

Question: Can I create a DFF on any database column?
Answer: Not really. Oracle delivers a predefined list of columns for each table that are meant for DFF usage. Only those columns can be mapped to DFF segments. These columns are named similar to ATTRIBUTE1, ATTRIBUTE2, ATTRIBUTE3 ETC. Usually Oracle provides upto 15 columns, but this number can vary.

Question: Can I add hundreds of fields to a given screen?
Answer: This depends on the number of attribute columns in the table that screen uses. Also, those columns must be flagged as DFF enabled in DFF Registration screen. Don't need to worry much about this because all the ATTRIBUTE columns are by default flagged for their DFF usage.

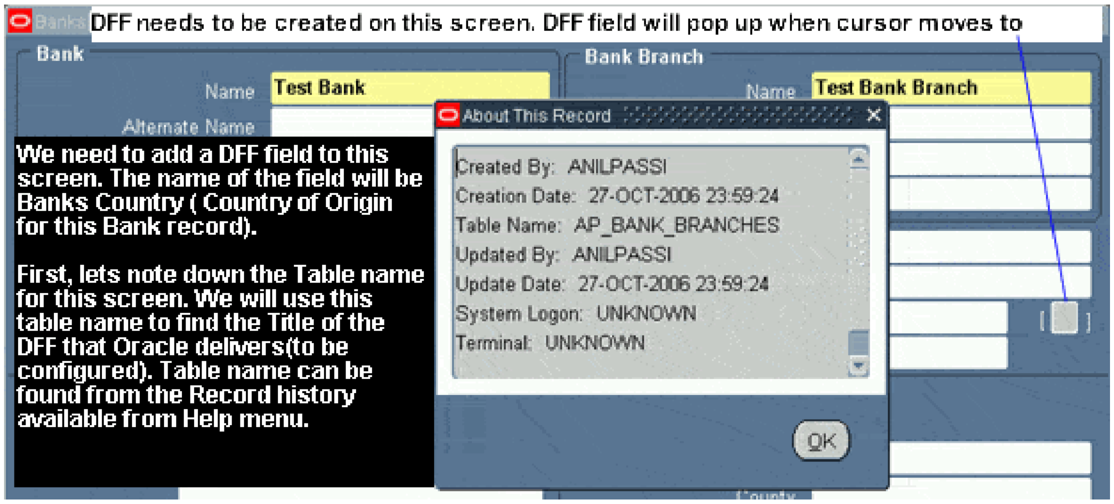
Question: Hmmm, I can see that DFFs are related to table and columns...
Answer: Yes correct. Each DFF is mapped to one table. And also each segment(or call it field) is mapped to one of the attribute columns in that table.

Question: I want these fields to appear in screen only when certain conditions are met. Is it possible?
Answer: Yes, we have something known as **Context Sensitive Descriptive** Flexfields.

In Order to do this, we will follow the below steps(screenshots will follow) :-
1. Navigate to the DFF Registration screen in Oracle Apps and query on Table AP_BANK_BRANCES. Now click on Reference Field
2. Navigate to DFF Segments screen and query on the Title of the “Bank Branch” and Unfreeze the Flexfield and add segments as to Section "GLOBAL Data Elements" as shown in screenshots.

Here are the screenshots......The descriptions are embedded within the screenshots.

We are in "Bank Branches screen" below, that is available in Payables responsibility. We need to add a new field as below.


DFF needs to be created on this screen. DFF field will pop up when cursor moves to

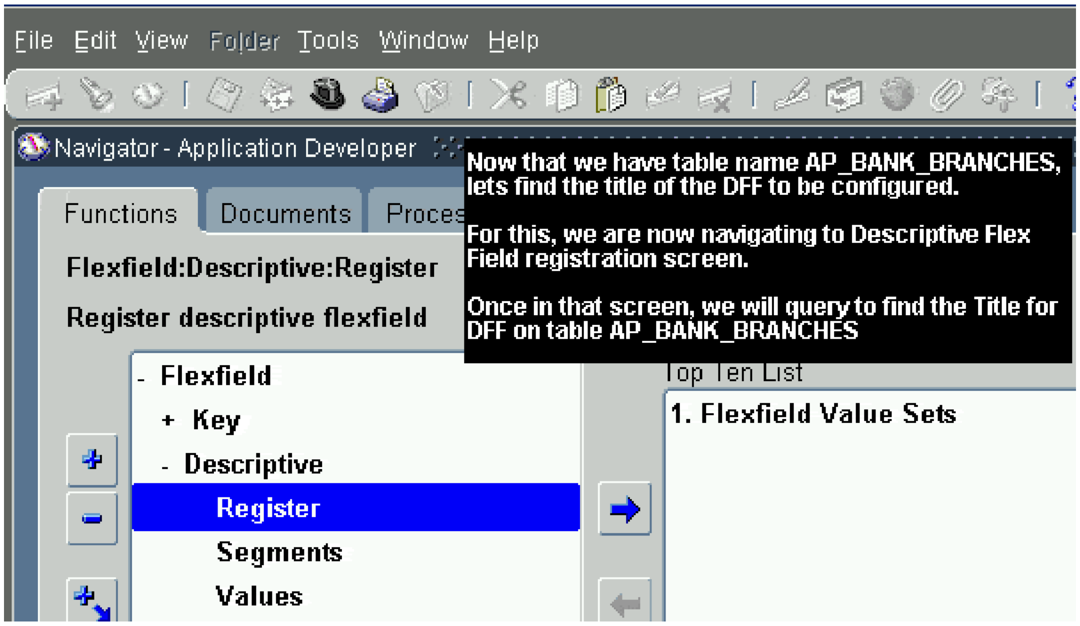
We need to add a DFF field to this screen. The name of the field will be Banks Country (Country of Origin for this Bank record).

First, lets note down the Table name for this screen. We will use this table name to find the Title of the DFF that Oracle delivers(to be configured). Table name can be found from the Record history available from Help menu.

About This Record

Created By: ANILPASSI
Creation Date: 27-OCT-2006 23:59:24
Table Name: AP_BANK_BRANCHES
Updated By: ANILPASSI
Update Date: 27-OCT-2006 23:59:24
System Logon: UNKNOWN
Terminal: UNKNOWN

Once having noted down the table, we try to find the Title of the DFF for that Table. We go to Flexfield/Register


Now that we have table name AP_BANK_BRANCHES, lets find the title of the DFF to be configured.

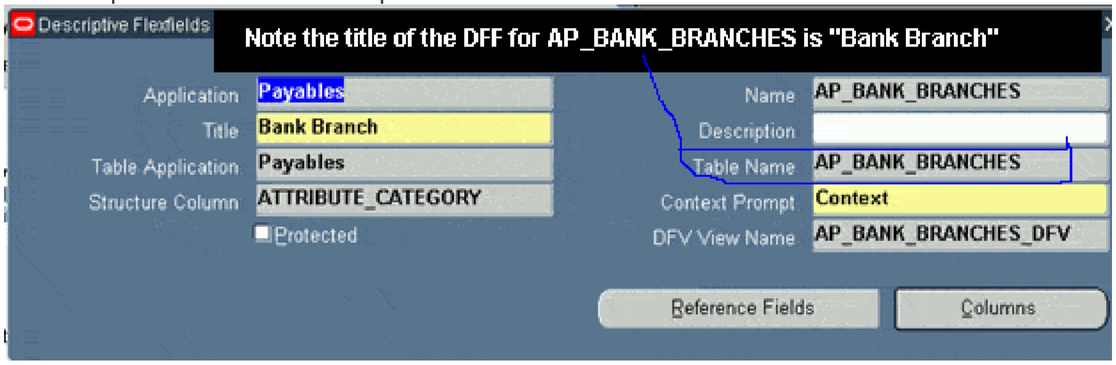
For this, we are now navigating to Descriptive Flex Field registration screen.

Once in that screen, we will query to find the Title for DFF on table AP_BANK_BRANCHES

Flexfield:Descriptive:Register

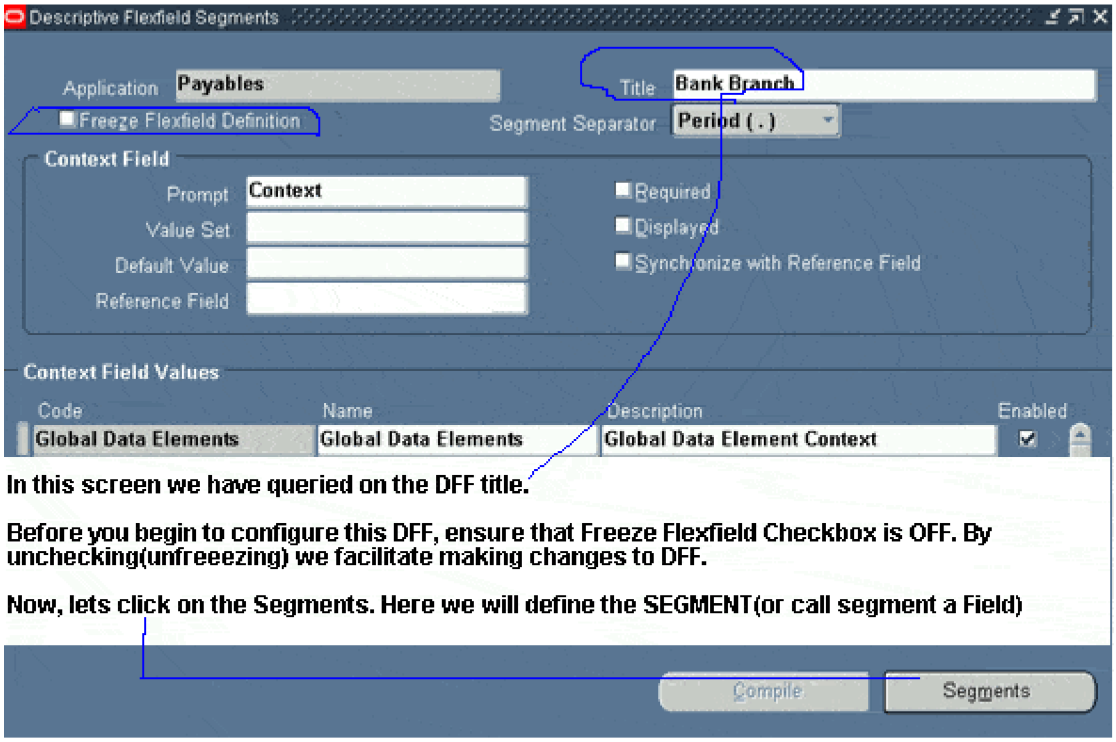
Register descriptive flexfield

Here we pick the Title of the respective DFF


Note the title of the DFF for AP_BANK_BRANCHES is "Bank Branch"

Application	Payables	Name	AP_BANK_BRANCHES
Title	Bank Branch	Description	
Table Application	Payables	Table Name	AP_BANK_BRANCHES
Structure Column	ATTRIBUTE_CATEGORY	Context Prompt	Context
Protected	☐	DFV View Name	AP_BANK_BRANCHES_DFV

Query on that DFF Title from Descriptive Flexfield Segment Screen



Descriptive Flexfield Segments

Application	Payables	Title	Bank Branch
☐ Freeze Flexfield Definition		Segment Separator	Period (.)

Context Field

Prompt	Context	☐ Required
Value Set		☐ Displayed
Default Value		☐ Synchronize with Reference Field
Reference Field		

Context Field Values

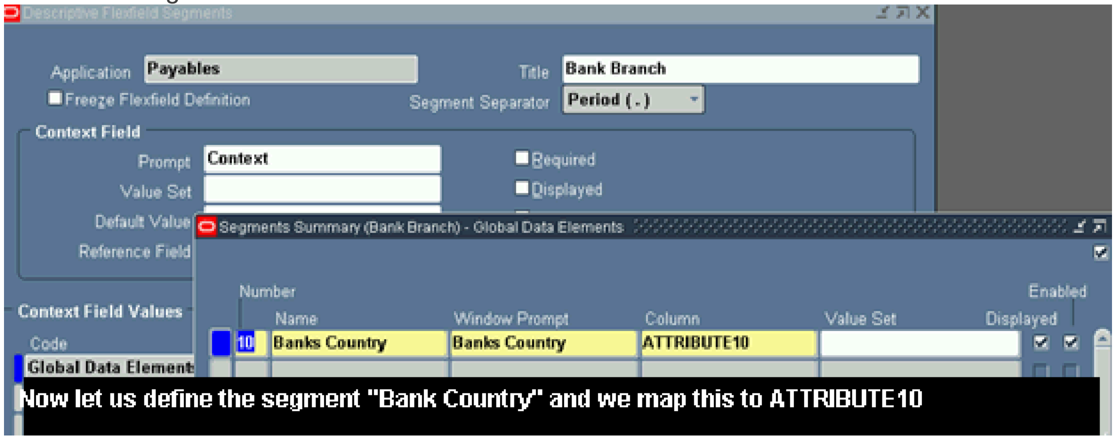
Code	Name	Description	Enabled
Global Data Elements	Global Data Elements	Global Data Element Context	✓

In this screen we have queried on the DFF title.

Before you begin to configure this DFF, ensure that Freeze Flexfield Checkbox is OFF. By unchecking(unfreeezing) we facilitate making changes to DFF.

Now, lets click on the Segments. Here we will define the SEGMENT(or call segment a Field)

Add a new segment under "Global Data Elements"

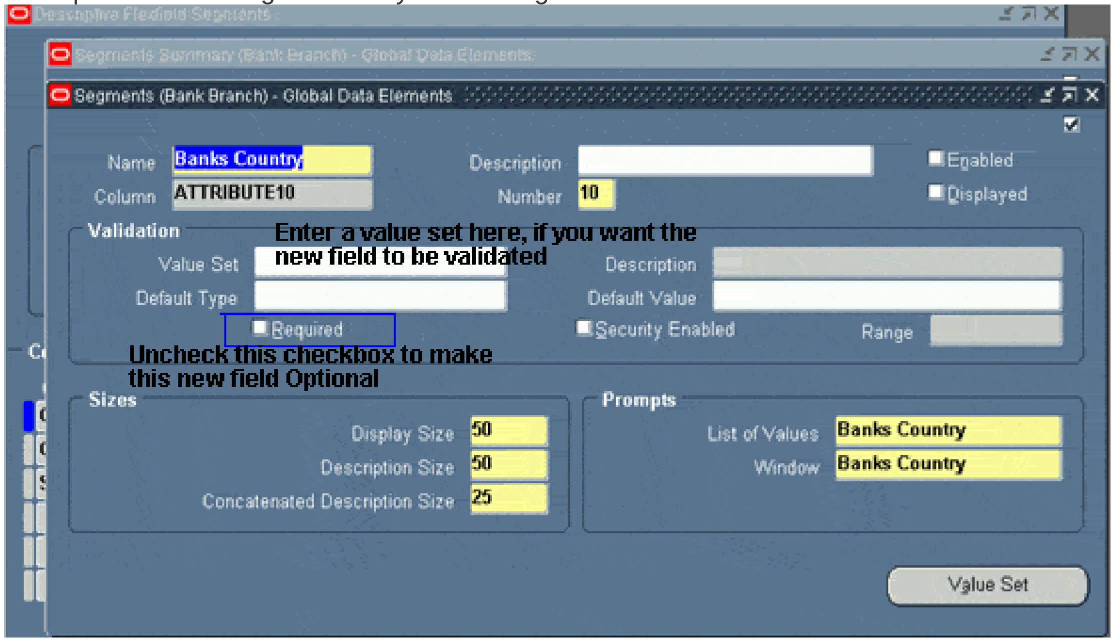


Segments Summary (Bank Branch) - Global Data Elements

Number	Name	Window Prompt	Column	Value Set	Displayed	Enabled
10	Banks Country	Banks Country	ATTRIBUTE10		✓	✓

Now let us define the segment "Bank Country" and we map this to ATTRIBUTE10

The options for making mandatory or enabling validations for the new field.



Descriptive Flexfield Segments

Segments Summary (Bank Branch) - Global Data Elements

Segments (Bank Branch) - Global Data Elements

Name	Banks Country	Description	
Column	ATTRIBUTE10	Number	10

Enabled ☐ Displayed ☐

Validation

Enter a value set here, if you want the new field to be validated

Value Set		Description	
Default Type		Default Value	

☐ Required ☐ Security Enabled Range

Uncheck this checkbox to make this new field Optional

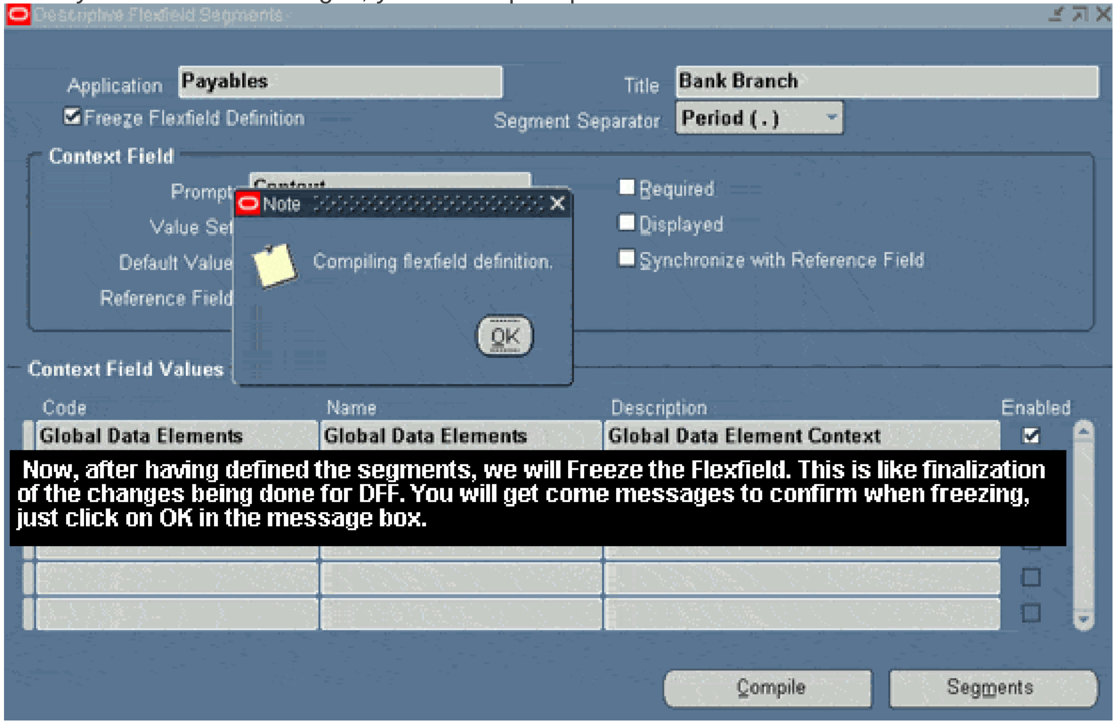
Sizes

Display Size	50
Description Size	50
Concatenated Description Size	25

Prompts

List of Values	Banks Country
Window	Banks Country

Once you finalize the changes, you will be prompted to Freeze the DFF definition. Click on OK



Descriptive Flexfield Segments

Application	Payables	Title	Bank Branch
☑ Freeze Flexfield Definition		Segment Separator	Period (.)

Context Field

Prompt Context

Value Set

Default Value

Reference Field

☐ Required ☐ Displayed ☐ Synchronize with Reference Field

Note

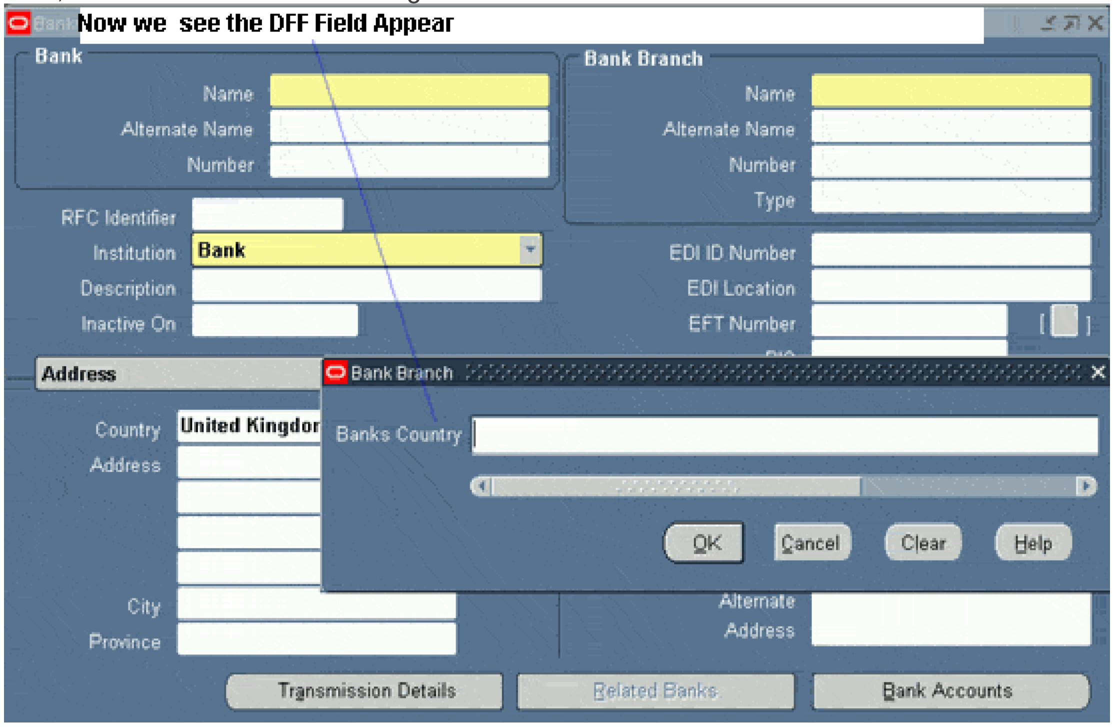
Compiling flexfield definition.

Context Field Values

Code	Name	Description	Enabled
Global Data Elements	Global Data Elements	Global Data Element Context	☑

Now, after having defined the segments, we will Freeze the Flexfield. This is like finalization of the changes being done for DFF. You will get come messages to confirm when freezing, just click on OK in the message box.

Now, we see the fruits of our configuration


Now we see the DFF Field Appear

Bank: Name, Alternate Name, Number

Bank Branch: Name, Alternate Name, Number, Type

RFC Identifier

Institution: Bank

Description

Inactive On

EDI ID Number

EDI Location

EFT Number

Address

Country: United Kingdor

Address

City

Province

Bank Branch

Banks Country

Alternate Address

.
written by Anil Passi , November 15, 2006
hi santosh,
If Oracle does not provide DFF,then do the following....
1. Create a custom screen with custom table.
2. Enable Special Menu in std oracle screen
3. Using **Custom.pl**l or forms personalization, let the user navigate to custom screen to capture the values for additional fields.
Needless to say, the custom table will have a foreign key towards main Oracle Apps table

What is balancing segment
The system insure that journal entries entry are balanced,debit=credit,for each value of the balancing segment you should make the company segment,so that journal entry always balance by company

What is cost center segment
Cost Center define functional area of organization,
Such as accounting,facilities,shipping

REQUEST SECURITY GROUP When a request group is assigned to responsibility then its called REQUEST SECURITY GROUP

When creating table based value, in the where clause we have to follow the following syntax
WHERE column_name =:$FLEX$.param_name (this useful to refer the above one parameter in the query).
What is ADDITIONAL COLUMNS fields (Which is after WHERE clause field) when we create the table based Value set.

What is the difference between po_headers_all and po_headers.

Ans: po_headers_all   --TABLE
po_headers   --VIEW

Q 23:-->can u put data file anywhere in API. (Like CUS_TOP, AU_TOP)?
Yes, we can put, But it is better to put it Bin in directory

how can u handle the exception raised by update statement? if not how can u raise exception?

ans: update does not raise any exception.u have to use another select statement to raise exception.

advantage of writting API's rather than pl/sql?

1) Code of pl/sql can be seen whereas that of API's cant be seen
2) When using APIs we need not know DB structure.

What is significance of MO% or MO operating unit ?

Organization id -- > ensures that data of one operating unit is NOT seen by another op. unit.

Types of PO ?

Standard PO: one time purchase from one supplier.
Blanket   : Long time purchases, item and terms and conditions is fixed, deliveries and qty are flexible
Planned PO : long term delivery schedules, suppliers items are fixed, deliveries are tentative.
contract PO : Terms and conditions is fixed.

what is intercompany relationship ?

relation of transactinos between two organizations (shipping and selling for example).
e.g. Payment terms, invoice generated type etc.

How many types of purchase documents are there ?

Requisition(Int and pur), RFQ(Standard, catalog), Quotation(Std, Bid, Catalog),
PO(Std., PLanned, Blanket, Contract),Releases,Receipt

How can you call one program to another?

u Ans- Using FND_REQUEST.SUBMIT_REQUEST we can call one concurrent program from other concurrent program. For testing in on sql prompt we have to set the SQL environment for Apps support by executing following procedure.
exec FND_GLOBAL.APPS_INITIALIZE(user_id,resp_id,resp_appl_id,security_group_id).
(Parameter values will be available from help menu/examine tab.)

What are the names of the parameters u pass to the Procedure which u register in the apps?

1) retcode in varchar2
2) errbuf   in varchar2

What is the clause in SQL * Loader to program to override data into table

REPLACE

How do you set profile in oracle applications In Application Developer responsibility?

A   Open 'Profile' Function

What is the syntax for loading data through SQL * Loader from multiple files simultaneously

A.   Sqlldr scott/tiger@orcl control = ctlfile
parfile -- parameter file: name of file that contains parameter specifications
parallel -- do parallel load   (Default FALSE)

How to run a concurrent program. What all concurrent programs u have   created.

Ans:- (Definition :- A concurrent program is an instance of an execution file, along with parameter definitions and incompatibilities. Concurrent programs use concurrent program executables to locate the correct execution file.)
Oracle Tool Concurrent Program * A concurrent program written in

* Oracle Reports, PL/SQL package procedures,
SQL*Loader, SQL*Plus, Host Scripting.
How to Run : * Write a execution file and place in correct directory.
* Establish executables in Oracle apps specify execution file and method.
* Define Concurrent Program (Program, Parameters and Incompatibilities)
* Call your Program (- Thu application form, from other concurrent program.
- OR through standard request submission, you must check the 'USE in SRS check box' and register your program parameters when you define your concurrent program. Add your program into the request security group for your custom application.)
I have created reports through concurrent program, load(sql*loader/pl-sql pkg-proc) the file through concurrent program

What is Profile? Explain different levels of Profile.

Ans:- A user profile is a set of changeable options that affects the way your applications run. Oracle Application Object Library establishes a value for each option in a user's profile when the user logs on or changes responsibility. Your user can change the value of profile options at any time a) To create Profile Option. (Profile Option can created by developer in application developer area) b)set the value (Values of the profile option , who will have what value at various levels is set by SYSADMIN). Oracle Application Object Library provides many options that. (Edit profile feature for every user is available to set any value to allow the user).your users can set to alter the user interface of your applications to satisfy their individual preferences.
Profile Option - set at run time like - User Related, responsibility, Sequence, Printer, Security.
Values in 4 Levels(HIEARCHY WISE) :-
A. USER
B. RESPONSIBILITY
C. APPLICATION
D. SITE
Application Developer create the profile.
System Administrator make profile option.
(NOTE:- If any change in value, it will active when you re-login or switch
to the responsibility.)
(Usage in the multi-tier, the profile is biggest impact)

Alerts

u **What are the different business uses of Alerts**

Keep you informed of critical activity in your database
Deliver key information from your applications, in the format you choose
Provide you with regular reports on your database information
Automate system maintenance, and routine online tasks Information about exception conditions.

u **What are the different types of alerts**

You can define one of two types of alerts: an *event alert* or a *periodic alert*.

What is the difference:

event alert	: An event alert immediately notifies you of activity in your database as it occurs.
periodic alert	: checks the database for information according to a schedule you define.

u What database events can cause what actions:
a) an insert and/or an update to a specific database table.

4. What actions can you perform in an alert:
An action can entail sending someone an electronic mail message, running a concurrent program, running an operating script, or running a SQL statement script. You
include all the actions you want Oracle Alert to perform, in an
action set.

u What do you specify when creating an Periodic Alert

a) A SQL Select statement that retrieves specific database information.
The frequency that you want the periodic alert to run the SQL statement.
Actions that you want Oracle Alert to perform once it runs the SQL statement. An action can entail sending the retrieved information to someone in an electronic mail message, running a concurrent program, running an operating script, or running a SQL statement script. You include all the actions you want Oracle Alert to perform, in an action set.

- u Can you define Alert on Oracle Applications Tables. Yes
- u How alert is different from database triggers ?
  - u Code can be modified and viewed in a screen
  - u Periodic alert is not possible thru DB trigger

c) Oracle Alert will also transfer your entire alert definition across databases. You can instantly leverage the work done in one area to all your systems.
d) **Customizable Alert Frequency**
With Oracle Alert, you can choose the frequency of each periodic alert.
You may want to check some alerts every day, some only once a month, still others only when you explicitly request them.

7. Can you define detailed or summary actions in alert.
Yes, **Detail or Summary Actions** You can choose to have Oracle Alert perform actions based on a single exception or a combination of exceptions found in your database.

8. Can you perform actions when NO exceptions are found: **No Exception Actions**
Oracle Alert can perform actions if it finds no exceptions in your database. Same as alert actions.

9. Can you specify History maintenance
Alert History Oracle Alert can keep a record of the actions it takes and the exceptions it finds in your database, for as many days as you specify.

10. What is escalation capabilities in Oracle Alert: **Action Escalation**
You can define a sequence of actions and have Oracle Alert perform the next action in that sequence each time it finds the same exception or exceptions in your database.

- u Can alert check for user response

Yes, it has response processing feature. **Response Processing**
Oracle Alert can take certain predefined actions based on a user's response to an alert message. The response can cause Oracle Alert to send another alert message, run a SQL script or an operating system script, or submit a concurrent request, or any combination of the above.

- u Can it send mail message: Yes it can send. Oracle Alert allows you to send electronic mail messages directly to your mail system.

- u Can you distribute a report, log etc with Alert: Yes With Oracle Alert, you can include a file created by another application as part of an alert message. You can also define an alert that distributes an electronic copy of a report, log file, or any other ASCII file.

- u How can you use PL/SQL functions in Oracle Alert.

15. What are the four steps to create Periodic Alert
Define your periodic alert and specify its frequency
Specify the details for your alert
Define actions for your alert
Create action sets containing the actions you want your alert to perform

u How do you refere the variables whose value is selected in SQL statement of Alert:
You Use Into statement with &variablename

u Any limitation on SQL statement : Yes, 64K
u Can you use SQL statement from a file : Is it necessary to have into clause in it:
Yes you can use a file. If the Select statement in the file does not contain an Into clause,
Oracle Alert automatically inserts an Into clause into the alert Select statement as &OUTPUT1, &OUTPUT2, and &OUTPUT3.

18. Is it necessary that the application that owns an alert and on which executes have to be same: NO but both applications must reside in the same Oracle database and the application that owns the alert has to have Select privileges on the tables listed in the alert Select statement.
u Can you use view as an event alert table: You cannot use a view as the event table for your alert.
u What are the four types of actions in alert can be specified.
There are four types of actions you can create:
message actions
concurrent program actions
operating script actions
SQL statement script actions.

u Are there any limitations in Response processing whan getting inputs back from user.
Yes, User must respond answert in a specific format and content.

u How does Alert works
Oracle Alert checks your database for the exceptions you want to know
about using several concurrent programs:
Periodic Alert Scheduler
Check Periodic Alert
Check Event Alert
Response ProcessorThe Periodic Alert Scheduler (ALEPPE) is a concurrent program that automatically checks your scheduled periodic alerts.

23. How Check Event Alert Works
Once you define an event alert to monitor a table for inserts and/or
updates, any insert or update to the table will trigger the event alert.
When an insert or update to an event table occurs, Oracle Alert submits
to the concurrent manager, a request to run a concurrent program
called Check Event Alert (ALECTC). The concurrent manager runs
this request according to its priority in the concurrent queue. When the
request is run, Check Event Alert executes the alert Select statement.

The Response Processor (ALPPIM) is the Oracle Alert concurrent
program that processes responses to an alert message.

24. What is the purpose of **:DATE_LAST_CHECKED**
An Oracle Alert
implicit input that contains the date and
time that an alert was last checked. Oracle
Alert automatically provides the value for

:DATE_LAST_CHECKED. You must use
:DATE_LAST_CHECKED to create self–referencing alerts.

25. Action History A record of the actual actions
performed for each action set check that
includes the value substituted for each
output.

26. Can you perform actions if recipients doesn't respond:
Yes using No Response field to specify The actions you want Oracle
Alert to perform if the recipient does not respond within the number of Response
Days specified in the Action Details window of the Alerts form.
known as the response processor.

Apps

- **How will u register RDF file and run it? Tell the Sequence?**

  Steps a. Save the copy of ur reports in rdf file in ur local directory.
  b. Transfer or copy the rdf file to cus_top under reports directory through ftp.
  C. Then go concurrent program under executable menu where u define executable file and program name
  d. Then go to define the program name (which ur executable file name) and check the srs box and define the parameter and give the parameter name in token
  e. Attach the program(request to ur responsibility)
  d run the program and view the out put is srs through ur responsibility

.

Which flexfield qualifiers are mandatory?
Ans:- 'Balancing Segment' flexfield qualifier is mandatory.

What Difference Between report development and report customization.
Ans:- Report Development means - make the new report and attach to the oracle apps.
Report Customization means - customize the existing oracle apps report.
What is MULTI-ORG and what is structure of multi-org.
Ans:- Use a single installation of any oracle applications product to support any number of organizations. if those organizations use different set of books.
Support any number or legal entities with a single installation of oracle applications.
Secure access to data so that users can access only the information that is relevant to them.

Structure :- Business Unit
-HRMS(Employee)
-GL(Set of Books)(Currency, Calendar, Chart of Account)
|
Balancing Segment(You can do multiple balancing segment)
-Operating Units (Purchase, Selling, Fixed Asset, Payable,
Receivables)
-Inventory Organizations (Storing Items, Transaction Happening,
Ware Housing)
(Note:- Means if you maintaining GL(set of book id), If u have operating unit, if you have inventory then its called MULTI-ORG)
.**What are the default types of parameters. What is the use of each one of it.**

Ans:-******

Differnet type of execution methods in Conc.Progs. Explain Each Type.
Ans:- a.Oracle Reports- You can register your report as executable file type is oracle reports.
b. PL/SQL Package Procedure - You can register your PL/SQL Package Procedure as executable file type is oracle PL/SQL Package Procedure.
c. SQL Loader- You can register your SQL Loader SQL Loader is your executable file type.(for data loading)
d. SQL*Plus :- You can register your SQL script as SQL*Plus executable type.
e. Host Scripting:- You can write down Unix Host scripting and register here.
:$FLEX$. value_set_name
Retrieves a value (the hidden ID value, if a hidden ID value is defined) in a prior segment.
:$PROFILES$. profile_option
Retrieves the current value of a profile option. You must specify the option name of the profile option, such as GL_SET_OF_BKS_ID (which does not contain the Accounting Flexfield structure number).
Note that your profile option must be set wherever you use this value set (including the View Requests form if this value set is used as a report parameter and the user tries to view the status of the report after submission), or your user will see error messages.
:block.field
Gets the current value in a field. You must ensure that this value set is only used for forms that have the same block.field. For example, the following user exit on a Validate event obtains the Structure (NUM) of the key flexfield from a profile option:

What is the difference between ORG_ID and ORGANIZATION_ID in Multi Org? At where we can set ORG_ID and ORGANIZATION_ID level it comes in the Structure of the Multi Org.
Org Id will be set at operating unit level and Organization id will be set at inventory level. The structure of the Multi Org is
Business World (Set of Book id)-->Legal Entities-->operating Unit -->Inventory
For GL we have Set of Books id For the Remaining modules like AP,AR we have to set the Org Id at the operating unit level.

16. ORG_ID can be set at master level or transaction level.
a) we have to set Org_Id in master level and transaction level also.

What is the difference between a package procedure and procedure and difference between function & procedure
a) When we are working on modules it is good to write all the procedures in one package rather than scattered. The main difference between the function and procedure is normally a function will return a value and procedure wont
.
What is Chart of accounts?
Ans: A complete listing of the accounts to identify specific accounts to be increase or decrease.

What is the Relationship between Request Group, Data group, , Users and Menu.
Users ---- Responsibility---- Data group, Menu. Request Group

- Tables are need to be registered when used for ALERT, FORMS, AUDITING purpose (by using AD_DD package)
- When a request group is assigned to responsibility then its called REQUEST SECURITY GROUP.
- Following tables are used for users and responsibility info. FND_USER, FND_RESPONSIBILITY_VL (maid up of FND_RESPONSIBILITY_TL, RESPONSIBILITY).
- Type of concurrent managers – Internal, Standard, Specialized, Library based manager, conflict resolution manager.

Questions asked by Icelerate

- **How to attach reports in Oracle Applications ?**

Ans: The steps are as follows :

- u Design your report.
- u Generate the executable file of the report.
- u Move the executable as well as source file to the appropriate product's folder.
- u Register the report as concurrent executable.
- u Define the concurrent program for the executable registered.
- u Add the concurrent program to the request group of the responsibility.

- **What is the use of cursors in PL/SQL? What is REF Cursor?**

Ans.: The cursors are used to handle multiple row queries in PL/SQL. Oracle uses implicit cursors to handle all it's queries. Oracle uses unnamed memory spaces to store data used in implicit cursors, with REF cursors you can define a cursor variable, which will point to that memory space and can be used like pointers in our 3GLs.

- **What is record group?**

Ans: Record groups are used with LOVs to hold sql query for your list of values. The record group can contain static data as well it can access data from database tables thru sql queries.

- **What are autonomous transactions? Give a scenario where you have used autonomous transaction in your reports?**

Ans: An autonomous transaction is an independent transaction started by another transaction, the main transaction. Autonomous transactions let you suspend the main transaction, do SQL operations, commit or roll back those operations, then resume the main transaction.
Once started, an autonomous transaction is fully independent. It shares no locks, resources, or commit-dependencies with the main transaction. So, you can log events, increment retry counters, and so on, even if the main transaction rolls back.

More important, autonomous transactions help you build modular, reusable software components. For example, stored procedures can start and finish autonomous transactions on their own. A calling application need not know about a procedure's autonomous operations, and the procedure need not know about the application's transaction context. That makes autonomous transactions less error-prone than regular transactions and easier to use.

Furthermore, autonomous transactions have all the functionality of regular transactions. They allow parallel queries, distributed processing, and all the transaction control statements including SET TRANSACTION.

Scenario : You can use autonomous transaction in your report for writing error messages in your database tables.

- **What is the use of Temp tables in Interface programs ?**

Ans : Temporary tables are used in Interface programs to hold the intermediate data. The data is loaded into temporary tables first and then, after validating through the PL/SQL programs, the data is loaded into the interface tables.

- **What are the steps to register concurrent programs in Apps ?**

Ans : The steps to register concurrent programs in apps are as follows :

- u Register the program as concurrent executable.
- u Define the concurrent program for the executable registered.
- u Add the concurrent program to the request group of the responsibility

- **How to pass parameters to a report ? do you have to register them with AOL ?**

Ans: You can define parameters in the define concurrent program form. There is no need to register the parameters with AOL. But you may have to register the value sets for those parameters.

- **Do you have to register feeder programs of interface to AOL ?**

Ans : Yes ! you have to register the feeder programs as concurrent programs to Apps.

- **How to use Flexfields in reports ?**

Ans: There are two ways to use Flexfields in report. One way is to use the views (table name + '_KFV' or '_DFV') created by apps, and use the concatenated_segments column which holds the concatenated segments of the key or descriptive flexfields.
Or the other way is to use the FND user exits provided by oracle applications.

Questions asked by HCL

What is MRC and what are it's use?
Ans: The Multi Reporting Currency Feature allows you to report and maintain records at the transaction level in more than one functional currency. You can do by defining one or more set of books in addition to primary set of books.

What are ad-hoc reports ?
Ans: Ad-hoc Report is made to meet one-time reporting needs. Concerned with or formed for a particular purpose. For example, ad hoc tax codes or an ad hoc database query
What is FSG and what is it's use ?
Ans: FSG is a powerful and flexible tool you can use to build your own custom reports without programming. FSG is only available with GL.

Questions asked by RELQ

Useful website

http://teachmeoracle.com/

http://download-east.oracle.com/docs/cd/A58617_01/server.804/a58227/toc.htm

asktom.oracle.com

http://getappstraining.blogspot.com/

$FLEX$: Enables to match the prior segment with either value set name or segment name.
Let v2 be the value set definition of 2nd parameter and v1 be the value set definition for the first parameter then
In the value set definition of v2 = value $FLEX$.v1

How to find the custom directory in front end.

From Application Developer responsibility, navigate to Application –> Register. Query for the custom application name. The value in the field Basepath, is the OS system variable that stores the actual directory info.

What are APIs

API stands for **application program interface**. Oracle has its own api facility to transfer data from a external source to oracle base table so api is an standard oracle tool to transfer data to oracle database. It is a very simple method and a person with no technical background also can use api. Example-uploading journal and budget data through API.

Difference between Lookups and Value Sets

Difference 1
Value sets can be attached to parameters of a concurrent program, whereas Lookups can't.

Difference 2
Certain types of Lookups are maintainable by the users too, for example HR Users will maintain "Ethnic Minority" lookups. Value Sets are almost never maintained by end users, with the exception of GL Flexfield codes. Value sets are usually maintained by System Administrators.

Difference 3
Value sets can contain values that are a result of an SQL Statement.
Hence it is possible to make Value Set list of values dynamic.
On the contrary, Lookup Codes are Static list of values.

What are the ingredients for a concurrent program?
Answer: A concurrent executable and a program attached to that executable.

Will executable for a pl/sql concurrent program be a database stored procedure?
Answer: Yes, but in addition to above you can also register a procedure within a package as an executable. However, you can't make a Function to become the executable for a stored procedure.

Does this stored procedure need to have some specific parameters, in order to become an executable of a concurrent program?
Answer: Yes, such procedure must have at least two parameters
(errbuff out VARCHAR2, retcode out NUMBER)

Can we add additional parameters to such pl/sql procedures that happen to be Conc Prog Executables?
Answer: Sure you can, but those parameters must be defined after the first two parameters. Effectively I mean first two parameters must always be errbuff and retcode. The sequence of the remaining parameters must match with the sequence in which parameters are registered in define concurrent program-parameters window.

Can those parameters be validated or will these parameters be free text?
Answer: These parameters can be attached to a value set, hence this will avoid users passing free text values.

What are the possible things that a concurrent pl/sql program can do?
Firstly your stored procedure would have been created in apps. This concurrent program will connect to "apps schema" from where access to all the tabes across every module will be available.
You can do the following:-
1. Insert records in tables(usually interface or temp tables)
2. Update and delete records
3. Initiate workflows
4. Display messages in the output file or the log file of the concurrent program.
5. Make this concurrent program complete with status Error or Warning or Normal.

Please give me an example of a pl/sql concurrent program in Oracle apps in real life?
Answer: Lets say you have an external application which is integrated with apps. Assume that it is Siebel application where the new customer records are created. Siebel is assumingly hosted on a Oracle database from where it has database links to your Oracle apps database.
In this case, siebel inserts sales order records into your custom staging tables.
You can then develop a concurrent process which will do the following:--------
Loop through the records in that staging table
Check if the customer already exists in Oracle AR TCA
If customer already exists, thencall the api to update existing customer
If this is a new customer, then update existing TCA Customer/Party record

Question: Ok, how do I do the above?
Answer: Find the steps below for doing so

Step 1
Connect xxschema/password
Create table xx_stage_siebel_customers (customer_Id integer, customer name varchar2(400));
Grant all on xx_stage_siebel_customers to apps ;

Step 2
Connect apps/apps_password
Create or replace synonym xx_stage_siebel_customers for xxschema.xx_stage_siebel_customers ;

Step 3 (again in apps schema)
Create or replace procedure xx_synch_siebel_cust (errbuff out varchar2, retcode out varchar2) is
n_ctr INTEGER := 0 ;
Begin
for p_rec in (select * from xx_synch_siebel_cust) LOOP
Select count(*) into n_ctr from hz_parties where party_number = p_rec.customer_number;
If n_ctr=0 then
Hz_party_create(pass appropriate parameters here).
Else
Hz_party_update(pass appropriate parameters here);
End if;
END LOOP ;
delete from xx_synch_siebel_cust ;
End xx_synch_siebel_cust

Step 4
Create concurrent program executable (for example of screenshot, visit link)

Step 5
Create concurrent program for that executable

Step 6
Add this concurrent program to request group

What are Profile Options in Oracle Apps ?

Profile Options provide flexibility to Oracle Apps. They are a key component of Oracle Applications, hence these much be understood properly. I will be taking multiple examples here to explain what profile options mean. I will also try to explain by stepping into Oracle shoes "How will you design a program that is flexible", by using Profile Options.

Following that, if you still have questions regarding profile options, then leave a comment and I promise to respond. For the learners of Oracle Apps, understanding profile options is mandatory.

What is profile option?
The profile option acts like a Global Variable in Oracle.

Why does Oracle provide profile options?
These are provided to keep the application flexible. The business rules in various countries and various companies can be different. Hence the profile options are delivered by Oracle in such a manner to avoid hard-coding of logic, and to let the implementation team at site decide the values of those variables.

For screenshots of below listed examples in this article, please click this link

Enough definitions, give me some scenarios where profile options are used by Oracle....
1. There are profile options which can turn the debugging on, to generate debug messages. Say one of 1000 users reports a problem, and hence you wish to enable debugging against just that specific user. In this case you can “Turn On” the debugging profile option "again that specific user".
2. There are profile options that control which user can give discount to their customers at the time of data entry. You can se profile option "Discount Allowed" to a value of either Yes or No against each Order Entry user.
3. Lets assume an Organization has department D1 and D2. Managers of both the Departments have "HRMS Employee View" responsibility. But you do not want Manager of D2 to be able to see the list of Employees in Organization D1. Hence you can set a profile option against the username of each of these users. The value assigned to such profile option will be "Name of the Organization" for which they can see the employees. Of course, the SQL in screen that displays list of employees will filter off the data based on “logged in users profile option value”.

Let’s take an example. Let’s assume you are a developer in Oracle Corporation building a screen in ERP. Let us further assume that you are developing an Order Entry screen.
Assume that business requirements for your development work is:-
1. Screen should be flexible to ensure that different users of the screen can give different levels of discounts. For example, a clerk Order Entry User can give no more than 5% discount. But Sales Manager can enter an Order with 15% discount.
2. There should not be any hard-coding regarding the maximum permissible discount.
3. In the screen there will be a discount field.
4. When the discount value is entered in discount field, an error will be raised if user violates the maximum permissible discount.

Here is how Oracle will code this screen
1. They will define a profile option named "OEPASSI Maximum Discount Allowed".
2. The short name of this profile option is "OEPASSI_MAX_DISCOUNT"
2. In the when-validate-item of the discount field(assuming Oracle Forms), following code will be written

```
IF :oe_line_block.discount_value > fnd_profile.value('OEPASSI_MAX_DISCOUNT')
THEN
message(
'You can’t give discount more than '
|| fnd_profile.value('OEPASSI_MAX_DISCOUNT') || '%' ) ;
raise form_trigger_failure ;-- I mean raise error after showing message
END IF ;
```

Here is how, the client implementing Oracle Order Entry will configure their system.
1. Navigate to System administration and click on system profile menu.
2. For Clerk User(JOHN), set value of profile "OEPASSI Maximum Discount Allowed" to 5
For Sales Manager User(SMITH), set value of profile "OEPASSI Maximum Discount Allowed" to 15

Question: This sounds good, but what if you have 500 Order Entry Clerks and 100 Order Entry Sales Managers? Do we have to assign profile option values to each 600 users?
Answer : Well, in this case, each Clerk will be assigned Responsibility named say “XX Order Entry Clerk Responsibility”

Each Sales Manager will be assigned Responsibility named say “XX Order Entry Sales Manager Responsibility”
In this case, you can assign a profile option value to both these responsibilities.
“XX Order Entry Clerk Responsibility” will have a value 5% assigned against it. However, “XX Order Entry Sales Manager Responsibility” will have a profile option value of 15% assigned.
In the when-validate-item of the discount field, following code will then be written

```
IF :oe_line_block.discount_value > fnd_profile.value('OEPASSI_MAX_DISCOUNT')
THEN
message(
'You can’t give discount more than '
|| fnd_profile.value('OEPASSI_MAX_DISCOUNT') || '%' ) ;
raise form_trigger_failure ;-- I mean raise error after showing message
END IF ;
```

Please note that our coding style does not change even though the profile option is now being assigned against responsibility. The reason is that API fnd_profile.value will follow logic similar to below.
Does Profile option value exist against User?
--Yes: Use the profile option value defined against the user.
--No: Does Profile option value exist against Responsibility
-----Yes: Use the profile option value defined against the current responsibility in which user has logged into.
-----No: Use the profile option value defined against Site level.

For screenshots of examples in this article, please refer this link

Q. Whats main concurrent Manager types.
ICM - Internal Concurrent Manager which manages concurrent Managers
Standard Managers - Which Manage processesing of requests.
CRM - Conflict Resolution Managers , resolve conflicts in case of incompatibility.

Q. Where is Concurrent Manager log file location.
By default standard location is $APPLCSF/$APPLLOG , in some cases it can go to $FND_TOP/log as well.

How will you migrate Oracle General Ledger Currencies and Sets of Books Definitions fromone environment to another without reKeying? Will you use FNDLOAD?
Answer: FNDLOAD can not be used in the scenario. You can use migrator available in "Oracle iSetup" Responsibility

Which responsibility do you need to extract Self Service Personalizations?
Answer: Functional Administrator

Question: You have just created two concurrent programs namely "XX PO Prog1" & "XX PO Prog2". Now you wish to create a menu for Concurrent Request submission such that only these two Concurrent Programs are visible from that Run Request menu. Please explain the steps to implement this?
Answer:
a) Define a request group, lets say with name "XX_PO_PROGS"
b) Add these two concurrent programs to the request group "XX_PO_PROGS"
c) Define a new Form Function that is attached to Form "Run Reports"
d) In the parameter field of Form Function screen, enter
REQUEST_GROUP_CODE="XX_PO_PROGS" REQUEST_GROUP_APPL_SHORT_NAME="XXPO"
TITLE="XXPO:XX_PO_PROGS"
e) Attach this form function to the desired menu.

Does oracle support partitioning of tables in Oracle Apps?
Answer: Yes, Oracle does support partitioning of tables in Oracle Applications. There are several implementations that partition on GL_BALANCES. However your client must buy licenses to if they desire to partition tables. To avoid the cost of licensing you may suggest the clients may decide to permanently close their older GL Periods, such that historical records can be archived.
Note: Before running the archival process the second time, you must clear down the archive table GL_ARCHIVE_BALANCES (don't forget to export archive data to a tape).

Question: What will be your partitioning strategy on GL_BALANCES? Your views please?
Answer: This really depends upon how many periods are regularly reported upon, how many periods are left open etc. You can then decide to partition on period_name, or period ranges, or on the status of the GL Period.

Question: Does Oracle support running of gather stats on SYS schema in Oracle Apps?
Answer: If your Oracle Applications instance is on 10g, then you can decide to run stats for SYS schema.
 This can be done by exec dbms_stats.gather_schema_stats('SYS');
Alternately using command dbms_stats.gather_schema_stats('SYS',cascade=>TRUE,degree=>20);
I will prefer the former with default values.
If you wish to delete the stats for SYS use exec dbms_stats.delete_schema_stats('SYS');
You can schedule a dbms_job for running stats for SYS schema.

Question: Can you use concurrent program "Gather Schema Statistics" to gather stats on sys schema in oracle apps?
Answer: No, "Gather Schema Statistics" has no parameters for SYS schema. Please use dbms_job.

Question: Which table is used to provide drill down from Oracle GL into sub-ledger?
Answer: GL_IMPORT_REFERENCES

Question: How to debug a document manager in Oracle Apps?
Answer: Document manger runs within the concurrent manager in Oracle Applications. When an application uses a Document Manager, it sends a pipe signal which is picked up by the document manager.
<u>There are two mechanisms by which to trace the document manager</u>
1. Set the debugging on by using profile option
   STEP 1. Set profile option "Concurrent:Debug Flags" to TCTM1
   This profile should only generate debugs when set at Site level(I think, as I have only tried site), because Document Manager runs   in a different session.
   STEP 2. Bounce the Document Managers
   STEP 3. Retry the Workflow to generate debugs.
   STEP 4. Reset profile option "Concurrent:Debug Flags" to blank
   STEP 5. have a look at debug information in table fnd_concurrent_debug_info

2. Enable tracing for the document managers
This can be done by setting profile option "Initialization SQL Statement – Custom" against your username before reproducing the issue. The value of this profile will be set so as to enable trace using event 10046, level 12.

How to make concurrent program end with warning?
Answer: If the concurrent program is of type PL/SQL, you can assign a value of 1 to the "retcode" OUT Parameter.
For a Java Concurrent program, use the code similar to below
ReqCompletion lRC;
//get handle on request completion object for reporting status

lRC = pCpContext.getReqCompletion();
lRC.setCompletion(ReqCompletion.WARNING, "WARNING");

Can you do fnd_request.submit_request from SQL Plus in Oracle?
Answer: You will need to initialize the global variables first using fnd_global.initialize

```
DECLARE
 v_session_id INTEGER := userenv('sessionid') ;
BEGIN
fnd_global.initialize
(
SESSION_ID  => v_session_id
,USER_ID=> <your user id from fnd_user.user_id>
,RESP_ID  => <You may use Examine from the screen PROFILE/RESP_ID>
,RESP_APPL_ID  => <You may use Examine from the screen PROFILE/RESP_APPL_ID>
,SECURITY_GROUP_ID => 0
,SITE_ID => NULL
,LOGIN_ID => 3115003--Any number here
,CONC_LOGIN_ID => NULL
,PROG_APPL_ID => NULL
,CONC_PROGRAM_ID => NULL
,CONC_REQUEST_ID => NULl
,CONC_PRIORITY_REQUEST => NULL
) ;
commit ;
END ;
/
```

Optionally you may use fnd_global.apps_initialize, which internally calls fnd_global.initialize

```
 fnd_global.apps_initialize(user_id => :user_id,
resp_id => :resp_id,
 resp_appl_id => :resp_appl_id,
 security_group_id => :security_group_id,
 server_id => :server_id);
```

By doing the above, your global variables upon which Concurrent Managers depend upon will be populated. This will be equivalent to logging into Oracle Apps and submitting the concurrent request from a responsibility.

What is Partial Backup ?
A Partial Backup is any operating system backup short of a full backup, taken while the database is open or shut down.

What is Mirrored on-line Redo Log ?

A mirrored on-line redo log consists of copies of on-line redo log files physically located on separate disks, changes made to one member of the group are made to all members.

What is Full Backup ?
A full backup is an operating system backup of all data files, on-line redo log files and control file that constitute ORACLE database and the parameter.
What is the use of transactional triggers?
Using transactional triggers we can control or modify the default functionality of the oracle forms.

. What are the names of the parameters u pass to the Procedure which u register in the apps?

u 1) retcode out varchar2

2) errbuf out varchar2

Types of PO
Table names, column names, mandatory columns (why)
FND packages?
How do you set a printer a value thru FND package?
What is the transaction type?
Shipment, pricing tables?
Concept of Drop shipment?
How does the data flow from OM to Inv?

Steps General Ledger

16) **Creat Validation for flexfield**
17) **creat keys for felxfield**
18) **give values for esch keys(segments)**
19) **creat Calander type**
20) **define a/cing periods**
21) **Restrict Transaction days**
22) **creat set of books**
23) **Accounting for Multiple Compnies**
24) **open/close period of a/cs**
25) **creat Responsibility**
26) **creat user**
27) **creat profile**
28) **Assign Pf to user**
29) **creat a Journal Batch**
30) **posting a JE Batch**
31) **Creat a statistical Entry**
32) **Creat Recurring JE 1. std 2.Skeleton 3.Formula**
33) **Creat Unbalanced JE**
34) **Creat Monetary + Statistical JE**
35) **Creat FC ,JE**
36) **Creat Intercompany A/CS**
37) **Define Currency Rates 1. Period 2.Revalvation 3. Daily**
38) **creat reversal category**
39) **Encumbrance Type**
40) **GL AD.**
41) **massallocation**
42) **budget**
43) **creat budget**
44) **creat budget organization**
45) **creat consolidation**
46) **transfer Data**
47) **Various Report**
48) **standard a/cs**
49) **Financial**
50) **Assign Tax Structures.**

Account Payble

- **Control Payble Period – a/cs**
- **Payment Terms**
- **Distribution Sets**
- **types of Matching**
- **Tax Code**
- **Tax group**
- **Options Pa yables**
- **options financials**

- **Supplier Entry**
- **Invoice Entry**
- **Distribution**
- **Action**
- **Hold**
- **Exps Reports**
- **Request**
- **Payble Invoice Import**
- **creation of Batches**
- **Run a report for a/cing Entry**
- **Payment bank**
- **Prepayment Amount**
- **Spl. Calander**
- **Recurring Invoice to be Generated**
- **Paybles**
- **payment Single**
- **payment in Batches**
- **Freight /other changes**
- **Stop Paymnet**
- **Supplier merge**
- **A/c Process**
- **pass Entry to GL**
- **EXCEPTIONAL REPORT**

. **Can u create procedure or function without declaring it in Package specs?**
Ans: YES, It is called private procedure.

what is private function and public functions in package?
Ans: If the function is declared in Package Specification then it is called Public Function.
Public function can be called outside of Package.
If the function is not declared in Package Specification then it is called Private Function.
Private function can not be called outside of Package.

8. how do u call private functions in package?
Ans:

```
pack spcs p1...
        func f1(); -- Public function
        func f2(); -- Public function
end;

pack body p1...
        func f1(){};  -- public
        func f2(){};  -- public
        func f3(){};  -- Private
        func f4(){};  -- Private
end;
```

to call private call it in public function and public fun can be called from outside.

create a syquence, open a new session and execute first statement as select
sequence.curval from dual; what will happene?
Ans: It will give an error. First time we have to fire next val & then only we can use curval.
11. I have t1 table in scott .. and same i have scott1 schema with same name... i grant select on scott1.t1 to scott, now i create a synonym for scott1.t1, what happenes when it is created. will it give runtime error or error while creating synonym?
Ans: This will give an error, Same name can not be used

tell me diff between .. 7.x, 8, 8i, 9i (undo tablespace)

Ans: **9i New Features--------------**

1. Rollback Segment is being replaced by Undo Tablespace. (We can use either)
2. Flashback Query - It is used to retrack wrongly committed transaction
3. Online Table Creation (e.g. Create table as select * from .... will generate updat
4. List Partition - Table can be partitioned on a list
5. Buffer catche size is now dynamic (upto 4 different sizes can be specified for buffers)

. **types of tuning?**

Ans: Application Tuning, Database Tuning, Memory Tuning, O/S Tuning

.

What is the difference between po_headers_all and po_headers.

Ans: po_headers_all --TABLE
po_headers --VIEW

IF I WANT TO COPY BY ONE BUSINESS AREA TO ANOTER IS IT POSSIBLE? HOW?

Yes ,go to Tool menu then Select manage Folder There You can Copy one **BA** to another **BA**

. **WHAT IS THE EXECUTABLE TYPE AND EXTENSION FOR REGISTERING REPORT IN APPS ?** (Exe Type is Oracle Reports and No Need to mention extention When You Register Your Report in Apps)

23. CAN I USED INDEPENDENT VALUE SET IN INDEPENDENT VALUE SET?
Not Possible.
Can use INDEPENDENT VALUE SET IN DEPENDENT VALUE SET

. **IF I WANT TO USED DISTINCT VALUES FROM MY COLUMN BY USING TABLE VALUE SET HOW I CAN DO THAT?** (Writing Select Statement in Table Text box or Using ROWID in WHERE Clause)

....?

23. Define security and user and role...?

How do I identify the package name and version of the API?

```
>SELECT text
FROM all_source
WHERE name like '%your_api_name%'
--(example: '%HR_POSITION_API%')
AND text like '%Header%';
```

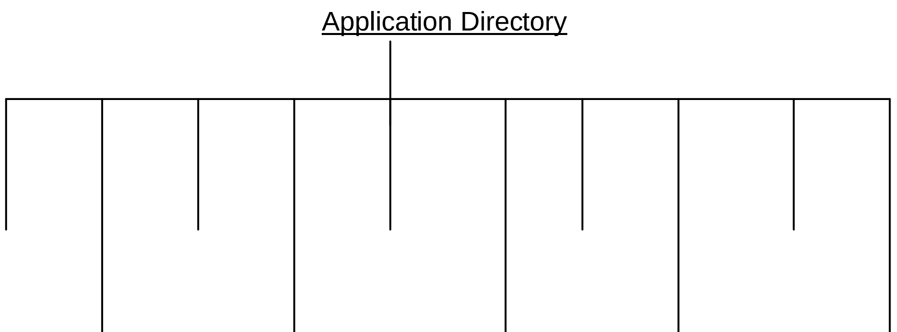
When u will need to setup multi-org setup?

Answer

No, you do not have to setup multi-org to have multiple organizations.
Multi-org is only required if you intend to have multiple sets of books.

What is the Directory structure on the server ?

SERVER – SIDE DIRECTORY TREE TO STORE FILES



Bin SQL log srw mesg

Lib rpt forms (Lang) out plsql

u Bin: Contains executable code of your concurrent programs written in a programming language such as C, Pro*C, Fortran, or an operating system script.
u Lib: Contains compiled object code of your concurrent programs.
u SQL: Contains concurrent programs written in SQL*Plus and PL/SQL scripts.
u Rpt: Contains concurrent programs written with SQL*Reports.
u Log: Contains log files from concurrent programs.
u Forms/(Language): Each language has a subdirectory (such as US). The language subdirectory holds the forms .fmx files.
u Srw: Contains concurrent programs written with Oracle Reports.
u Out: Contains output files from concurrent program.
u Mesg: Holds your application message files for Message dictionary.
u PLSQL: Contains PL/SQL libraries used with Oracle reports.

What are the salient features of AOL and Sysadmin functions?

APPLICATION OBJECT LIBRARY

As evident from the name AOL i.e. Application Object Library is the Library that contains all the Objects of an Application. For Oracle Apps. To recognize any object, such object must be registered with this Library.

Salient Features –

u Registering Tables with Oracle Apps.

u Registering Forms with Oracle Apps.

u Registering Concurrent Programs with Oracle Apps.

u Building Menus.

u Building Flexfields.

u Enabling Zoom.

u Building Message Dictionary.

SYSTEM ADMINISTRATION

- u Manage Oracle Applications security.
- u Manage Concurrent programs & Reports.
- u Manage Concurrent processing.
- u Manage Printers.
- u Manage Profile Options.
- u Manage Document Sequences.

How do you register tables in Apps? What is the PL/SQL package used for registering?

CONCURRENT PROCESSING IN ORACLE APPS.

Definitions

What is a Concurrent Program?

An instance of an execution file, along with parameter definitions and incompatibilities. Several concurrent programs may use the same execution file to perform their specific tasks, each having different parameter defaults and incompatibilities.

An executable program or report in Oracle, which is executed in the background, and allows user to continue with other work while the job is being executed.

What is a Concurrent Program Executable?

An executable file that performs a specific task. The file may be a program written in a standard language, a reporting tool or an operating system language.

What is a Concurrent Request?**A request to run a concurrent program as a concurrent process.**

What is a Concurrent Process?**An instance of a running concurrent program that runs simultaneously with other concurrent processes.**

What is a Concurrent Manager?**A program that processes user's requests and runs concurrent programs. System Administrators define concurrent managers to run different kinds of requests.**

What is a Concurrent Queue?**List of concurrent requests awaiting processing by a concurrent manager.**

What is a Spawned Concurrent program?**A concurrent program that runs in a separate process than that of the concurrent manager that starts it.**

PL/SQL stored procedures run in the same process as the concurrent manager; use them when spawned concurrent programs are not feasible.

What is the difference between Request group and request set?L

Reports and concurrent programs can be assembled into request groups and request sets.

· A request group is a collection of reports or concurrent programs. A System Administrator defines report groups in order to control user access to reports and concurrent programs. Only a System Administrator can create a request group.

· Request sets define run and print options, and possibly, parameter values, for a collection of reports or concurrent program. End users and System Administrators can define request sets. A System Administrator has request set privileges beyond those of an end user.

Standard Request Submission and Request Groups

Standard Request Submission is an Oracle Applications feature that allows you to select and run all your reports and other concurrent programs from a single, standard form. The standard submission form is called Submit Requests, although it can be customized to display a different title.

· The reports and concurrent programs that may be selected from the Submit Requests form belong to a request security group, which is a request group assigned to a responsibility.

· The reports and concurrent programs that may be selected from a customized Submit Requests form belong to a request group that uses a code.

In summary, request groups can be used to control access to reports and concurrent programs in two ways; according to a user's responsibility, or according to a customized standard submission (Run Requests) form.

Once the report is defined what are the next steps involved in running it through Apps?

1. Define Executables….

2. Define Concurrent program and its parameters

3. Put the <FILENAME.rdf> on the server directory for the application of executables /appl/databse/application/1.0/reports

DEFINE CONCURRENT PROGRAM EXECUTABLE

DEFINE AND MODIFY YOUR CONCURRENT PROGRAMS

Prerequisites

· Build the execution file for your concurrent program.

· Use the Concurrent Program Executables window to define a concurrent program executable for your operating system program.

Concurrent Programs Block

The combination of application name plus program name uniquely identifies your concurrent program.

Program

You see this longer, more descriptive name when you view your requests in the Concurrent Requests window. If this concurrent program runs through Standard Request Submission, you see this name in the Submit Requests window when you run this program.

Short Name

Enter a brief name that Oracle Applications can use to associate your concurrent program with a concurrent program executable.

Application

The program's application determines what ORACLE username your program runs in and where to place the log and output files.

Enabled

Indicate whether users should be able to submit requests to run this program and the concurrent managers should be able to run your program.

Disabled programs do not show up in users' lists, and do not appear in any concurrent manager queues. You cannot delete a concurrent program because its information helps to provide an audit trail.

Name

Select the concurrent program executable that can run your program. You define the executable using the Concurrent Program Executables window. You can define multiple concurrent programs using the same concurrent program executable. See: Concurrent Program Executables.

Options

If your concurrent program is a SQL*Report program, you can enter execution options or switches for the Report Formatter (RPF). To use several options at the same time, do not repeat the '-'. For example, enter -FI to choose the form feed and initial page eject options. You can enter these options in either uppercase or lowercase and in any order. Valid options for SQL*Report are:

-F	Form Feed Choose this option to send a form feed character before printing a page.
-I	Initial Page Eject Choose this option to eject a page before printing.

-P:n:m	Page Choose this option to print from page n topage m.
-R	Reverse Underlining Order Choose this option to print the underline character on devices that do not allow overstriking.
-U	Upper Case Choose this option to print all of your report in uppercase.

If you define a concurrent program with the bitmapped version of Oracle Reports, you must set the execution option field to "VERSION=2.0b".

You can also pass ORIENTATION=<value> and PAGESIZE=<width>x<height> parameters to your bitmapped Oracle Reports program. Units of width and height are set in your Oracle Reports program.

There should be no spaces before or after the execution options values. The parameters should be separated by only a single space.

Method

The execution method your concurrent program uses appears here.

Valid values are:

Spawned	Your concurrent program is a stand-alone program in C or Pro*C.
Host	Your concurrent program is written in a script for your operating system.
Immediate	Your concurrent program is a subroutine written in C or Pro*C. Immediate programs are linked in with your concurrent manage and must be included in the manager's program library.
Oracle Reports	Your concurrent program is an Oracle Reports script.
PL/SQL Stored Procedure	Your concurrent program is a stored procedure written in PL/SQL.
SQL*Loader	Your concurrent program is a SQL*Loader program.
SQL*Plus	Your concurrent program is a SQL*Plus or PL/SQL script.

.SQL*Report   Your concurrent program is a SQL*Report program. You can enter switch options, such as the form feed option, in the Execution Options field.

Priority

You can assign this program its own priority. The concurrent managers process requests for this program at the priority you assign here.

If you do not assign a priority, the user's profile option Concurrent:Priority sets the request's priority at submission time.

Request

Type

If you want to associate your program with a predefined request type, enter the name of the request type here. The request type can limit which concurrent managers can run your concurrent program.

You can define a concurrent manager to run only certain types of concurrent requests.

Use in SRS

Check this box to indicate that users can submit a request to run this program from a Standard Request Submission window.

If you check this box, you must register your program parameters, if any, in the Parameters window accessed from the button at the bottom of this window.

Allow Disabled Values

If you check the Use in SRS box, you can also check this box to allow a user to enter disabled or outdated values as parameter values.

Many value sets use special table columns that indicate whether a particular value is enabled (using ENABLED_FLAG, START_DATE_ACTIVE, and END_DATE_ACTIVE columns). These value sets normally allow you to query disabled or outdated values but not enter them in new data. For Standard Request Submission, this means that a user would not normally be allowed to enter disabled values as report parameter values when submitting a report, even if the report is a query-only type report.

Run Alone

Indicate whether your program should run alone relative to all other programs in the same logical database. If the execution of your program interferes with the execution of all other programs in the same logical database (in other words, if your program is incompatible with all programs in its logical database, including itself), it should run alone.

You can enter any specific incompatible programs in the Incompatible Programs windows.

Save

Indicate whether to automatically save the output from this program to an operating system file when it is run. This value becomes the default for all requests submitted for this program. The output of programs with Save set to No is deleted after printing.

If this is a Standard Request Submission program, users can override this value from the Submit Requests window.

Print

If you enter No, your concurrent program's output is never sent to the printer.

Columns / Rows

Enter the minimum column and row length for this program's report output. Oracle Applications uses this information to determine which print styles can accommodate your report.

Style

The print style you select depends on your system and printer setup. Print styles include:

- **132 columns and 66 lines (Landscape)**
- **180 columns and 66 lines (Landwide)**
- **80 columns and 66 lines (Portrait)**
- **132 columns and 62 lines (A4)**

Your list is limited to those styles that meet your program's columns and row length requirements.

Style Required

If your program requires a specific print style (for example, a checkwriting report), use this check box to enforce that print style.

Printer

If you want to restrict your program's output to a single printer, enter the name of the printer to which you want to send your output. If your program has minimum or maximum columns or rows defined, your list of values is limited to those printers that can support your program's requirements.

Users cannot override your choice of printer from the Submit Requests or Concurrent Requests windows.

Copy to...	**Choose this button to create another concurrent program using the same executable, request and report information. You can elect to copy the incompatibility and parameter details as well.**
Incompatibilities	**Choose this button to open the Incompatible Programs window.**
Parameters	**Choose this button to open the Concurrent Program Parameters window.**

CONCURRENT PROGRAM PARAMETERS

Enter and update the program parameters that you wish to pass to the program executable. Program parameters defined here should match the variables in your execution file.

Sequence

Choose the sequence numbers that specify the order in which your program receives parameter values from the concurrent manager.

The default value for this field is the next whole sequence number. If you frequently change sequence numbers or frequently insert and delete parameters, carefully check the default value to ensure it matches your expectations.

Enabled

Disabled parameters do not display at request submission time and are not passed to your execution file.

Validation Information

Value Set

Enter the name of the value set you want your parameter to use for validation. You can only select from independent, table, and non-validated value sets.

The maximum size of your value set is 240 characters.

Default Type

If you want to set a default value for this parameter, identify the type of value you need.

Valid types include:

Constant	**The default value can be any literal value.**
Current Date	**The default value is the current date in the format DD-MON-YY or DD-MON-YYYY, depending on the length of the segment.**

Maximum Size	Date Format
9	DD-MON-YY
11	DD-MON-YYYY

Current Time	**The default value is the current time or the current date and time, depending on the length of the segment.**

Maximum Size	Time Format
5	HH24:MI
8	HH24:MI:SS
15	DD-MON-YY HH24:MI
17	DD-MON-YYYY HH24:MI
18	DD-MON-YY HH24:MI:SS
20	DD-MON-YYYY HH24:MI:SS

Profile	**The default value is the current value in the user profile option defined in the Default Value field. Use the profile option name, not the end-user name. You do not need to include $PROFILE$.**
SQL Statement	**The default value is determined by the SQL statement you defined in the Default Value field.**
Segment	**The default value is the value entered in a prior segment of the same parameter window.**

If you choose Current Date or Current Time, you skip the next field.

Default Value

You can enter a default value for the parameter. This default value for your parameter automatically appears when you enter your parameter window. You determine whether the default value is a constant or a context-dependent value by choosing the default type.

Your default value should be a valid value for your value set. Otherwise you see an error message when you enter your parameter window on the Run Request window and your default value does not appear.

Required

If the program executable file requires an argument, you should require it for your concurrent program.

Enable Security

If the value set for this parameter does not allow security rules, then this field is display only. Otherwise you can elect to apply any security rules defined for this value set to affect your parameter list.

Range

Choose either Low or High if you want to validate your parameter value against the value of another parameter in this structure. Parameters with a range of Low must appear before parameters with a range of High (the low parameter must have a lower number than the high parameter). For example, if you plan two parameters named "Start Date" and "End Date," you may want to force users to enter an end date later than the start date. You could assign "Start Date" a range of Low and "End Date" a range of High. In this example, the parameter you name "Start Date" must appear before the parameter you name "End Date."

If you choose Low for one parameter, you must also choose High for another parameter in that structure (and vice versa). Otherwise you cannot commit your changes.

Window Information

Display

Indicate whether to display this parameter in the Parameters window when a user submits a request to run the program from the Submit Requests window.

You should provide a default type and value for any non-displayed parameter.

Display Size

Enter the field length in characters for this parameter. The user sees and fills in the field in the Parameters window of the Submit Requests window.

You should ensure that the total of the value set maximum sizes (not the display sizes) for all of your parameters, plus the number of separators you need (number of parameters minus one), does not add up to more than 240. If your program values' concatenated length exceeds 240, you may experience truncation of your data in some forms.

Description Size

Enter the display length in characters for the parameter value description. Your window may show fewer characters of your description than you specify here if there is not enough room (determined by the sum of your longest prompt plus your display size for this parameter plus seven). However, your window does not display more characters of the description than you specify here.

Prompt

A user sees the prompt instead of the parameter name in the Parameters window of the Submit Requests window.

Concatenated Description Size

Enter the display length in characters for the parameter value description. The user sees the parameter value in the Parameter Description field of the Submit Requests and View Requests forms. The Parameter Description field concatenates all the parameter values for the concurrent program.

Suggestion: We recommend that you set the Concatenated Description Size for each of your parameters so that the total Concatenated Description Size for your program is 80 or less, since most video screens are 80 characters wide.

Token

For a parameter in an Oracle Reports program, the keyword or parameter appears here. The value is case insensitive. For other types of programs, you can skip this field.

What are the types of profile options? User and System….

PROFILES

A user profile is a collection of changeable options that affect the way your applications run. Oracle Applications establishes a value for each option in a user's profile when the user logs on or changes responsibility. Oracle Applications provides these options so that you can alter the behavior of your applications to suit your own preferences.

Oracle Applications uses a set of user profile options that are common to all the application products. In addition, each application product has its own unique set of user profile options. The reference guide for your application contains a complete list and a description of each user profile option that is specific to your application.

User profile options can be set at one or more of four levels: Site, Application, Responsibility, and User. Your system administrator can set default option values at any of these levels.

SYSTEM PROFILE VALUES

Use this window to view and set profile option values.

You can view and set profile options at the site, application, responsibility, and user levels. Your settings affect users as soon as they sign on or change responsibility. See: Overview of Setting User Profiles.

Profile Values Block

Set values for profile options at one or more levels. Each value overrides those set to its left. For example, a User Value setting overrides a Responsibility Value setting, which overrides an Application Value setting, which overrides a Site Value setting.

Profile

This field displays the name of a profile option.

Site

This field displays the current value, if set, for all users at the installation site.

Application

This field displays the current value, if set, for all users working under responsibilities owned by the application identified in the Find Profile Values block.

Responsibility

This field displays the current value, if set, for all users working under the responsibility identified in the Find Profile Values block.

User

This field displays the current value, if set, for the application user identified in the Find Profile Values block.

Suggestion: You should set site-level default values for any required options after installation of an application. If you do not assign a particular profile option at any of the four levels, that option does not have a default value and may cause errors when you use forms, run reports, or run concurrent requests.

17) different types of payment methods…
Payment by check ,payment by cash

-) .

u **Data Migration**: It is the just the process of transfer of data from 1 Environment to other.
u It may be transfer of procedure
or package or function from 1 database to another or transfer of objects from Repository to Database.

***Data Conversion**: It is the Process of Converting Existing data of a Table by mapping that table columns with 1 or more than 1 database columnS

2. What is the function of SQL*Net ?
Ans) If the user and server processes are on different computers of a network or if the user processes connect to shared server processes through dispatcher processes, the user process and server process communicate using SQL*Net. Dispatchers are optional background processes, present only when a multi–threaded server configuration is used. SQL*Net is Oracle's interface to standard communications protocols that allows for the proper transmission of data between computers. See "Oracle and SQL*Net" on page 1–48.

.If I have one business group nd 4 operating units....how can deny access of one unit to others ?
(N) profile>System-->MO Operating Unit-
This can also be done from Inv-->set up-->orgn-->orgn. Access

Key Flexfields Basics

Question: Key Flexfields help us capture additional fields, and so does descriptive flexfield too? What is the deal here?
Answer: Ok, let's assume for a minute that there is no such thing as a key flexfield. All we have is a descriptive flex (lets assume).
Requirement is this:-
Your client wants to capture values in following additional fields for a purchase order transaction and invoices...
Company name: GM
Cost Centre: IT
Project: OFP --means Oracle Fusion Project

Expense Type: OCC -- Oracle Consultant Cost

In a DFF ONLY WORLD, when your client raises Purchase Order to IT Consulting Company, in PO_DISTRIBUTIONS_ALL table record you will store
ATTRIBUTE1 :- GM
ATTRIBUTE2 :- IT
ATTRIBUTE3 :- OFP
ATTRIBUTE4 :- OCC

When an invoice is received from consulting company, the Payables clerk will capture the Invoice Line accounting as below in AP_INVOICE_DISTRIBUTIONS_ALL
ATTRIBUTE1 :- GM
ATTRIBUTE2 :- IT
ATTRIBUTE3 :- OFP
ATTRIBUTE4 :- OCC
These 4 text values for fields(above) are physically being duplicated in each module, for the related/unrelated transactions.

Imagine further when this transaction flows to Oracle General Ledger, would you again expect oracle to physically store the 4 columns into table GL_JE_LINES? If so your table GL_JE_LINES will have following values in its DFF (Descriptive Flex) columns....
ATTRIBUTE1 :- GM
ATTRIBUTE2 :- IT
ATTRIBUTE3 :- OFP
ATTRIBUTE4 :- OCC

Surely, such design using a descriptive flexfield will be flawed, as it causes duplication of data at various places.
Now that you understand why Descriptive flexfield does not fit into this design, lets consider a new scenario.

Consider an alternate approach.(using KFF)
Let's have a table named gl_code_combinations with following columns.
CODE_COMBINATION_ID
SEGMENT1
SEGMENT2
SEGMENT3
SEGMENT4

Let's capture A SINGLE record in this table as below:-
CODE_COMBINATION_ID : 10902
SEGMENT1 : GM
SEGMENT2 : IT
SEGMENT3 : OFP
SEGMENT4 : OCC

Note the above combination of 4 fields can be uniquely identified by 10902(CODE_COMBINATION_ID).

In PO_DISTRIBUTIONS_ALL table, we will have below column with value
CODE_COMBINATION_ID : 10902
NOTE: Now we are not storing all four columns here in PO Dist table, as we store the Unique ID of the record in Key Flexfield table.

Again, in Account Payables, even though the clerk will enter in screen values for four columns (four each segment), the database will only store value 10902 in column CODE_COMBINATION_ID of payables distributions table.
Ditto for the entry in GL_JE_LINES table in oracle general ledger, only the ID that references those 4

columns will be stored.

Hence all the tables(PO Dist, AP Dist, GL JE Lines) will reference just the CODE_COMBINATION_ID.
Now some Q & A below

Question: Does this mean, for each key flexfield, there will be a dedicated table? And such table will hold the unique combination of field values that can be reused?
Answer: correct. For gl accounting key flexfield, there is a table named gl_code_combinations. Other examples are grades in oracle human resources. A grade can be defined as a combination of say Clerk + Senior or Clerk + Junior. These combinations will be stored in per_grades table.

Question: do all the tables which are used for storing key Flexfields have columns named segment1,segment2...segmentx?
Answer : Correct, it is a standard practice used by oracle. Thee segments columns are generic columns so that each client can call them by whatever name as they desire.

Question: Does Oracle deliver Key-Flexfields out of the box, which will pop-up a window with relevant fields, as configured during setup.
Answer : Yes, and if value sets are attached, the fields can be validated too.

Question : Is there a step by step example of setting up key flexfield.
Answer : Have a look at article for Setting up Special Information Types in HRMS using Key FlexField .
For change in perspective, I have covered a HRMS Key Flexfield, named Special Information Types.

The unique ID is generated from a sequence.
When you register a Key Flex Field in Application developer responsibility, at that time ayou specify the UNIQUE ID COLUMN for the KFF.

The most common example is accounting flexfield, which uses a sequence named gl_code_combinations_s.nextval to generate the code_combination_id

Hierarchy Profile Options

Can all the profile options be set at Organization Level and Server level?
You need to set "Hierarchy Type" field in Profile Option definition to enable that profile to be eligible for Hierarchy Level. This functionality was introduced in 11.5.9

How to find from SQL*Plus whether a Profile Option is Hierarchy Enabled?
SELECT HIERARCHY_TYPE FROM fnd_profile_options_vl where profile_option_name = 'profile option short name' ;
If the above SQL returns SERVER, then the profile is enabled at Server Level.
If the above SQL returns ORG, then the profile is enabled for Organization Level.
If the above SQL returns SERVRESP, then the profile is enabled for both Server and Responsibility Level.

Why the need of Organization Level profile option?
For example multiple responsibilities can be attached to same Operating Unit. In such event, it can get tedious to attach the same profile option to each such responsibility.

By attaching an Organization Level Profile Option to an Operating Unit implies that all such responsibilities inherit Org Level Profile value.

Will Organization Level Profile override the Responsibility Level Profile?
Yes

Give me some example usages of Server Level Profiles
Managing Timezones
In a single Global DB Instance implementation, you may desire to display different time in UI screen, depending upon the local timezone. If you have a separate Middle Tier Server for each timezone, then timezone profile options [Enable Timezone Conversions, Client Timezone, Server Timezone] can be set at Server Level.

External WWW Facing Mid Tiers
It is possible that for security reasons, you may desire to change the functionality of Application when its accessed from External WWW facing mid-tier. Such security profiles can be assigned at server level.
For example, profile option "Node Trust Level" can be set to a value of external against a server to flag the specific mid-tier server as External. See Link for details.

Give me some example usages of Organization Level Profiles
Managing Timezones
In a single Global DB Instance implementation, you may desire to display different time in UI screen, depending upon the local timezone. If you can't afford to have a separate Middle Tier Server for each timezone, then timezone profile options [Enable Timezone Conversions, Client Timezone, Server Timezone] can be set at Organization Level, as each Organization assumingly will be in a specific timezone.

Setting Org Level Profile Option in HRMS
A HRMS Business Group can be assigned to various Oracle HRMS Responsibilities. Keeping a track of profiles against all such responsibilities, which belong to a specific Business Group can be tedious. Hence you may simplify this by simply assigning the profile to Business Group Organisation.

What is the order of precedence for Profiles?
Security Profiles: Site Level >> Application Level >> Resp Level >> User Level
Server Profiles: Site Level >> Server Level >> User Level
Organization Profiles : Site Level >> Organization Level >> User Level

would like to know from the form where we assign a responsibility to an OU.
Please give me the answer for this.

Your answer is in link http://getappstraining.blogspo...-apps.html

To attach an OU to responsibility, navigate to SYS Administrator, and click on menu profile.

Select the responsibility, and in Profile name, select MO%Operatin%Unit

This is where the relation is defined

Lookup Types and Lookup codes in Oracle Apps

Question : What is a lookup in Oracle Apps
Answer: It is a set of **codes and their meanings**.

Question: Any examples?
Answer: The simplest example is say a lookup type of Gender.
This will have definitions as below

Code	**Meaning**
------	-------------
M	Male
F	Female
U	Unknown

Question: But where is it used, any examples of its usages?
Answer: Let us say that there is a table for employees, and this table named PER_PEOPLE_F & has following columns
----
FIRST_NAME
LAST_NAME
DATE_OF_BIRTH
GENDER

Question: Will the gender column in above table hold the value of M or F or U?
Answer: Correct, and the screen that displays people details will in reality display the meaning of those respective codes (i.e. Male, Female, Unknown etc) instead of displaying the code of M or F or U

Question: hmmm...so are lookups used to save the bytes space on database machine?
Answer: Noooo. Imagine a situation as below
a. There are 30,000 records in people table of which 2000 records have gender value = U. In the screen, their Gender is being displayed as "Unknown". Now let's say you want this to be changed to "Undisclosed". To implement this change, all you have to do is to change the meaning of the lookup codes for lookup type GENDER. Hence it will look like

Code	Meaning
------	-------------
M	Male
F	Female
U	Undisclosed

Here lies the beauty of lookups, you do not need to modify 2000 odd records in this case.

Question : Any other usage of lookups?
Answer : Sure, lets take another example. In HRMS, there is a field named Ethnicity. By default Oracle ERO delivers the below values

Lookup code	lookup meaning
----------------	---------------------
AS	Asian
EU	European

Now, if your client wants to track Ethnicity at a granular level, they can amend the Oracle delivered lookup definition as below

Lookup code	lookup meaning
----------------	---------------------
ASI	Asian-Indian
ASP	Asian-Pakistani
EU	European

Hence these values will then be available in the list of values for Ethnicity field.

Question: Are we saying that all the lookups delivered by oracle can be modified?
Answer: Depends. If oracle has a lookup called termination status, and if based on the termination status code Oracle has

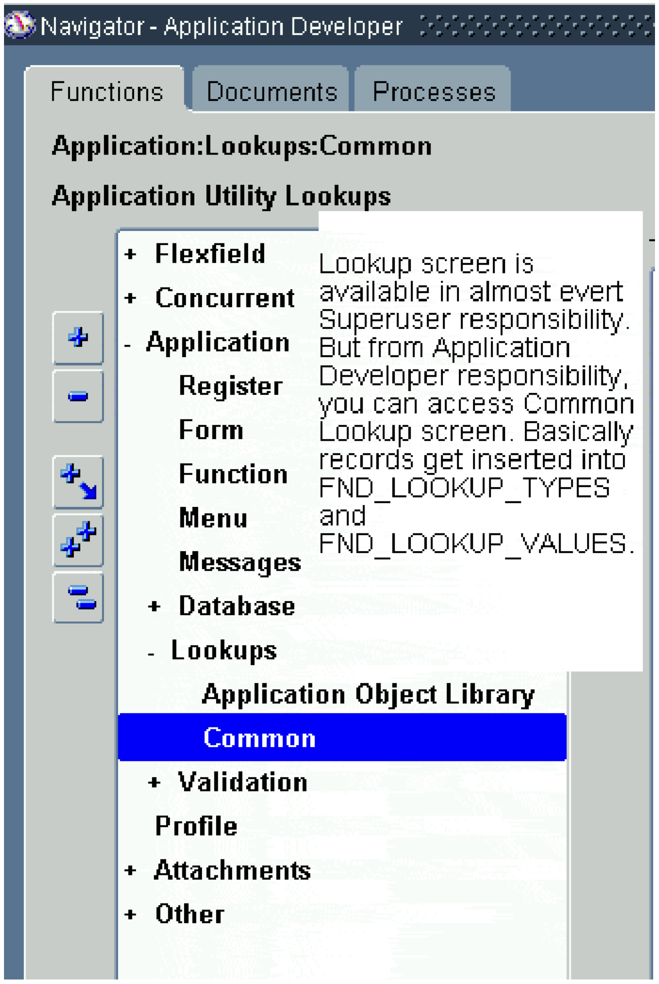
some rules defined within Payroll Engine....!! Surely Oracle Payroll Engine will not like it if you end date an existing status code or add a new status code to termination. For this very reason, Oracle flags some lookups as System lookups, and the lookup entry screen will not let you modify those lookup codes.

Question: OK, what if I do not wish to modify existing Lookup codes, but only wish to add new Lookup codes to an existing Oracle delivered Lookup Type?
Answer: You can do so, provided the Oracle delivered Lookup Type is flagged as Extensible. Please see the screenshot

Question: Can we add our own new lookup types?
Answer: Yes you can, for this you will first **define a lookup type** and will then define a set of lookup codes against the Lookup Type. In our example above, GENDER is the LOOKUP_TYPE

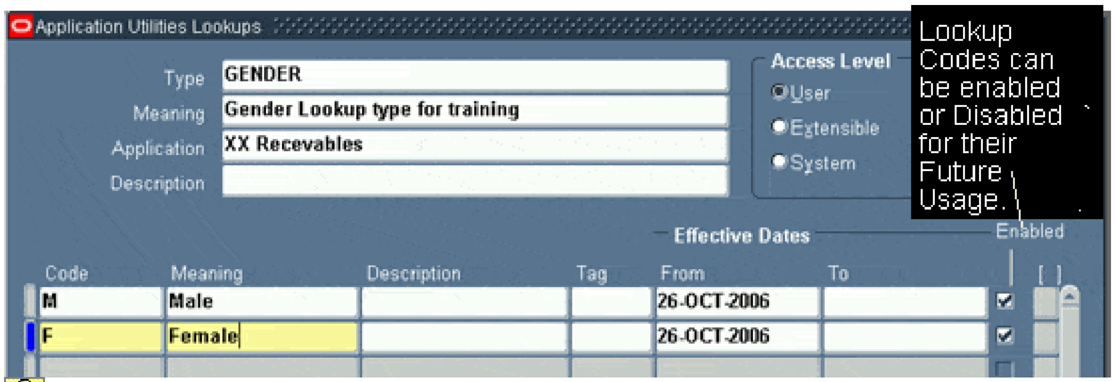
Question: Does a LOOKUP_TYPE get attached to a Descriptive Flexfield…just like Value Sets?
Answer: Not really. There is no direct relation between lookup and Descriptive Flexfield.

Now, the screenshots. Click on the menu as below to invoke Lookup Screen.



Lookup screen is available in almost evert Superuser responsibility. But from Application Developer responsibility, you can access Common Lookup screen. Basically records get inserted into FND_LOOKUP_TYPES and FND_LOOKUP_VALUES.

Once in the screen, you can define your lookup type and lookup codes as below.


Application Utilities Lookups

Type	GENDER
Meaning	Gender Lookup type for training
Application	XX Recevables
Description	

Access Level: [✓ User] Extensible System

Lookup Codes can be enabled or Disabled for their Future Usage.

Code	Meaning	Description	Tag	Effective Dates From	Effective Dates To	Enabled
M	Male			26-OCT-2006		✓
F	Female			26-OCT-2006		✓

Can you also show (screen shots) how to attach a look up type to a screen?

written by Anil Passi , November 22, 2006
You can create a value set of type table, and then attach that value set to DFF Segment or a KFF segment.
Alternately, you need to write a select on Lookup table in the LOV query[when developing a custom screen]

I am working on a 6 i Report which I have created and now I want to use the lookup codes which will be maintained by the user.
The look up code will have values which are there in the report also.
We want that these lookup codes act as parameter for the report.
Can we do that and how?

ritten by Anil Passi , January 25, 2007
Hi Anuj

you need to creaet a Value Set of Type Table, that uses fnd_lookup_values/your lookup table. The value set will get attached to the parameter of report.

what is the difference between lookup codes and quick codes

Lookups are mainly stored in fnd_lookup_values. Some people call a quickcode to be something which helps in quick data entry, via a list of values.

If a lookup is being used to facilitate entries in LOV, then both mean the same.

You can use lookup codes for validation purposes too[not just LOV]. But quickcodes mainly means LOV.

Thanks,
Anil Passi

...
written by Raj , June 26, 2007
How can i check whether a table column is based on a lookup? i mean is there a way to know which lookup type a table column is linked to...

...
written by Anil Passi , June 26, 2007
Run trace for that screen while you do data entry.
If a hit is made to fnd_lookup_values, then it means YES, your table is using lookup.

Another way is to reference eTRM

I just need a confirmation.We can define lookups at 2 level fnd and application like hr.
Is that fnd lookups are availables to all applications and if you want lookup specific to your application you go for application lookup like hr lookups. So that other application users can not modify any values for that.

Just need a confimation Is this correct?

.
written by Anil Passi , July 06, 2007
Hi Saurabh

Even the application specific lookup codes are stored in table fnd_lookup_values itself.

The application specific lookups views[like hr_lookups] are database views created on top of fnd_lookup_values itself.

If the Title of the Lookup screen is "Application Utilities Lookup" then it will let you query lookups accross all aplications.

In other cases, for application specific lookup screens, a WHERE clause similar to "WHERE view_application_id = 800" will be applied, and hence the data will be secured

This too is configurable.
All Application Specific Lookup screens use the same form i.e. "Define Lookups".
However there form function is passed a parameter similar to "VIEW_APPLICATION=PER"
This form function parameter is translated into "WHERE view_application_id = 800" within the Lookup Screen.

Thanks,
Anil Passi

...
written by Krishnab , August 28, 2007
In Oracle Apps Some tables ends with TL,VL,what is the significance of the TL and VL?

...
written by Krishnab , August 28, 2007
What is the Use of User Exit in Oracle apps and when we are developeing the new report you need to pass one parameter called P_CONC_REQUEST_ID,what is the use of this parameter and is it mandatory.Can't we develop report without this User exit's and P_CONC_REQUEST_ID parameter?

Hi Anil,
What is the difference between AOL lookups and Common lookups? I am still at a point when I am learning Oracle Apps on dummy applications. So, under what circumstances do I need to create what type?

...
written by Anil Passi , September 25, 2007
Being a learner, for all practical purposes, think of both to be the same.

Thanks,
Anil Passi

FND Lookups
written by Girish Tawry , October 04, 2007
Hi Anil,

There are many lookup types defined in the application and all these are stored in fnd_lookup_values table and using view_application_id we can differentiate what type of lookup is that. Is there any way to know which view_application_id is mapped to which type of Lookup. For example as you said view_application_id 800 is for Application Utilities Lookup

Look up Values
written by SSB , November 29, 2007
Hi Anil,
How to get the name of the already defined Lookup type so that it can be queried in the lookup window and look up values can be added to it.

Simply run SQL*Trace with bind variable option.
In the trace file, you will find the lookup type

Thanks,
Anil Passi

Smart Descriptive Flexfields

.

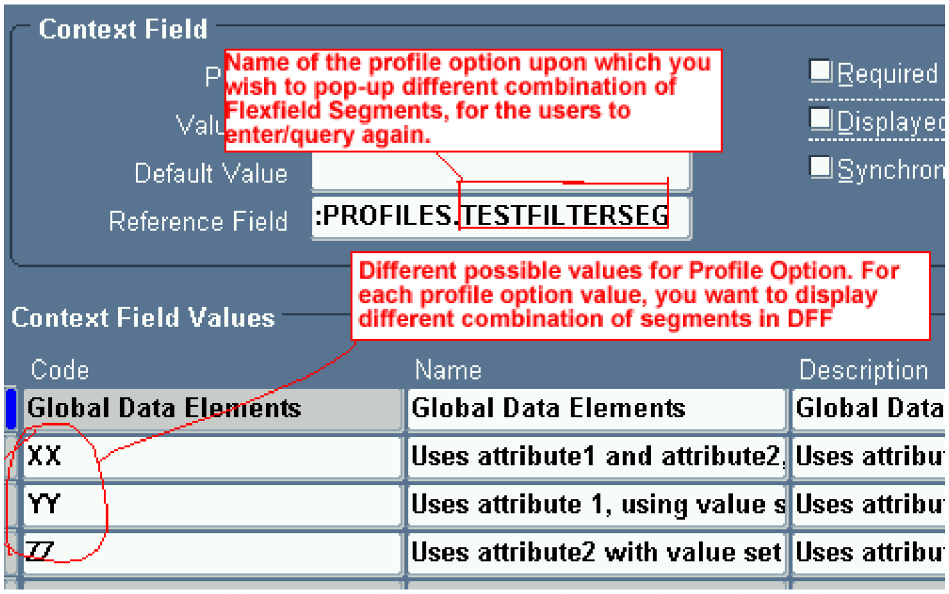
Scenario 1. Depending upon the responsibility user has logged into, you wish to either show two flexfield segemts or three segments. This is a fairly common requirement.
Scenario 2. There are two different oracle screens, both based on same table but different functionality. Hence the share the same descriptiveflexfields. You wish to use notation :block.fieldname in the value set. But this will work in one screen and error in another.

Can't we simply use context sensitive flexfields?
You certainly can. However, what if you do not want to make your user select the value in context field manually? In this case you need to take your design a step further.
In fact, before you proceed further with this article, you must read this link Basic Concepts of Context Sensitive Descriptive Flexfields.
In case you do not know much about profile options, you must also read link basic concepts of Profile Options

What are the options at hand for solutions to Scenatio 1 and Scenario 2?
Option 1. Use profile option as the context
Option 2. Use a system global variable as the context.

How does this work?
In case of using profile option, value in Context Reference Field of the descriptive flexfield will be the profile option[see picture below].
Lets say for Responsibility-1 this profile is set to XX, and for Responsibility-2 this profile is set to YY, and for a special user this profile is set to value ZZ.

You can then define three different contexts, as below.
Context XX
  Uses attribute1 and attribute2, using value sets vs1 and vs2
Context YY
  Uses attribute 1, using value set vs3
Context ZZ
  Uses attribute2 with value set vs4 , making this descriptive flexfield segment mandatory.



Context Field

Name of the profile option upon which you wish to pop-up different combination of Flexfield Segments, for the users to enter/query again.

Default Value	
Reference Field	:PROFILES.TESTFILTERSEG

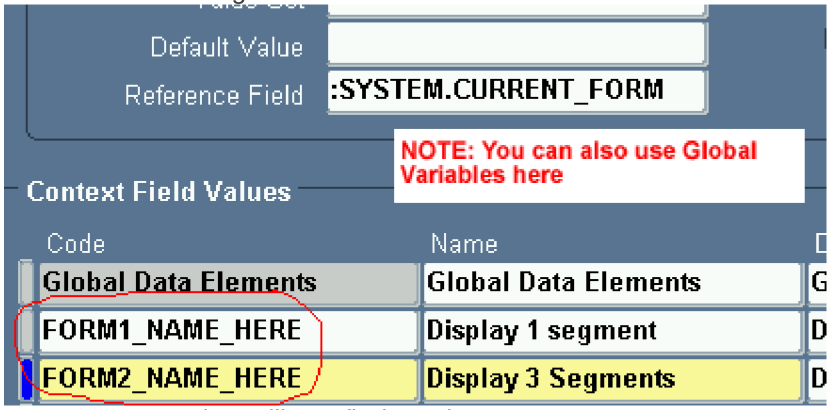
Context Field Values

Different possible values for Profile Option. For each profile option value, you want to display different combination of segments in DFF

Code	Name	Description
Global Data Elements	Global Data Elements	Global Data
XX	Uses attribute1 and attribute2,	Uses attribu
YY	Uses attribute 1, using value s	Uses attribu
ZZ	Uses attribute2 with value set	Uses attribu

Depending upon which responsibility user logs into, and also depending upon which user logs into the screen, they will see different flexfield segments popping up.

Ah, what if same user in a single responsibility has access to different screens that share same DFF?
In this scenario, you can design your descriptive flexfield on a System variable context. Hence your context will be :SYSTEM.CURRENT_FORM. Depending upon which form the user has navigated to, they will see different segments.



Default Value	
Reference Field	:SYSTEM.CURRENT_FORM

NOTE: You can also use Global Variables here

Context Field Values

Code	Name	D
Global Data Elements	Global Data Elements	G
FORM1_NAME_HERE	Display 1 segment	D
FORM2_NAME_HERE	Display 3 Segments	D

Note: Use Examine utility to find out the Current Form Name

Sounds good, but what are the pitfalls?
1. :blockname.fieldname convention does not works for OA Framework screens.
2. There is only one context reference field available per Descriptive Flexfield. If you choose to have your DFF context sensitive on a profile option, then you will have to live with it forever(ah should I say fusion- when D2K forms are gone). However this limitation can be overcome by using fnd_profile.put api, to alter the profile option value at runtime for forms session. Effectively you can modify the context[by changing profile option value pragmatically], using formspersonalizations.

written by Kishore Ryali , March 27, 2007
Hi Anil
I have a doubt regarding how to change the context using custom.pll or form personalisation?

Since the context is not defined as a field in the form, u cannot identify the context field just by saying :block.field_name
If it is the case, we cannot populate a value into context field using custom.pll or form personalization.

then how do we achieve the above?

Regards
Kishore Ryali

...
written by Anil Passi , March 28, 2007
Hi Kishore

Assuming you have context sensitive DFF, if context is not based on field, then it will be based either on Profile option or on global variable.

You can use forms customization to call fnd_profile.set for changing profile value for your current session and see if DFF responds to the dynamic change in context value.

thanks
anil
I
thnx for the reply

so if the context is dependent on profile or global variable, we have to use tht profile in context reference field and populate the profile value programmatically using custom.pll/fp.

thnx again.

written by Prasad Kotha , April 18, 2007
The Invoice Line Information DFF is used in AR Transactions Form and AR Credit Transactions form. I need to set a Reference Field of BATCH_SOURCE name in both the Forms. How can I achieve this?

Thanks
Prasad

...
written by Neerja Dhillon , August 16, 2007
Anil,

I recently came across your blog and am very impressed with the work you are doing. Thanks for taking the time to share your knowledge with others in such detail.

Below is my issue:

'PO Distributions' DFF is used by both PO Entry/'POXPOEPO' and PO Summary/POXPOVPO (View Only) forms.
Two of the PO Distribution DFF Segments have a Value Set with :Block.field reference. The value set uses the :block.field as seen in the PO Entry Form (block name is different on PO Summary Distributions Form). PO Entry forms PO Distributions are not an issue. However, there is an error when accessing PO Distributions from the PO Summary form. The error is due to the :block.field reference on the value set where clause.

I read above to use :SYSTEM.CURRENT_FORM as a reference field. I tried that but the forms are still accessing the Global Data Elements Segments. Below is what I did:

Reference Field: :SYSTEM.CURRENT_FORM
Context Field Values
Global Data Elements (original defintion)
POXPOEPO (defined two segments attribute 1 and 2 with the value set with :block.field reference.)
POXPOVPO (defined two segments without Value Sets since this is a view only form)

What am I doing wrong and how do I make this work?

Any help would be appreciated.

Thanks,
Neerja

How to hide context in DFF
written by som , October 23, 2007
Business requirement:

FND Lookup Form - Common Lookups
The DFF in Lookup Values has 2 contexts, freight and ship method.
User Query for Lookup Type = Carrier.
DFF should show both the contexts so that user may set or unset the values for any context value.

NOW: User queries Ship_Method
DFF should not show the context Freight. It should only show "Ship_Method"

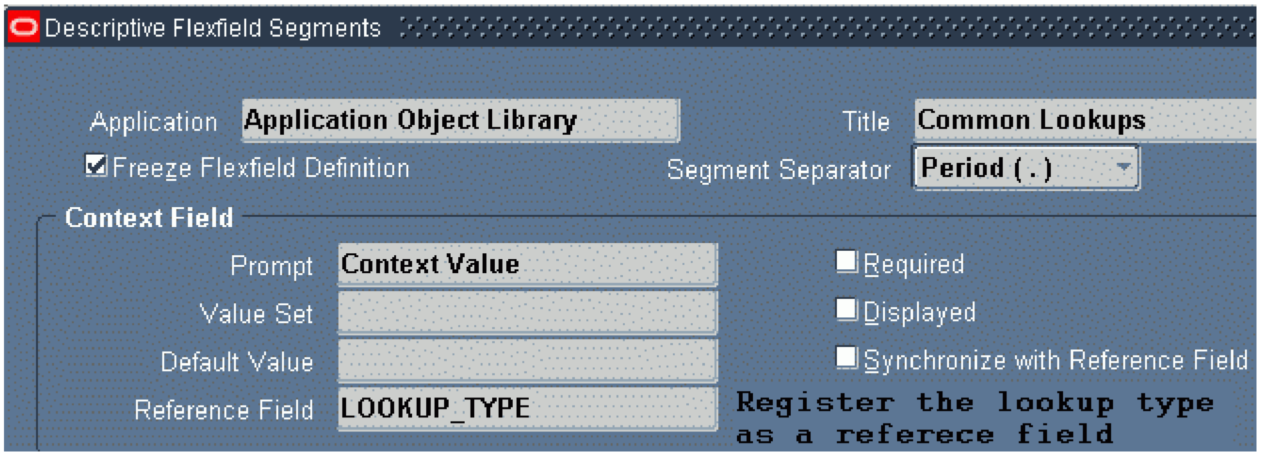
Is it possible to hide Contexts in DFF?

Regards
Som

...
written by Anil Passi , October 23, 2007
Hi Som

The Display Context checkbox can be unchecked to hide the context field.
However, if context is being displayed, then there is no way for you to display selective Context values Only

I suggest you register the lookup type itself as a context reference
See below image


Descriptive Flexfield Segments

Application	Application Object Library	Title	Common Lookups
☑ Freeze Flexfield Definition		Segment Separator	Period (.)

Context Field

Prompt	Context Value	☐ Required
Value Set		☐ Displayed
Default Value		☐ Synchronize with Reference Field
Reference Field	LOOKUP_TYPE	

Register the lookup type as a referece field

Thanks,
Anil Passi

Context fields in Concurrent Request Form
hi anil,
can we have context fields in the concurrent request parameter form, so that we can dynamically include or exclude parameters.

regards

Hi Justin

Not really, thats not possible.
However, you can make one valueset dependent upon value in other.
Hence Parameters can have inter-dependency of their values

Cheers
Anil

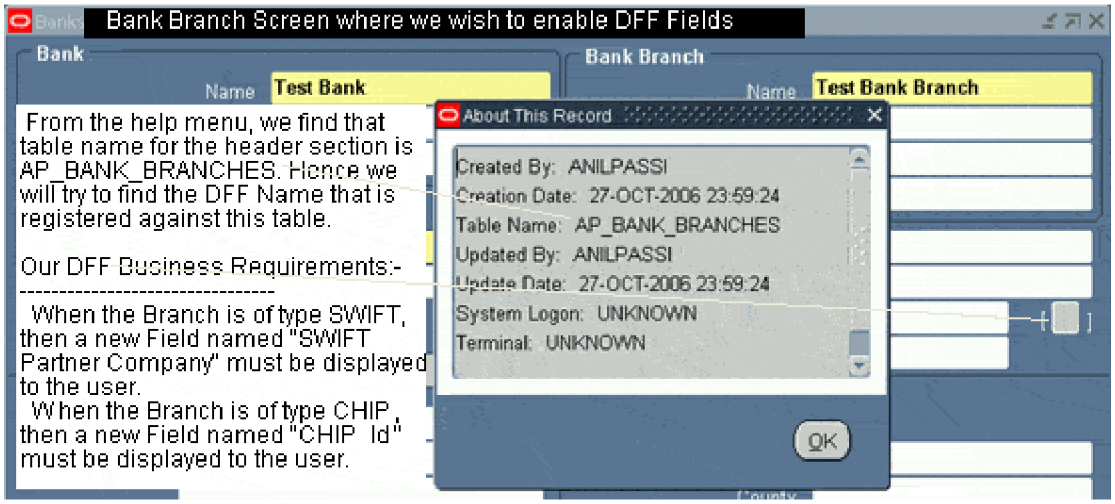
Context Sensitive Descriptive Flexfields

Question: I want these Flex fields to appear in screen only when certain conditions are met. Is it possible?
Answer: Yes, you can. Lets take an Example. You have a "Bank Branch" screen where you enter the Bank names and their branches . There is a field named Branch Type in that screen. You wish to show & capture following new fields:-
a. Banks Country of Origin Field (regardless of bank branch type, we must show this new field). As we configured in earlier training chapter.
b. If user entered a value of SWIFT in Branch type, then display a new DFF segment "SWIFT Partner field".
c. If Branch type CHIPS is selected by the user, then display a DFF segment "Chip ID" field.

In Order to do this, we will follow the below steps(screenshots will follow) :-
1. Navigate to the DFF Registration screen in Oracle Apps and query on Table AP_BANK_BRANCES. Now click on Reference Field, and ensure that "BANK_BRANCH_TYPE" field can be used as a context switch. In our case we add this field as a reference field.
2. Navigate to DFF Segments screen and query on the Title of the "Bank Branch" and Unfreeze the Flexfield and add segments as in screenshot below. Register the BANK_BRANCH_TYPE field as a reference item.
3. Create the contexts for each possible value of Bank Branch Type field(for which you want conditional display of fields)....

Hmmmm not clear yet, see the the screenshots and you will surely understand......

Find out the table name for Bank Branch screen, this will enable us to find the DFF that is available for Bank Branch


Bank Branch Screen where we wish to enable DFF Fields

From the help menu, we find that table name for the header section is AP_BANK_BRANCHES. Hence we will try to find the DFF Name that is registered against this table.

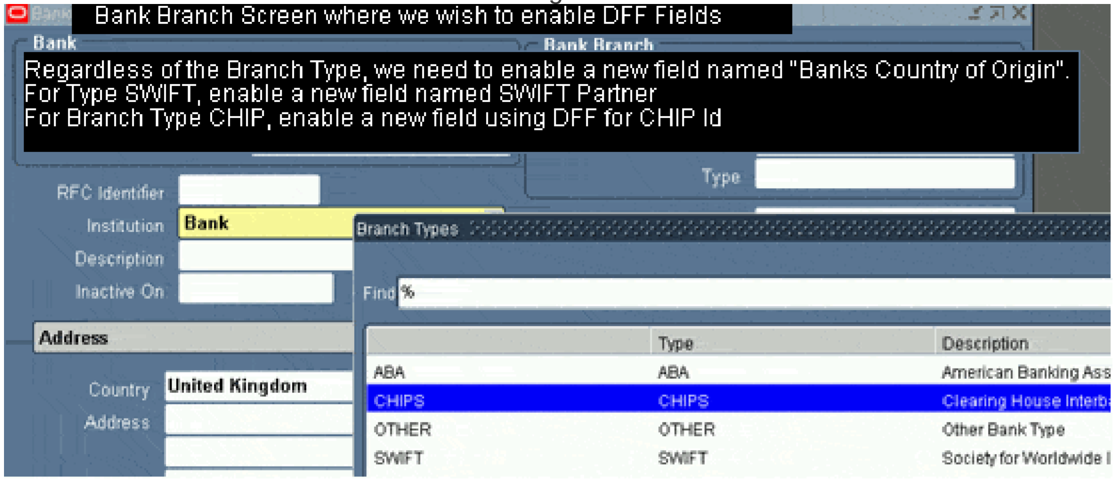
Our DFF Business Requirements:-

When the Branch is of type SWIFT, then a new Field named "SWIFT Partner Company" must be displayed to the user.
When the Branch is of type CHIP, then a new Field named "CHIP Id" must be displayed to the user.

About This Record

Created By: ANILPASSI
Creation Date: 27-OCT-2006 23:59:24
Table Name: AP_BANK_BRANCHES
Updated By: ANILPASSI
Update Date: 27-OCT-2006 23:59:24
System Logon: UNKNOWN
Terminal: UNKNOWN

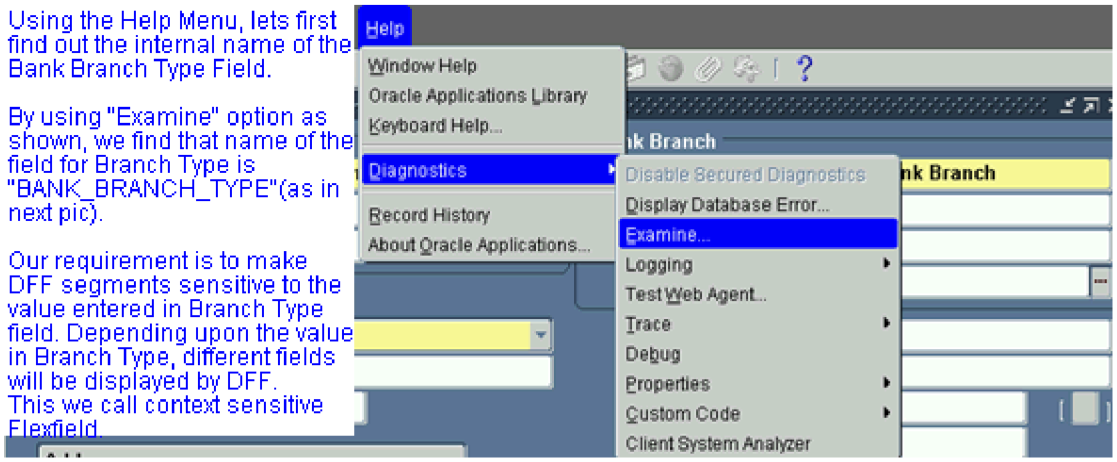
The Bank Branch can have one of the following values.


Bank Branch Screen where we wish to enable DFF Fields

Regardless of the Branch Type, we need to enable a new field named "Banks Country of Origin".
For Type SWIFT, enable a new field named SWIFT Partner
For Branch Type CHIP, enable a new field using DFF for CHIP Id

	Type	Description
ABA	ABA	American Banking Ass
CHIPS	CHIPS	Clearing House Interb
OTHER	OTHER	Other Bank Type
SWIFT	SWIFT	Society for Worldwide I

Given that our DFF will be sensitive to what we enter in Bank Branch, we must find out name of the field(not the display name). This can be done by using examine as shown

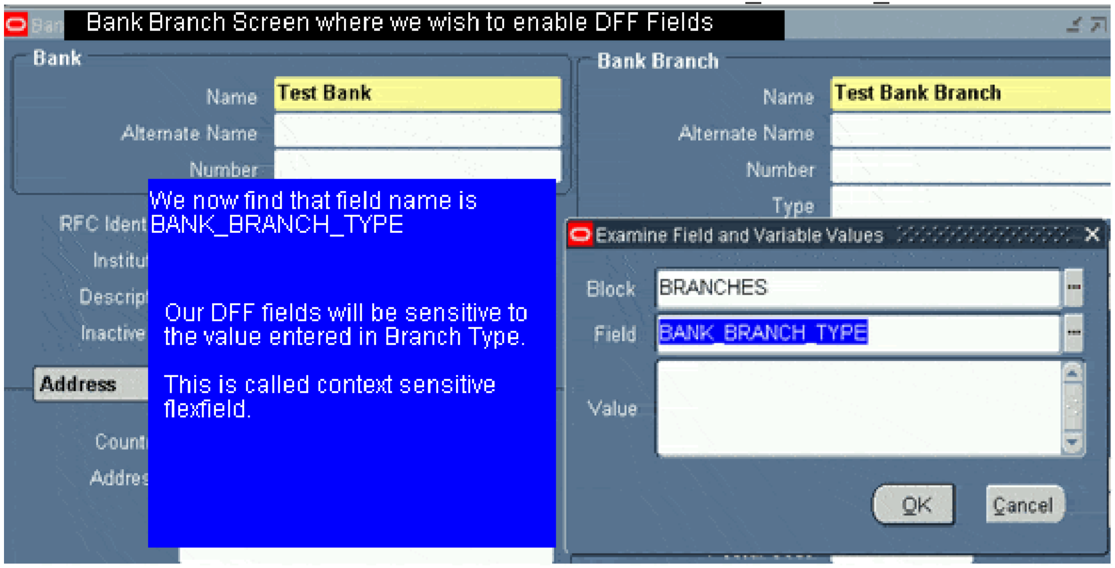


Using the Help Menu, lets first find out the internal name of the Bank Branch Type Field.

By using "Examine" option as shown, we find that name of the field for Branch Type is "BANK_BRANCH_TYPE"(as in next pic).

Our requirement is to make DFF segments sensitive to the value entered in Branch Type field. Depending upon the value in Branch Type, different fields will be displayed by DFF. This we call context sensitive Flexfield.

We can see that internal name of the Branch Type is "BANK_BRANCH_TYPE". We need to ensure that a DFF can be made sensitive to the value in BANK_BRANCH_TYPE



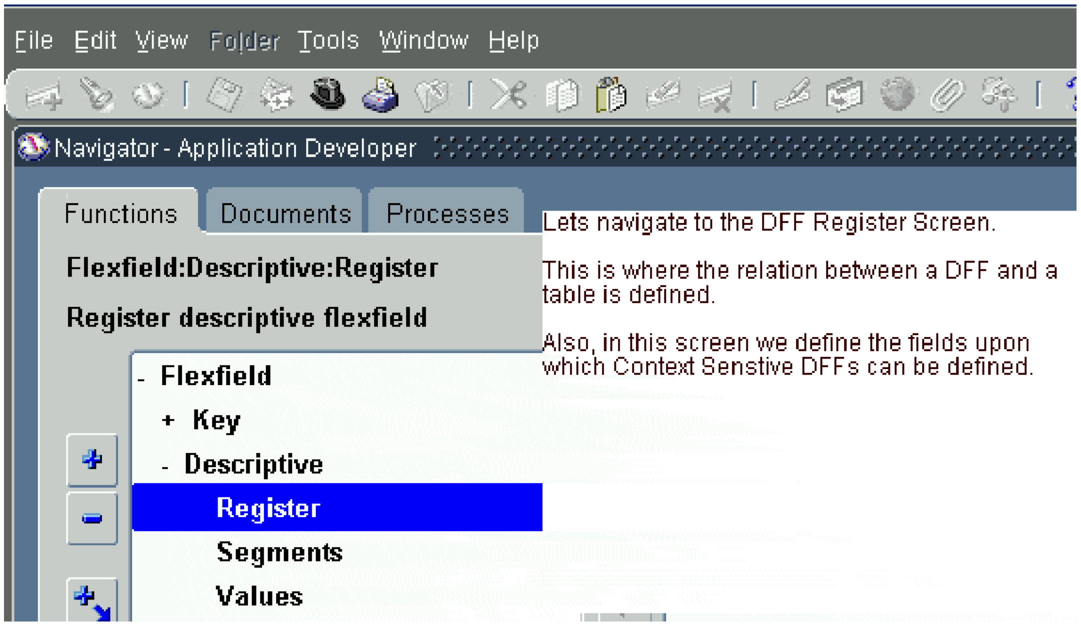
Bank Branch Screen where we wish to enable DFF Fields

We now find that field name is BANK_BRANCH_TYPE

Our DFF fields will be sensitive to the value entered in Branch Type.

This is called context sensitive flexfield.

Lets navigate to DFF registration screen, and using the table name, we query DFF for AP_BANK_BRANCHES.

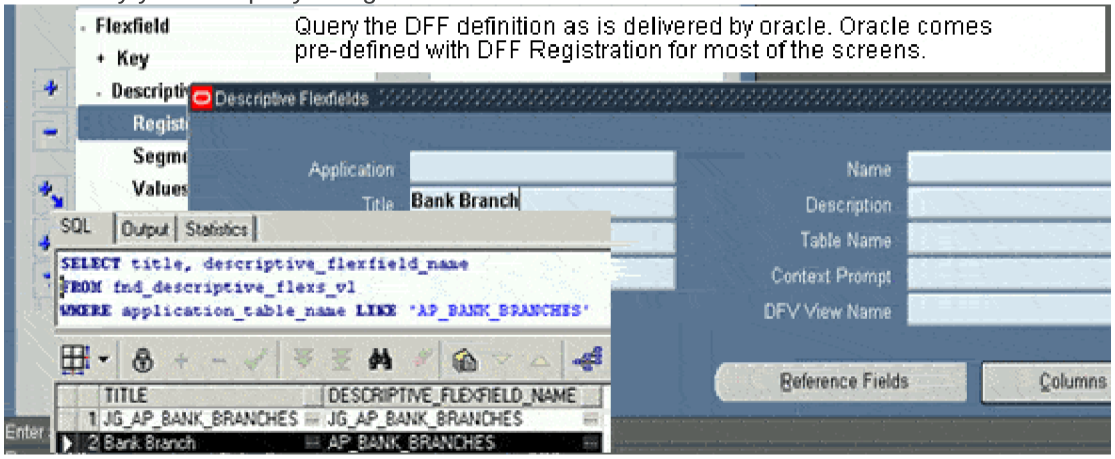

Lets navigate to the DFF Register Screen.

This is where the relation between a DFF and a table is defined.

Also, in this screen we define the fields upon which Context Senstive DFFs can be defined.

File Edit View Folder Tools Window Help
Navigator - Application Developer
Functions
Documents
Processes
Flexfield:Descriptive:Register
Register descriptive flexfield
- Flexfield
+ Key
- Descriptive
Register
Segments
Values

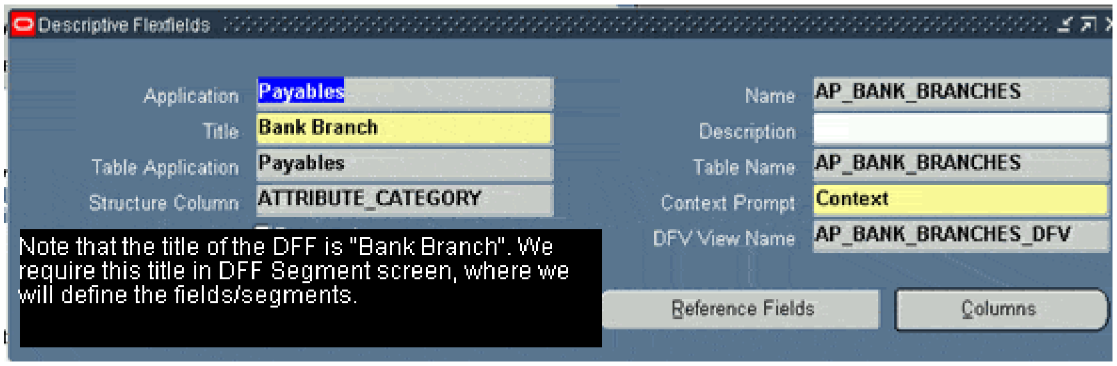
Alternately you can query using DFF title too.


Query the DFF definition as is delivered by oracle. Oracle comes pre-defined with DFF Registration for most of the screens.

```
SELECT title, descriptive_flexfield_name
FROM fnd_descriptive_flexs_vl
WHERE application_table_name LIKE 'AP_BANK_BRANCHES'
```

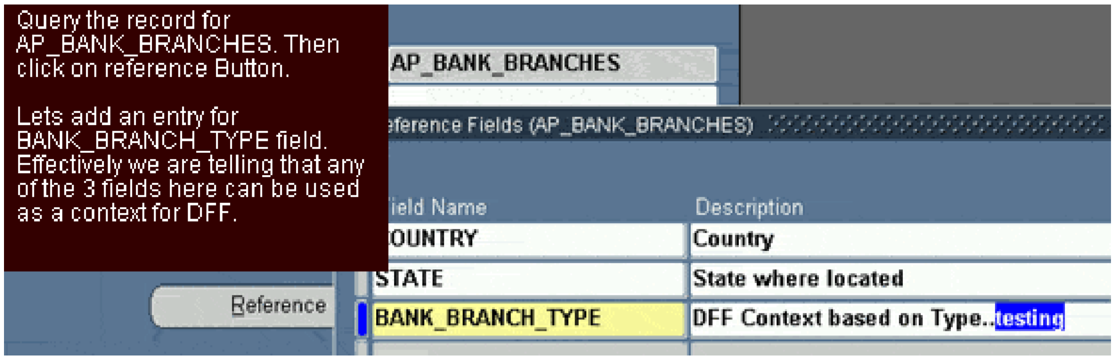
	TITLE	DESCRIPTIVE_FLEXFIELD_NAME
1	JG_AP_BANK_BRANCHES	JG_AP_BANK_BRANCHES
2	Bank Branch	AP_BANK_BRANCHES

Title: Bank Branch


Application	Payables	Name	AP_BANK_BRANCHES
Title	Bank Branch	Description	
Table Application	Payables	Table Name	AP_BANK_BRANCHES
Structure Column	ATTRIBUTE_CATEGORY	Context Prompt	Context
		DFV View Name	AP_BANK_BRANCHES_DFV

Note that the title of the DFF is "Bank Branch". We require this title in DFF Segment screen, where we will define the fields/segments.

Now lets add BANK_BRANCH_TYPE as a REFERENCE Field, by clicking on Button Reference



Query the record for AP_BANK_BRANCHES. Then click on reference Button.

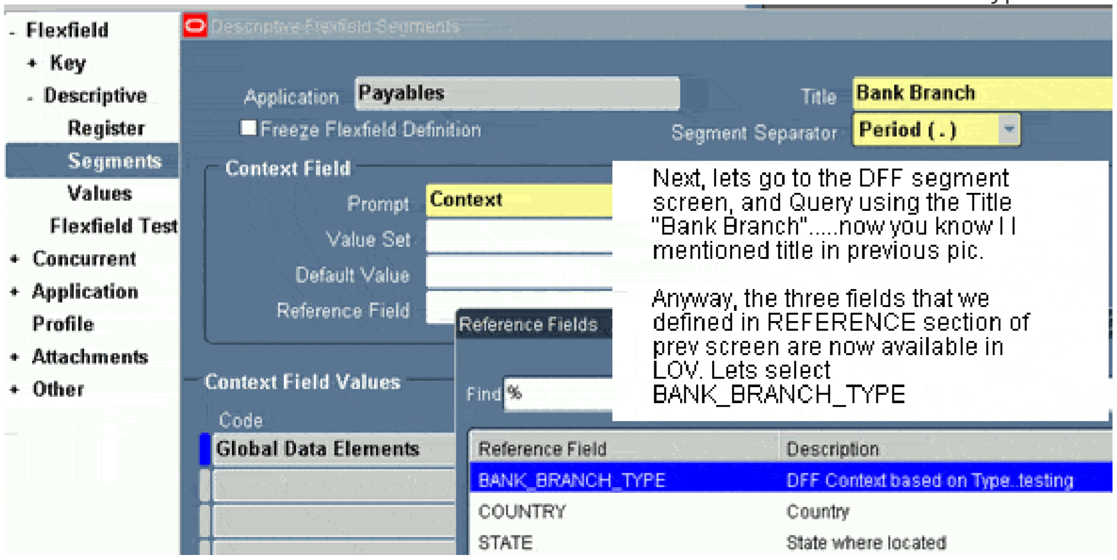
Lets add an entry for BANK_BRANCH_TYPE field. Effectively we are telling that any of the 3 fields here can be used as a context for DFF.

AP_BANK_BRANCHES

eference Fields (AP_BANK_BRANCHES)

ield Name	Description
OUNTRY	Country
STATE	State where located
BANK_BRANCH_TYPE	DFF Context based on Type..testing

Now we need to define the new fields(segments). This screen is accessed via menu /Descriptive/Segments . In this screen, lets make BANK_BRANCH_TYPE as the context/reference. This means that DFF will become sensitive to values in Branch Type field



Application: Payables
Title: Bank Branch
Freeze Flexfield Definition
Segment Separator: Period (.)

Context Field
Prompt: Context
Value Set:
Default Value:
Reference Field:

Context Field Values
Code: Global Data Elements

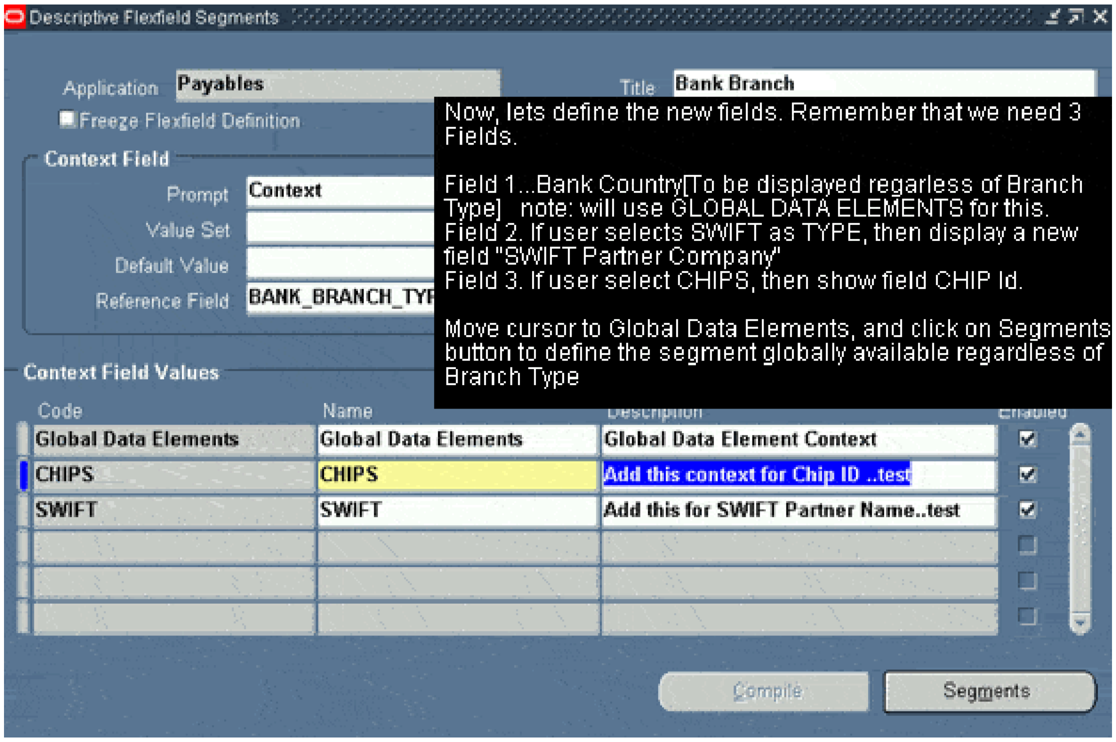
Next, lets go to the DFF segment screen, and Query using the Title "Bank Branch".....now you know I I mentioned title in previous pic.

Anyway, the three fields that we defined in REFERENCE section of prev screen are now available in LOV. Lets select BANK_BRANCH_TYPE

Reference Fields
Find %

Reference Field	Description
BANK_BRANCH_TYPE	DFF Context based on Type..testing
COUNTRY	Country
STATE	State where located

IMPORTANT: If your requirement is not to have any conditiional logic, then no need for all the "Reference Field" blaaa. All you need to do is to add your segment to Global Data Elements


Descriptive Flexfield Segments

Application: Payables

Title: Bank Branch

Freeze Flexfield Definition

Context Field

Prompt: Context

Value Set:

Default Value:

Reference Field: BANK_BRANCH_TYP

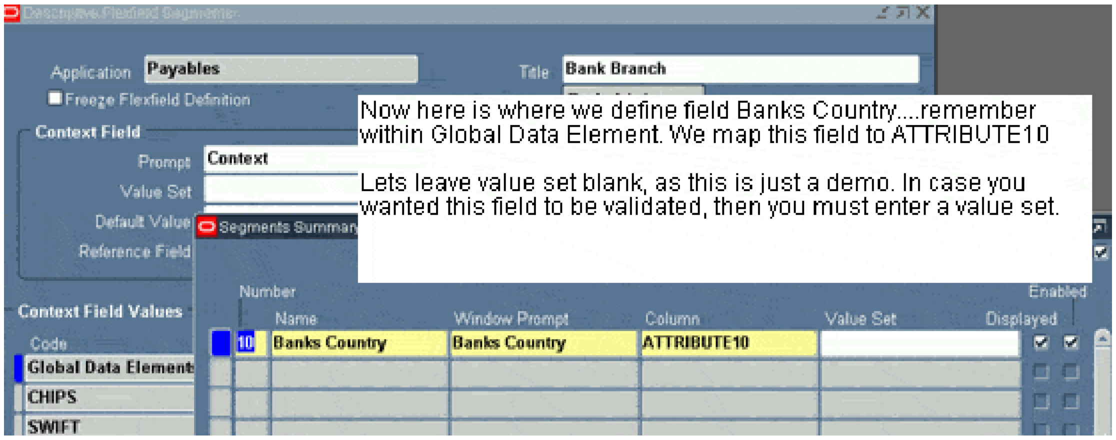
Now, lets define the new fields. Remember that we need 3 Fields.

Field 1...Bank Country[To be displayed regarless of Branch Type] note: will use GLOBAL DATA ELEMENTS for this.
Field 2. If user selects SWIFT as TYPE, then display a new field "SWIFT Partner Company"
Field 3. If user select CHIPS, then show field CHIP Id.

Move cursor to Global Data Elements, and click on Segments button to define the segment globally available regardless of Branch Type

Context Field Values

Code	Name	Description	Enabled
Global Data Elements	Global Data Elements	Global Data Element Context	✓
CHIPS	CHIPS	Add this context for Chip ID ..test	✓
SWIFT	SWIFT	Add this for SWIFT Partner Name..test	✓


Descriptive Flexfield Segments

Application: Payables

Title: Bank Branch

Freeze Flexfield Definition

Context Field

Prompt: Context

Now here is where we define field Banks Country....remember within Global Data Element. We map this field to ATTRIBUTE10

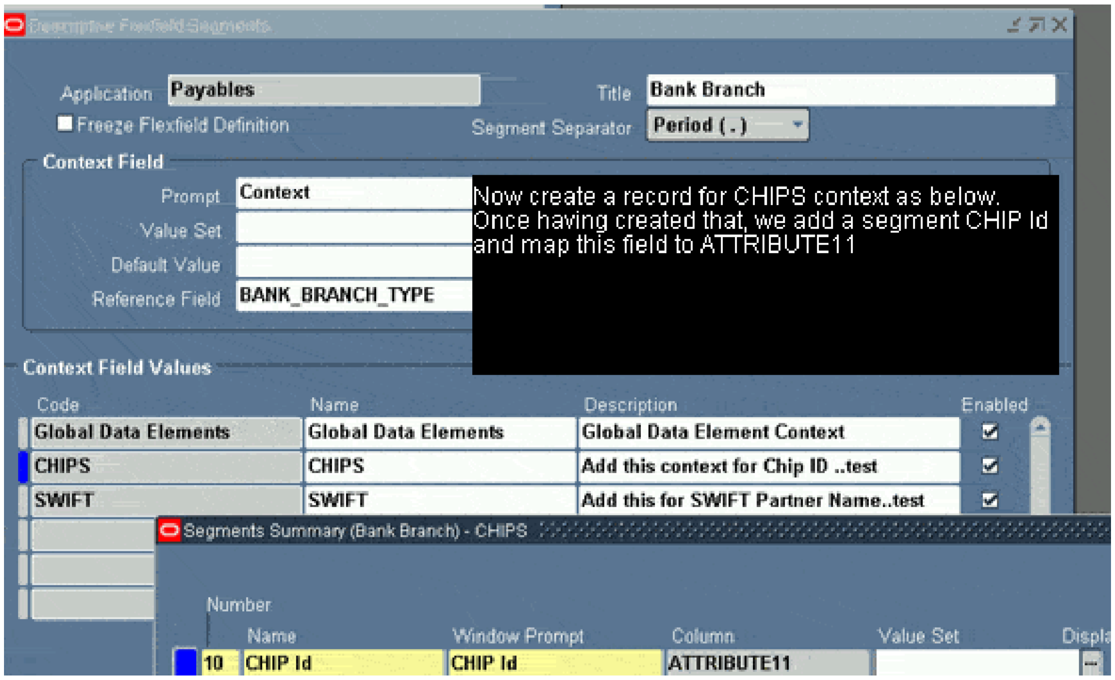
Lets leave value set blank, as this is just a demo. In case you wanted this field to be validated, then you must enter a value set.

Segments Summary

Number	Name	Window Prompt	Column	Value Set	Enabled	Displayed
10	Banks Country	Banks Country	ATTRIBUTE10		✓	✓

Context Field Values

Code
Global Data Element
CHIPS
SWIFT



Descriptive Flexfield Segments

Application	Payables	Title	Bank Branch
☐ Freeze Flexfield Definition		Segment Separator	Period (.)

Context Field

Prompt	Context
Value Set	
Default Value	
Reference Field	BANK_BRANCH_TYPE

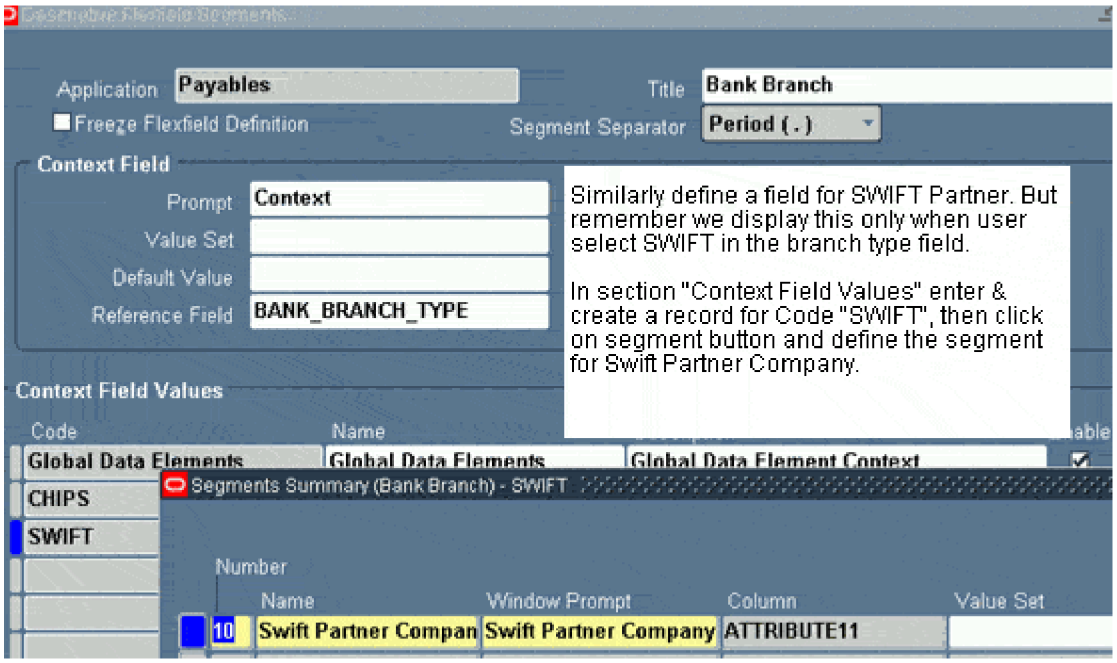
Now create a record for CHIPS context as below. Once having created that, we add a segment CHIP Id and map this field to ATTRIBUTE11

Context Field Values

Code	Name	Description	Enabled
Global Data Elements	Global Data Elements	Global Data Element Context	✓
CHIPS	CHIPS	Add this context for Chip ID ..test	✓
SWIFT	SWIFT	Add this for SWIFT Partner Name..test	✓

Segments Summary (Bank Branch) - CHIPS

Number	Name	Window Prompt	Column	Value Set	Displa
10	CHIP Id	CHIP Id	ATTRIBUTE11		



Descriptive Flexfield Segments

Application	Payables	Title	Bank Branch
☐ Freeze Flexfield Definition		Segment Separator	Period (.)

Context Field

Prompt	Context
Value Set	
Default Value	
Reference Field	BANK_BRANCH_TYPE

Similarly define a field for SWIFT Partner. But remember we display this only when user select SWIFT in the branch type field.

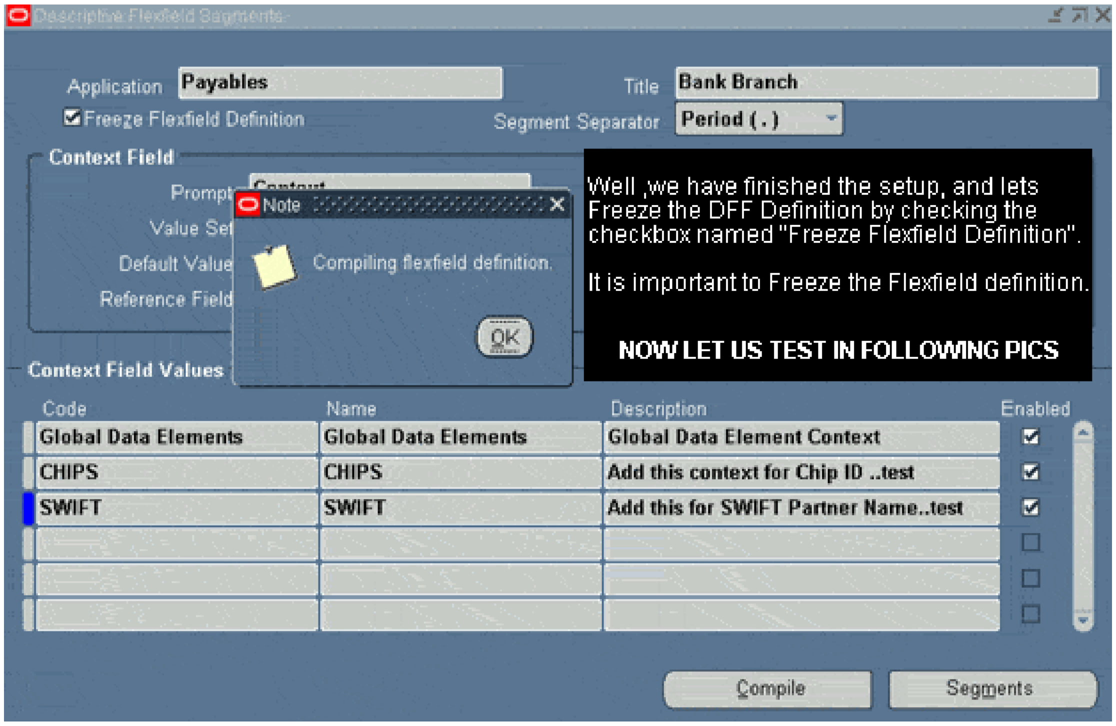
In section "Context Field Values" enter & create a record for Code "SWIFT", then click on segment button and define the segment for Swift Partner Company.

Context Field Values

Code	Name	Description	Enable
Global Data Elements	Global Data Elements	Global Data Element Context	✓
CHIPS			
SWIFT			

Segments Summary (Bank Branch) - SWIFT

Number	Name	Window Prompt	Column	Value Set
10	Swift Partner Compan	Swift Partner Company	ATTRIBUTE11	


Descriptive Flexfield Segments

Application	Payables	Title	Bank Branch
☑ Freeze Flexfield Definition		Segment Separator	Period (.)

Context Field

Prompt: Context
Value Set
Default Value
Reference Field

Note: Compiling flexfield definition. [OK]

Well ,we have finished the setup, and lets Freeze the DFF Definition by checking the checkbox named "Freeze Flexfield Definition".

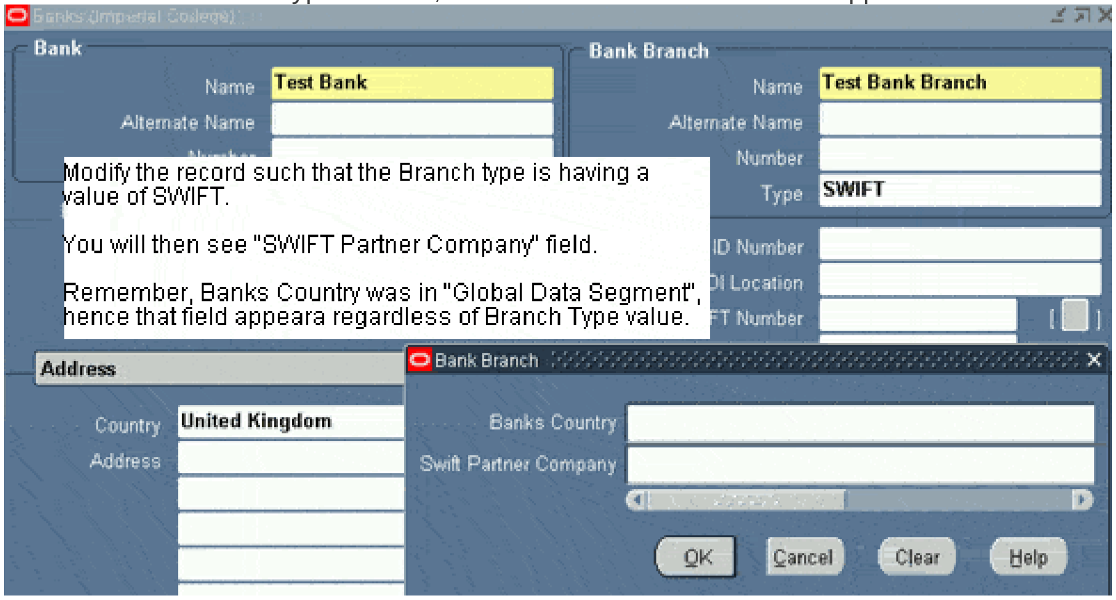
It is important to Freeze the Flexfield definition.

NOW LET US TEST IN FOLLOWING PICS

Context Field Values

Code	Name	Description	Enabled
Global Data Elements	Global Data Elements	Global Data Element Context	☑
CHIPS	CHIPS	Add this context for Chip ID ..test	☑
SWIFT	SWIFT	Add this for SWIFT Partner Name..test	☑
			☐
			☐
			☐

When the user selects Type=SWIFT, we see the relevant SWIFT field appear in Flexfield window


Banks (Imperial College)

Bank

Name: Test Bank
Alternate Name:

Bank Branch

Name: Test Bank Branch
Alternate Name:
Number:
Type: SWIFT
ID Number:
DI Location:
FT Number:

Modify the record such that the Branch type is having a value of SWIFT.

You will then see "SWIFT Partner Company" field.

Remember, Banks Country was in "Global Data Segment", hence that field appeara regardless of Branch Type value.

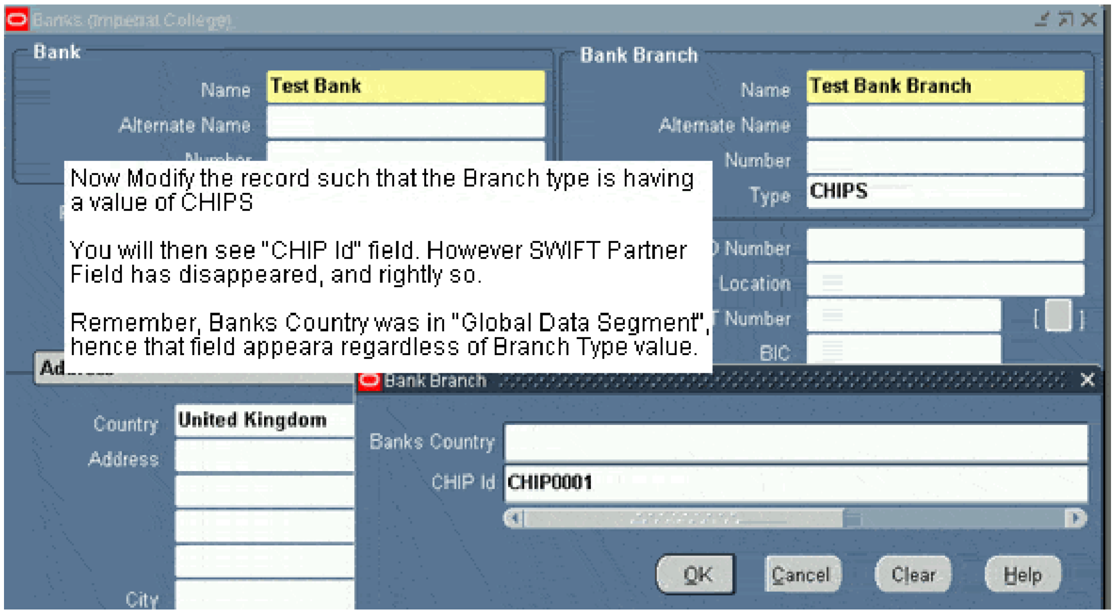
Address

Country: United Kingdom
Address:

Bank Branch

Banks Country:
Swift Partner Company:

When user selects Type=CHIPS, we see CHIP Id field appearing in Flexfield window.



Bank

Name: Test Bank

Alternate Name:

Bank Branch

Name: Test Bank Branch

Alternate Name:

Number:

Type: CHIPS

Country: United Kingdom

Address:

City:

Bank Branch

Banks Country:

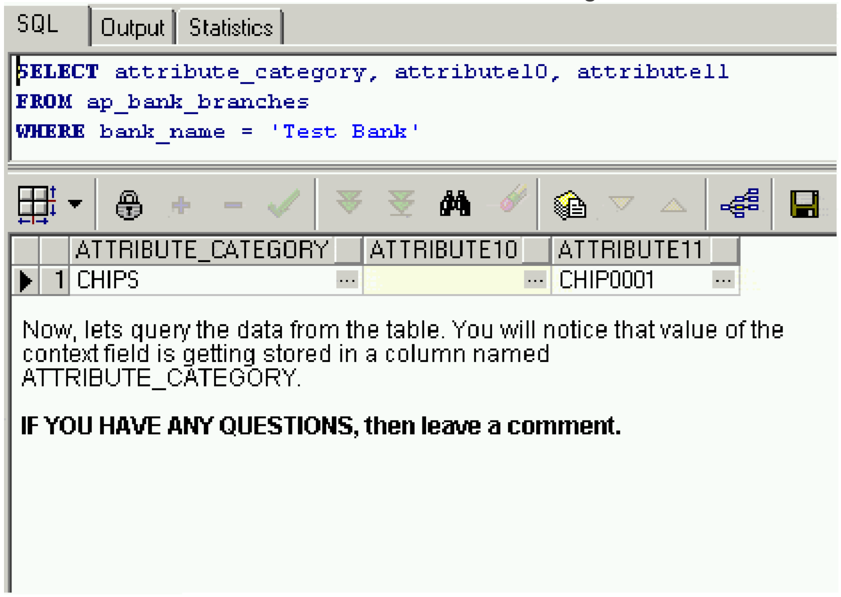
CHIP Id: CHIP0001

Now Modify the record such that the Branch type is having a value of CHIPS

You will then see "CHIP Id" field. However SWIFT Partner Field has disappeared, and rightly so.

Remember, Banks Country was in "Global Data Segment", hence that field appeara regardless of Branch Type value.

Here we now see that value entered in DFF field gets stored in the database column,



```
SELECT attribute_category, attribute10, attribute11
FROM ap_bank_branches
WHERE bank_name = 'Test Bank'
```

	ATTRIBUTE_CATEGORY	ATTRIBUTE10	ATTRIBUTE11
1	CHIPS		CHIP0001

Now, lets query the data from the table. You will notice that value of the context field is getting stored in a column named ATTRIBUTE_CATEGORY.

IF YOU HAVE ANY QUESTIONS, then leave a comment.

Hi Anil,
The way you explain the things, is very simple and easy to understand. I appreciate your work. One thing I thought to ask you. You navigated to Bank Branches window in the process of DFF Registration process. Can you please let me know what exactly is the navigation to flow for that Bank Branches? Not only, how can we know the navigation flow in general also. for example transactions, invoices, receipts or sites etc., I think it would be very useful, if u can explain how can we follow the navigation flow, for any particular window.

Thanks!

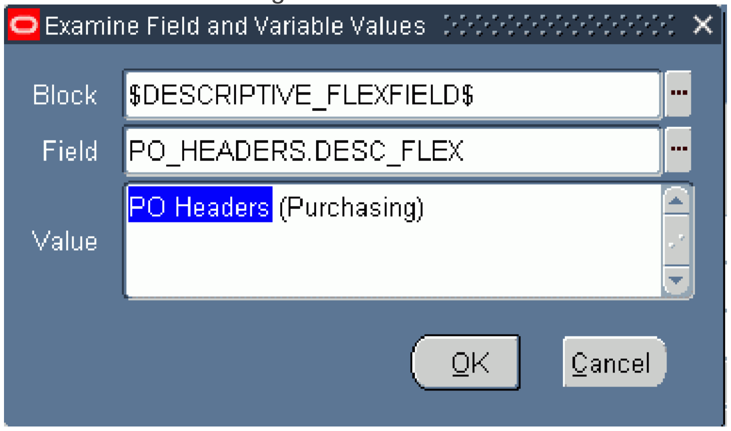
Hi User,
I usually never write about menu path navigation. To access any screen, all you need to do is,

when inside a responsibility, on keyboard hit, Control L.

By Control L, all the existing screen name will be and you can search using %Branch%

Thanks,
Anil Passi

...
written by Anil Passi , September 25, 2007
You can use examine to figure that out.
Have a look at the image below



Block: $DESCRIPTIVE_FLEXFIELD$
Field: PO_HEADERS.DESC_FLEX
Value: PO Headers (Purchasing)

Thanks,
Anil Passi

What are Profile Options in Oracle Apps ?

Profile Options provide flexibility to Oracle Apps. They are a key component of Oracle Applications, hence these much be understood properly. I will be taking multiple examples here to explain what profile options mean. I will also try to explain by stepping into Oracle shoes "How will you design a program that is flexible", by using Profile Options.

Following that, if you still have questions regarding profile options, then leave a comment and I promise to respond. For the learners of Oracle Apps, understanding profile options is mandatory.

What is profile option?
The profile option acts like a Global Variable in Oracle.

Why does Oracle provide profile options?
These are provided to keep the application flexible. The business rules in various countries and various companies can be different. Hence the profile options are delivered by Oracle in such a manner to avoid hard-coding of logic, and to let the implementation team at site decide the values of those variables.

For screenshots of below listed examples in this article, please click this link

Enough definitions, give me some scenarios where profile options are used by Oracle....
1. There are profile options which can turn the debugging on, to generate debug messages. Say

one of 1000 users reports a problem, and hence you wish to enable debugging against just that specific user. In this case you can “Turn On” the debugging profile option "again that specific user".
2. There are profile options that control which user can give discount to their customers at the time of data entry. You can set profile option "Discount Allowed" to a value of either Yes or No against each Order Entry user.
3. Lets assume an Organization has department D1 and D2. Managers of both the Departments have "HRMS Employee View" responsibility. But you do not want Manager of D2 to be able to see the list of Employees in Organization D1. Hence you can set a profile option against the username of each of these users. The value assigned to such profile option will be "Name of the Organization" for which they can see the employees. Of course, the SQL in screen that displays list of employees will filter off the data based on “logged in users profile option value”.

Let’s take an example. Let’s assume you are a developer in Oracle Corporation building a screen in ERP. Let us further assume that you are developing an Order Entry screen.
Assume that business requirements for your development work is:-
1. Screen should be flexible to ensure that different users of the screen can give different levels of discounts. For example, a clerk Order Entry User can give no more than 5% discount. But Sales Manager can enter an Order with 15% discount.
2. There should not be any hard-coding regarding the maximum permissible discount.
3. In the screen there will be a discount field.
4. When the discount value is entered in discount field, an error will be raised if user violates the maximum permissible discount.

Here is how Oracle will code this screen
1. They will define a profile option named "OEPASSI Maximum Discount Allowed".
2. The short name of this profile option is "OEPASSI_MAX_DISCOUNT"
2. In the when-validate-item of the discount field(assuming Oracle Forms), following code will be written

```
IF :oe_line_block.discount_value > fnd_profile.value('OEPASSI_MAX_DISCOUNT')
THEN
message(
'You can’t give discount more than '
|| fnd_profile.value('OEPASSI_MAX_DISCOUNT') || '%' ) ;
raise form_trigger_failure ;-- I mean raise error after showing message
END IF ;
```

Here is how, the client implementing Oracle Order Entry will configure their system.
1. Navigate to System administration and click on system profile menu.
2. For Clerk User(JOHN), set value of profile "OEPASSI Maximum Discount Allowed" to 5
For Sales Manager User(SMITH), set value of profile "OEPASSI Maximum Discount Allowed" to 15

Question: This sounds good, but what if you have 500 Order Entry Clerks and 100 Order Entry Sales Managers? Do we have to assign profile option values to each 600 users?
Answer : Well, in this case, each Clerk will be assigned Responsibility named say “XX Order Entry Clerk Responsibility”
Each Sales Manager will be assigned Responsibility named say “XX Order Entry Sales Manager Responsibility”
In this case, you can assign a profile option value to both these responsibilities.
“XX Order Entry Clerk Responsibility” will have a value 5% assigned against it. However, “XX Order Entry Sales Manager Responsibility” will have a profile option value of 15% assigned.
In the when-validate-item of the discount field, following code will then be written

```
IF :oe_line_block.discount_value > fnd_profile.value('OEPASSI_MAX_DISCOUNT')
```

THEN
message(
'You can't give discount more than '
|| fnd_profile.value('OEPASSI_MAX_DISCOUNT') || '%') ;
raise form_trigger_failure ;-- I mean raise error after showing message
END IF ;

Please note that our coding style does not change even though the profile option is now being assigned against responsibility. The reason is that API fnd_profile.value will follow logic similar to below.
Does Profile option value exist against User?
--Yes: Use the profile option value defined against the user.
--No: Does Profile option value exist against Responsibility
-----Yes: Use the profile option value defined against the current responsibility in which user has logged into.
-----No: Use the profile option value defined against Site level.

For screenshots of examples in this article, please refer this link

Anil Passi said ... (11:29 PM) :

Hi Vinod,

If someone creates a new profile option, then somewhere a piece of code must be written that will read the value of that profile and switch logic accordingly. As you can see, it is the developer that is closely involved in the process.

In answer to your question, either the technical or a functional person can create the profile, but most often it is a TECHNICAL persons job.
Reasons are below:-
1. Sometimes you would attach SQL to profile option, such that it no longer remains free text profile option.
2. Developer must change some pl/sql or other program to read the value of the profile option.

i

Anonymous said ... (3:52 AM) :

Hi Anil,

Just want to know about personel profile option what is that

Anil Passi said ... (3:57 AM) :

you mean User level profile option?

two call centre operators can have different limit for offering discounts to customer. in such case, you can have a profile option named "Max Discount Allowed" set at user level.

hope this explains

thanks
anil passi

hi anil,

i create one user assing to responsibilty but i can not access profile in menu option. if i am using general ledeger responsibilty then i can change my user level value in my profile. but i

Anil Passi said ... (7:08 AM) :

this profile may not be applicable for user update.

Go to application developer, query the profile, and see if user update/user view is enabled in checkbox

thanks
anilp assi

Anonymous said ... (7:35 AM) :

u are doing such great job...
i really impress by u.

what i want say that i am enable to access following menu for change profile detail in my user ,my responsibility.
edit-->preferrences-->>profile

means in my user this is showing disabled how can i enbaled it.

thanks
Rajendrasingh

Anil Passi said ... (7:51 AM) :

the profile screens Form Function is not available within your responsibility

Do the below:-

a. check if function/menu exclusion exists for this responsibility
b. if (a) is not true, then add the form function to the responsibility

thanks
anil

Hi Annil:

This is an urgent question. For one profile option there are two options. Either YES or NO. what will be the default if neither is selected?

Thanks,

Asm

Anil Passi said ... (1:49 PM) :

hi asm

it will depend upon how oracle did the programming for that profile.

lets take this scenario
IF NVL (fnd_profile.value ('XXPROF'),'Y') = 'Y' then
then null will be considered Y due to NVL

IF NVL (fnd_profile.value ('XXPROF'),'N') = 'Y' then
then null will be considered N due to NVL

in most cases, null value will equate to No, however, to be 100% sure check to see how oracle have written code for that profile.

thanks,
anil passi

Customization of Reports in Oracle Apps
Oracle Reports will become a thing of past in Fusion, however it will still demand resources for the next 5yrs or so.

Hence I decided to write this article to fulfill my commitment towards
http://getappstraining.blogspot.com

Question: I have been asked to customize Invoice Print program which happens to be an Oracle Report. What will be the steps, that I must follow.
Answer : Follow the steps below.

1. You will be told the name of the existing report that must be customized. Note down the exact name and query that name in Concurrent Program screen. Click on “Copy Program button” selecting checkbox option “Copy Parameters”. This will help you copy the current program definition to custom version of the program. Also note down the name of the executable as it appears in concurrent program definition screen.

2. In same responsibility i.e. Application Developer, navigate to screen concurrent executable and query on the field labeled "Executable Short Name".
Note down the application within which it is registered. If the application is Oracle Receivables, then you must go to the database server and get hold the file named RAXINV.rdf in $AR_TOP/reports/US.

3. Copy that file to your custom AR Top directory. Basically that is the directory where custom reports for AR will be deployed..
cd $XXAR_TOP/reports/us
cp $AR_TOP/reports/us/RAXINV.rdf $XXAR_TOP/reports/us

<u>Effectively you have now done the following:-</u>
1. Made the custom version of report registered with XXAR application. If you worked for say company named EA, then this might have been $EAAR_TOP/reports/US
2. When you run that report, Oracle concurrent manager will search for that report in $XXAR_TOP/reports/US
The report will be found there, and executed.

Note: We haven’t made any changes as yet. Also, you need to include the new concurrent program name in the relevant request group.

Now you can ftp that report to your pc, make modifications for necessary customizations, and then ftp that piece of rdf back to the server. Run it again, to see it working.

Some important tips:-
1. Avoid writing SQL in format trigger, although in rare cases it becomes necessary to do so.
2. Learn from Oracle's Report, by reverse engineering them.
3. Do not write a formula column for something which can be achieved by amending the query in data group itself.
4. Do not hardcode things like Currency Formatting. Have a look at Oracle's Amount fields, and use the same user exit.
5. srw2.message can be used for minor debugging, as those messages will appear in the log file of the concurrent program.
6. You can set the trace checkbox against the concurrent program definition, to generate SQL Trace. This trace will not produce bind variable values though.
7. Join between two queries in data group will always be outerjoined, by default.
8. Avoid filters on Data Group queries. Try to implement that logic within the query itself.

Find Request Group for Concurrent Programs

I have created a concurrent program, and I now wish to know which request group must this program be added to?

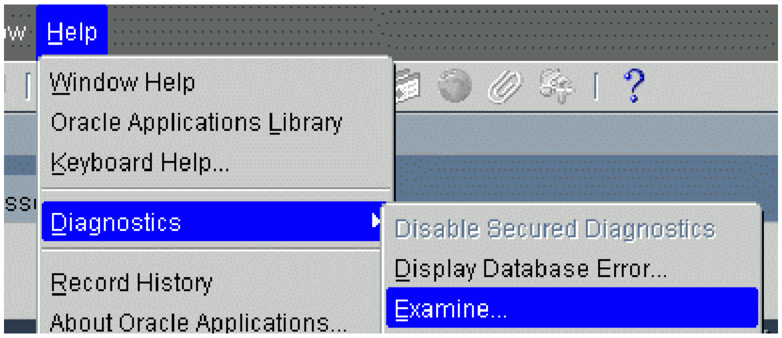
Step 1. Ensure that profile option "Utilities:Diagnostics" is set to Yes at your user level.

Note:- This step can be skipped in case you have the apps password.

Step 2. Navigate to the responsibility from where you will submit the concurrent program.

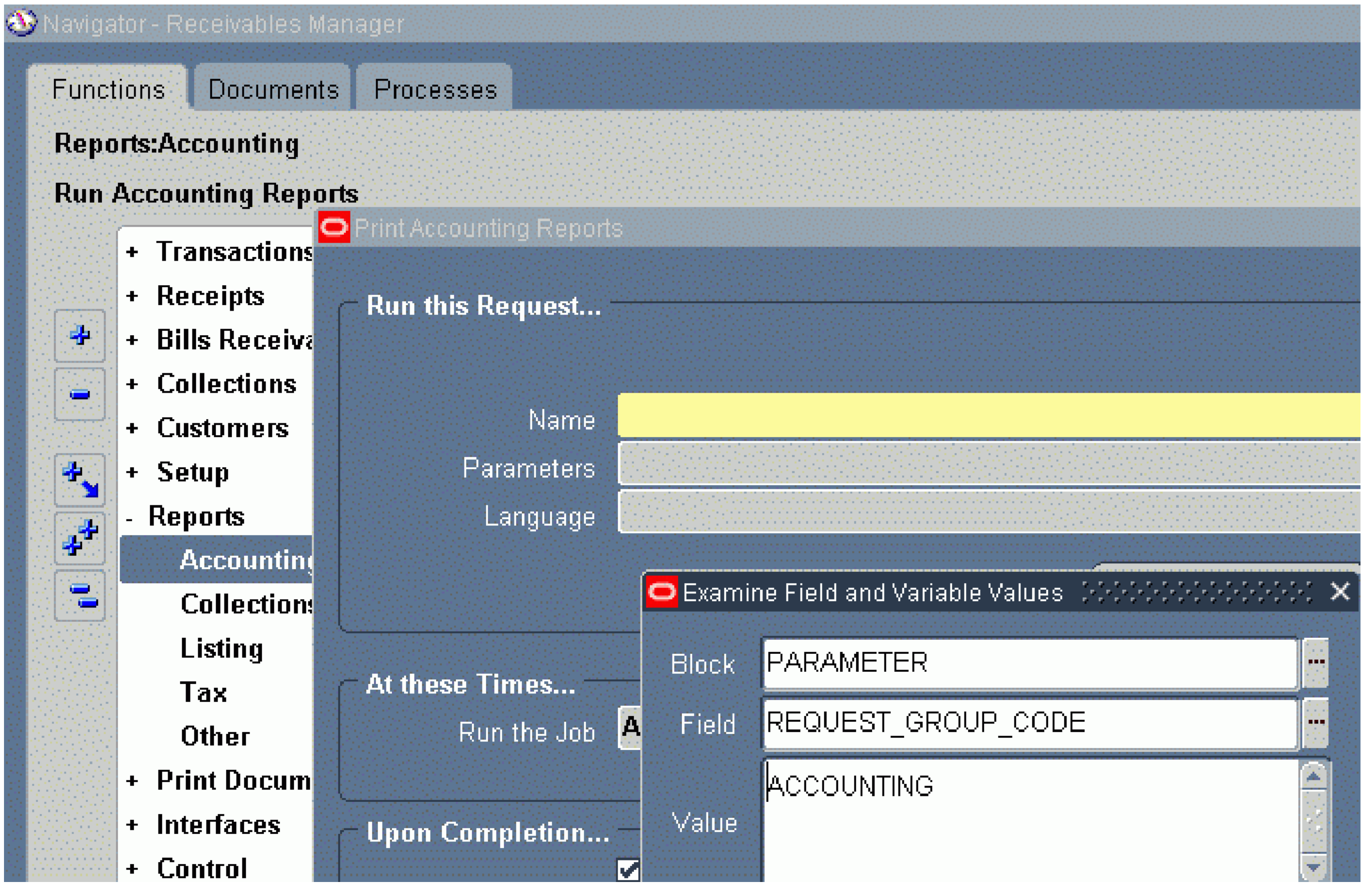
Step 3. Navigate to the "Submit New Request" screen, where you will enter the name of the concurrent program to run.

Step 4. Now you must do examine using the help menu as shown below.



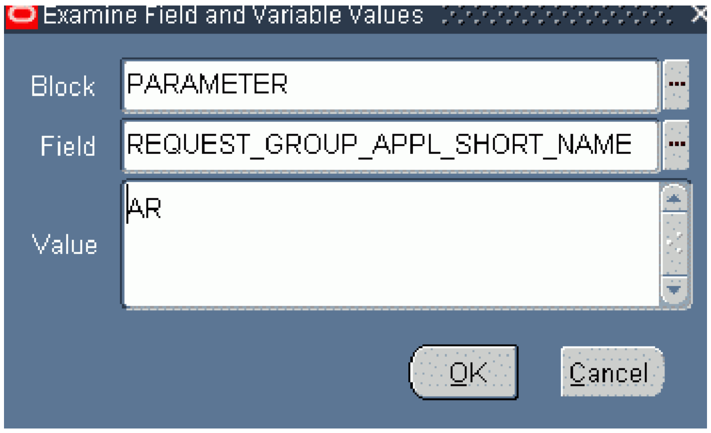
Help
Window Help
Oracle Applications Library
Keyboard Help...
Diagnostics
Record History
About Oracle Applications...
Disable Secured Diagnostics
Display Database Error...
Examine...

Step 5. In the block field, enter PARAMETER, and in the field name enter REQUEST_GROUP_CODE



Block: PARAMETER
Field: REQUEST_GROUP_CODE
Value: ACCOUNTING

Note down this value
Step 6. Now in the value field enter REQUEST_GROUP_APPL_SHORT_NAME



Block: PARAMETER

Field: REQUEST_GROUP_APPL_SHORT_NAME

Value: AR

Note down the two values from Step 5 and Step 6.

These will indicate the request group to which you must add your concurrent program.

Question: In the submit request screen, REQUEST_GROUP_CODE is blank??

In this case use the Request group specified in "Define Responsibility Screen"

The rule is :-

If request group code is being passed in as parameter to Submit Request form function, then use that request group. Or else use the request group at responsibility level.

NOTES:-

Why do I need both the request group code and the request group application?

Request group name and its application is a composite primary key, hence you need the application name. For example "All Reports" request group can be found under every possible application.

Where does this request group parameter come from?

This request group name and application are defined as parameters to the form functions.

Is this form function attached to submit request form?

Correct

If at the time of defining form function, we do not pass request group parameter, then request group from responsibility is used?

Correct, because request group specified against responsibility is the default request group.

Procure to Pay cycle-
1) Requisition creation in PO
2) Purchase order creation (Manual or Automatic)
3) Receipt creation
4) Invoicing in AP
5) Payment in AP

RE: Explain P2P cycle and the accounts that r effected...

(P to P) cycle: There are different steps involved in P to P cycle they are `

S.No	**Step**	**Defined At**
1	Define items	Inventory
2	Enter item information	Purchase Order
3	Raise requisition	Purchase Order
4	Request For Quotation(RFQ) raised	Purchase Order
5	Quotation received	Purchase Order
6	Quote analysis	Purchase Order
7	Raise purchase order	Purchase Order
8	Receive materials	Inventory
9	Enter invoice	Accounts payable
10	Payment process	Accounts payable
11	Transfer to general ledger	General ledger

Generating a purchase order

Purchasing order can be generated in two ways

1) Auto generation

2) Procedural generation

Auto generation: The various step that are involved in auto generation of purchase order is as under

1) Click on Switch responsibility and select purchasing module and press enter

2) Click on Supply Base Suppliers, a new form opens

3) Enter Supplier name and click on sites, a new form opens

4) Enter all the required information and also see that Purchasing and Pay options are enabled

5) Click on contacts and enter the required information, save the form and close it.

6) Click on Suppliers list –to create a supplier list and add the suppliers that we have created.

7) Save the form and close it.

Creating a Requisition

1) Click on Requisition   Requisitions a new form opens
2) Enter all the required information and save.
3) The approve button is enabled .click on approve button a new form opens.
4) Enter the hierarchal information and click on OK and close the form.

Raising a Purchase order directly

1) Click on the 'Auto create' a new form opens enter the requisition number and click on 'Find'.

2) A new form opens. Check the item line and Select the document type as 'Standard po'   and click on 'Automatic' tab.

3) A new form opens and click on 'Create' tab.

4) A window with the message with your purchase order number is displayed click 'ok' on it

5) Now click on "tools" and then on "copy Document" a new form opens,

6) click on 'ok' tab. And a new window with purchase order appears.

7) A new form opens Click on Approve button on this form

8) Enter the required information (real time) and click on 'ok' (submit approval)

9) Click 'ok' on new window that appears

10) And now we can see the status as 'approved' on the purchase order form.

2) Procedural generation The various step that are involved in auto generation of purchase order is as under

1) Click on Switch responsibility and select purchasing module and press enter

2) Click on Supply Base   Suppliers, a new form opens

3) Enter Supplier name and click on sites, a new form opens

4) Enter all the required information and also see that Purchasing and Pay options are enabled

5) Click on contacts and enter the required information, save the form and close it.

6) Click on Suppliers list –to create a supplier list and add the suppliers that we have created.

7) Save the form and close it.

Creating a Requisition

1) Click on Requisition

2) Enter all the required information and save.

3) The approve button is enabled .click on approve button a new form opens.

4) Enter the hierarchal information and click on OK and close the form.

Raising A RFQ

1) A new form opens. Select the document type as RFQ and click on 'Automatic' tab.

2) A new form opens and clicks on 'Create' tab.

3) A new form opens change the status as "Active" and click on "Suppliers" tab a new from opens, enter the required information and save the form and close it.

4) Click on tools and select "copy document" a new from opens

5) Enter the required information and click on “OK” a new form opens

Approving the Quotation

1) A new from opens

2) Change as status as”active” click on “Type” and then on “approve a new form opens

3) Enter the require information and click on “ok” and a message with text ‘quotation has been approved’ will appear, now click on ‘ok’ and close the form

4) Now click on “tools” and then on “copy Document” a new form opens, click ‘ok’ on the document where your purchase order will be created with a number..

5) Click ‘ok’ on the document a new form opens

Raising a Purchase Order

1) A new form opens, Click on Approve button on this form

2) Enter the required information (real time)and click on ‘ok’(submit approval)

3) Click ‘ok’ on new window that appears

4) And now we can see the status as ‘approved’ on the purchase order form.

To view Summary of the purchase order

1) Close all the existing opened forms and click on ‘notification summary’ In the main menu.

2) A new window opens in the internet explorer

3) Enter the user name and password provided by the system admin and login

4) A new window opens with all the purchase order details.

5) Click on the exact purchase order which is created by you. this can be done only if you remember your purchase order number.

Releasing the Order

The step by step process of releasing the order is as under:

1) Click on Purchase Order Releases –A new form opens

2) Enter the Purchase Order number and other required information and click on ‘Save’

3) ‘Approve’ button gets enabled now, hence click on it.

4) A new form opens, enter the required information and click on ‘ok’

5) Now click on ‘ok’ tab that is seen on message window.

6) Now the status changes to ‘Approved’

7) Now go to ‘Notification Summary’ window

8) You can see the ‘Blanket Purchase release document’ approved.

9) Click on it to view the **detailed** summary.

Receiving the Order

The step by step process of receiving the order is as under:

1) Click on Receiving Receipts –A new form opens select the organisation and click on ‘ok’ simultaneously a new form opens

2) Enter the P.O number in that form and click on ‘find’

3) ‘Receipt Header’ form opens-click on new receipt radio button and close the form

4) A new form opens, check box the item that is available there and save the form. Close it.

5) Click on ‘receiving transactions’, a new form opens.

6) Enter the P.O number and click on ‘Find’, a new form opens.

7) Check box the item that is available and select ‘Stores’ as sub inventory .save and close the form.

Yes We can create a synony for a trigger like this

```
Create or replace trigger test before insert on emp
begin
dbms_output.put_line('Hi');
end;
```

Create the trigger in scott schema

and connect to apps schema and create a synonym like this

```
create public synonym test for scott.test;
```

or

```
create synonym test1 for scott.test;
```

Steps for your first pl/sql Concurrent Program in Oracle Apps

I think this topic is already covered partially in one of the previous training lesson[for concurrent programs], but I would like to touch base on this again.

Lets revisit some basics first

Question: What are the ingredients for a concurrent program?
Answer: A concurrent executable and a program attached to that executable.

Question: Will executable for a pl/sql concurrent program be a database stored procedure?
Answer: Yes, but in addition to above you can also register a procedure within a package as an executable. However, you can't make a Function to become the executable for a stored procedure.

Question: Does this stored procedure need to have some specific parameters, in order to become an executable of a concurrent program?
Answer: Yes, such procedure must have at least two parameters
(errbuff out VARCHAR2, retcode out NUMBER)

Question: Can we add additional parameters to such pl/sql procedures that happen to be Conc Prog Executables?
Answer: Sure you can, but those parameters must be defined after the first two parameters. Effectively I mean first two parameters must always be errbuff and retcode. The sequence of the remaining parameters must match with the sequence in which parameters are registered in define concurrent program-parameters window.

Question: Can those parameters be validated or will these parameters be free text?
Answer: These parameters can be attached to a value set, hence this will avoid users passing free text values.

Question: What are the possible things that a concurrent pl/sql program can do?

Answer: Firstly your stored procedure would have been created in apps. This concurrent program will connect to "apps schema" from where access to all the tabes across every module will be available.
You can do the following:-
1. Insert records in tables(usually interface or temp tables)
2. Update and delete records
3. Initiate workflows
4. Display messages in the output file or the log file of the concurrent program.
5. Make this concurrent program complete with status Error or Warning or Normal.

Question: Please give me an example of a pl/sql concurrent program in Oracle apps in real life?
Answer: Lets say you have an external application which is integrated with apps. Assume that it is Siebel application where the new customer records are created. Siebel is assumingly hosted on a Oracle database from where it has database links to your Oracle apps database.
In this case, siebel inserts sales order records into your custom staging tables.
You can then develop a concurrent process which will do the following:--------
Loop through the records in that staging table
Check if the customer already exists in Oracle AR TCA
If customer already exists, thencall the api to update existing customer
If this is a new customer, then update existing TCA Customer/Party record

Question: Ok, how do I do the above?
Answer: Find the steps below for doing so

Step 1
Connect xxschema/password
Create table xx_stage_siebel_customers (customer_Id integer, customer name varchar2(400));
Grant all on xx_stage_siebel_customers to apps ;

Step 2
Connect apps/apps_password
Create or replace synonym xx_stage_siebel_customers for xxschema.xx_stage_siebel_customers
;

Step 3 (again in apps schema)
Create or replace procedure xx_synch_siebel_cust (errbuff out varchar2, retcode out varchar2) is
n_ctr INTEGER := 0 ;
Begin
for p_rec in (select * from xx_synch_siebel_cust) LOOP
Select count(*) into n_ctr from hz_parties where party_number = p_rec.customer_number;
If n_ctr=0 then
Hz_party_create(pass appropriate parameters here).
Else
Hz_party_update(pass appropriate parameters here);
End if;
END LOOP ;
delete from xx_synch_siebel_cust ;
End xx_synch_siebel_cust

Step 4
Create concurrent program executable (for example of screenshot, visit link)

Step 5
Create concurrent program for that executable

Step 6

Add this concurrent program to request group

Now your program is ready to roll....

posted by Anil Passi at 4:19 PM

Comments on "Steps for your first pl/sql Concurrent Program in Oracle Apps"

Simon said ... (8:12 AM) :

Hi,

I understand it's possible to call a stored procedure with parameters, but how would one go about deciding whether the procedure had completed okay, or if an error(s) had occurred.

Is there a way of informing the user (or an administrator) that the routine they had run had just failed?

I've thought that there are a number of potential ways of doing this. An error table could be updated, a log file could be produced, workflow could maybe be called or can the concurrent process be updated to show that a failure occurred and what this was?

In Oracle Apps, is there a standard or preffered way of doing this?

I'm looking at the Transaction Interface. I can catch errors but how do I inform users (in a tidy way) what has occurred?

Thanks.

Anil Passi said ... (9:58 AM) :

Hi Simon

When you submit a concurrent program, you can click on button labeled "Option". Here you can enter the name of the people that will receive email notification when the Concurrent Process completes.

The way I have implemented this is, in following steps:-
1. Make the Concurrent program complete with Warning in case of errors during interface.
2. Set the option such that Users receive notification on completion of request(click on option button)
3. The users will then receive email notification while will state..Your process Request Id xxx completed with Warning. Please click on Link to view the output.

Note: In your pl/sql concurrent program you must use fnd_file.put_line(which => fnd_file.output, buff => p_msg);
This will show all messages in output of the concurrent request.

Simon said ... (9:48 AM) :

Hi Anil,

Thanks a lot. This worked perfectly. One thing I couldn't find though was step 1:
"Make the Concurrent program complete with Warning in case of errors during interface."

Where do I do this?

Thanks again.

Anil Passi said ... (9:54 AM) :

If errors were encountered in the pl/sql concurrent process, at the very end of the pl/sql procedure assign 1 to retcode. Or call a function as below

```
FUNCTION get_ret_code RETURN VARCHAR2 IS
CURSOR c_check IS
SELECT 'x'
FROM "your error table here"
WHERE fnd_request_id = fnd_global.conc_request_id;
p_check c_check%ROWTYPE;
BEGIN
OPEN c_check;
FETCH c_check
INTO p_check;
IF c_check%FOUND
THEN
CLOSE c_check;
RETURN '1';
END IF;
CLOSE c_check;
RETURN NULL;
END get_ret_code;


====MAin process=-====
declare
begin
...your code here...
retcode := get_ret_code;
EXCEPTION jandler here

END ;
```

Simon said ... (4:03 AM) :

Hi Anil,

Thanks again. This worked as you said. I played around with it and found that I could change the retcode to give different status results as follows:

0 - 'Normal'
1 - 'Warning'
2 - 'Error'

In the table fnd_concurrent_requests the status code was saved as follows:
'C' - 'Normal'
'G' - 'Warning'
'E' - 'Error'

I'm guessing that the codes: 0, 1 & 2 are stored in the app somewhere as constants. As opposed to setting a retcode = 0, 1 or 2 I'd like to reference the Oracle constants. Do you know where these are stored (if they are)?

Thanks Again.

Anil Passi said ... (4:20 AM) :

Run the below SQL...these codes are stored in tables.

```
SELECT distinct lookup_code, meaning
FROM fnd_lookup_values
WHERE lookup_type = 'CP_STATUS_CODE'
AND trunc(SYSDATE) BETWEEN nvl(start_date_active
,trunc(SYSDATE)) AND
nvl(end_date_active
,trunc(SYSDATE))
```

Simon said ... (6:48 AM) :

Is that the same for retcode. Is its values; 0, 1 and 2 also stored in a table or are they package constants?

Anil Passi said ... (7:15 AM) :

Hi Simon
This appears to be hardcoded. Have a look at package FND_CONC_PRIVATE_UTILS
It has the below text.....

```
execute immediate sql_string using out errbuf,out retcode, in step;

/* If retcode is not set, then we'll assume success, since
* no exception was raised. */

if retcode is null then
retcode := 0;
end if;
```

Thanks
Anil Passi

Anonymous said ... (9:06 AM) :

Thanks , this blog really helped me

Anonymous said ... (1:02 AM) :

Hello Anil...

This site very useful for starters.I just moved into oracle application...I have a question ..

can u give me some common validation to validate data coming from legacy system on all major module.so it can be also helpful to learners

Thanks
Raj

Anonymous said ... (2:04 PM) :

Hi Anil,

This site is indeed incredible.. It shows the heights of efforts a person can put, in sharing knowledge and helping the newbies..Hats off for your hard work..

I have a question regarding concurrent program in apps.

I have a shell script which is registered as a host program.In that shell script I have a sql session too which uses fnd_global.apps_initialize(apps_user_id,apps_resp_id,application_id).

Now I have hardcoded the values of parameters for it. I am wondering if I could pass the parameters as variables(positional variables).

I think this will be needed when the program will be moved to other instances.

Could you please help me?

Thanks in advance

Anil Passi said ... (2:23 PM) :

Hi Anonymous,

Thanks for kind words.

The parameters passed by concurrent manager to a host program are
$0 - Shellscript name
$1 - Oracle username/password
$2 - User_id
$3 - Oracle Applications username
$4 - Concurrent request id

Please see metalink note Note:29086.1 for details.

It appears that you do have to hardcode some of the ID's if you wish to instantiate SQL*Plus Session.

Instead, why dont you write a Java Concurrent Program?

Thanks,
Anil Passi

Anonymous said ... (3:01 PM) :

Thanks for the immediate reply Anil,

As the user_id is passed as $2,as you said, can I pass that as $user_id in place of the parameter for the fnd_global.apps_initialize. Also can i include more positional parameters to use the resp_id and application_id;

for example;

shellscript_name= $1
user_id = $2
..
..
resp_id =$5
application_id = $6

and then in the sql session

fnd_global.apps_initialize($user_id,$resp_id,$application_id)

I have never used a java program, thats why I am proceeeding in this direction..

I made much effort to complete this request and registered as host program, everything is working fine.. but I am concerned just about this part.

Thanks again

Anil Passi said ... (3:44 PM) :

OK, I understand.

Please try the solution below:-

1. Create Parameters to your host concurrent program
2. These parameters will be ApplicationId and ResponsibilityId etc.
3. Assign these default values of type SQL Statement. The SQL Statement will be select fnd_global.resp_id etc.
5. In your hot program, access these are $5, $6 etc

Thanks,
Anil Passi

Anonymous said ... (12:08 PM) :

I have a question on passing parameter to a report which is registerd as a concurrent program.

I have a parameter "Operating unit".I have used the below part in the query of the report to enable this,

AND hru.organization_id = :P_OPERATING_UNIT

When the concurrent request for the report is submitted I want the parameter to act in such a way that, if the user gives null value, then the report must show all the results for all the operating unit, How can I make this possible? Now in such a situation, the report is not pulling any rows.

Anyone please help,

Thanks in advance.
Anita

Anonymous said ... (11:41 AM) :

Hi Anil,

Great blog!!

I have a question regarding a report submitted as a concurrent program.

I want to display "No Data Found" when the report output returns no rows.. How can this be made possible?

Thanks in advance.

Anonymous said ... (5:04 AM) :

Hi,
Please use the lexical parameter istead of bind parameters...
read something that give you good idea on lexical....

And regarding no data found to be displayed on the output...we can put a label called no_data_found in the layout and use the format trigger on it....thats it..

Thanks,
Kittu.

ORG_ID and Multi Org In Oracle Apps

Why do we need org_id
In any global company, there will be different company locations that are autonomous in their back office operations. For example, lets take the *example of a gaming company named GameGold Inc* that has operations in both UK and France.

Please note the following carefully:-
1. This company(GameGold Inc) has offices in both London and Paris
2. UK has different taxation rules that France, and hence different tax codes are defined for these countries.
3.GameGold Inc has implemented Oracle Apps in single instance(one common Oracle Apps database for both UK & France).
4. When "UK order entry" or "UK Payables" user logs into Oracle Apps, they do not wish to see tax codes for their French sister company. This is important because French tax codes are not applicable to UK business.
5. Given the single database instance of GameGold Inc, there is just one table that holds list of taxes. Lets assume that the name of the Oracle table is ap_tax_codes_all
6. Lets assume there are two records in this table.
Record 1 tax code -"FRVAT"
Record 2 tax code - "UKVAT"
7. Lets further assume that there are two responsibilities
Responsibility 1 - "French order entry".
Responsibility 2 - "UK order entry"
8. Now, users in France are assigned responsibility 1 - "French order entry"
9. Users in UK will be using responsibility named "UK order entry"
10. In the Order Entry screen, there is a field named Tax Code(or VAT Code).
11. To the French user, from the vat field in screen, in list of values UKVAT must not be visible.
12. Also, the "French order entry" user should only be able to select "FRVAT" in the tax field.
13. Similarly, UK order entry user, that uses responsibility "UK Order Entry", only "UKVAT" should be pickable.

How can all this be achieved, without any hard coding in the screen.
Well....the answer is org_id

ORG_ID/Multi-Org/Operating Unit are the terminologies that get used interchangeably.

In brief steps, first the setup required to support this....
The screenshots are at the bottom of the article
1. You will be defining two organizations in apps named "French operations" and "UK Operations". This can be done by using organization definition screen.
2. In Oracle Apps, an organization can be classified as HRMS Org, or Inventory Warehouse Org, or Business Group, Operating Unit Org or much more types. Remember, Organization type is just a mean of tagging a flag to an organization definition.
3. The two organizations we define here will be of type operating unit. I will be using words org_Id

& operating unit interchangeably.
4. Lets say, uk org has an internal organization_I'd =101
And french org has orgid =102.

Qns: How will you establish a relation betwee uk responsibility and uk organization.
Ans: By setting profile option MO : Operating unit to a value of UK Org, against uk order entry responsibility

Qns: How will the system know that UKVAT belongs to uk org?
Ans: In VAT code entry screen(where Tax Codes will be entered), following insert will be done
Insert into ap_vat_codes_all values(:screenblock.vatfield, fnd_profile.value('org_id').
Alternately, use USERENV('CLIENT_INFO')

Next question, when displaying VAT Codes in LOV, will oracle do: select * from ap_vat_codes_all where org_id=fnd_profile.value('ORG_ID')?
Answer: almost yes.

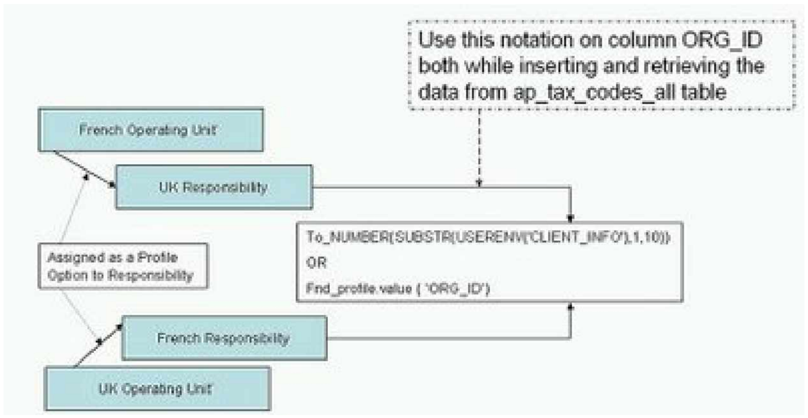
Oracle will do the following
1. At the tme of inserting data into multi-org table, it will do insert into (vatcode,org_id) ....
2. Creates a view in apps as below
Create or replace view ap_vat_codes as Select * from ap_vat_codes_all where org_id = fnd_profile.value('ORG_ID')
3. In the lov, select * from ap_vat_codes ,

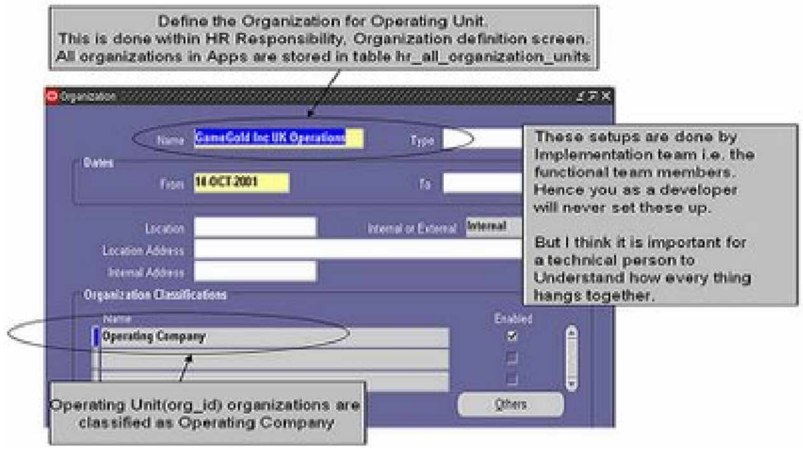
If the above doesn't make sense, then keep reading.

May be quick revesion is necessary:_
1. In multi org environment(like uk + france in one db), each Multi-Org Table will have a column named org_id. Tables like invoices are org sensitive, because UK has no purpose to see and modify french invoices. Hence a invoice table is a candidate for ORG_ID column.
By doing so, UK Responsibities will filter just UK Invoices. This is possible because in Apps, Invoice screens will use ap_invoices in their query and not AP_INVOICES_ALL.
2. Vendor Sites/Locations are partitined too, because UK will place its ordersfrom dell.co.uk whereas france will raise orders from dell.co.fr. These are called vendor sites in Oracle Terminology.
3. Any table that is mutli-org (has column named org_id), then such table name will end with _all
4. For each _all table , Oracle provides a correspondong view without _all. For examples *create or replace view xx_invoices as select * from xx_invoices_all where org_id=fnd _profile.value('org_id').*
5. At the time of inserting records in such table, org_id column will always be populated.
6. If you ever wish to report across all operating units, then select from _all table.
7. _all object in APPS will be a synonym to the corresponding _all table in actual schema. For example po_headers_all in apps schema is a synonym for po_headers_all in PO schema.
8. When you connect to SQL*Plus do the below
connect apps/apps@dbapps ;
--assuming 101 is French Org Id
execute dbms_application_info.set_client_info (101);
select tax_code from ap_tax_codes ;
---Returns FRVAT

--assuming 102 is UKOrg Id
execute dbms_application_info.set_client_info (102);
select tax_code from ap_tax_codes ;
---Returns UKVAT

Now some screenshots


Use this notation on column ORG_ID both while inserting and retrieving the data from ap_tax_codes_all table
French Operating Unit'
UK Responsibility
Assigned as a Profile Option to Responsibility
To_NUMBER(SUBSTR(USERENV('CLIENT_INFO'),1,10))
OR
Fnd_profile.value ('ORG_ID')
French Responsibility
UK Operating Unit'


Define the Organization for Operating Unit.
This is done within HR Responsibility, Organization definition screen.
All organizations in Apps are stored in table hr_all_organization_units

These setups are done by Implementation team i.e. the functional team members. Hence you as a developer will never set these up.

But I think it is important for a technical person to Understand how every thing hangs together.

Operating Unit(org_id) organizations are classified as Operating Company

Organization

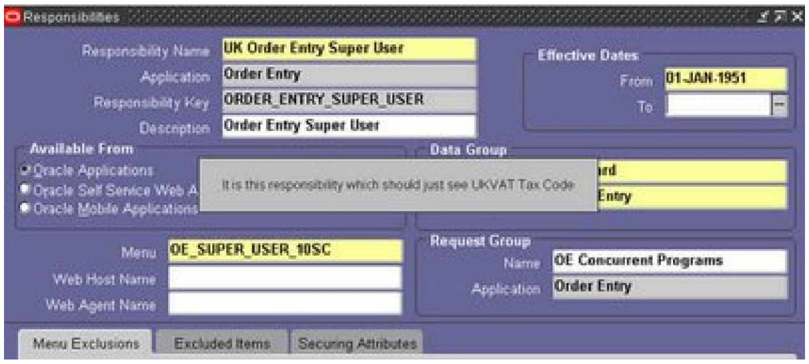
Name: GameGold Inc UK Operations Type:

Dates
From: 14-OCT-2001 To:

Location: Internal or External: Internal
Location Address:
Internal Address:

Organization Classifications

Name	Enabled
Operating Company	✓


Responsibilities

Responsibility Name: UK Order Entry Super User
Application: Order Entry
Responsibility Key: ORDER_ENTRY_SUPER_USER
Description: Order Entry Super User

Effective Dates
From: 01-JAN-1951
To:

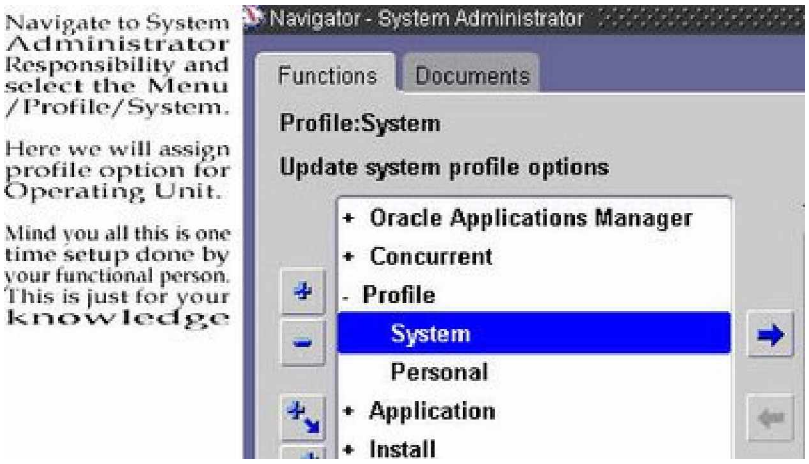
Available From
(● Oracle Applications) Oracle Self Service Web A / Oracle Mobile Applications

It is this responsibility which should just see UKVAT Tax Code

Data Group
rd
Entry

Menu: OE_SUPER_USER_10SC
Web Host Name:
Web Agent Name:

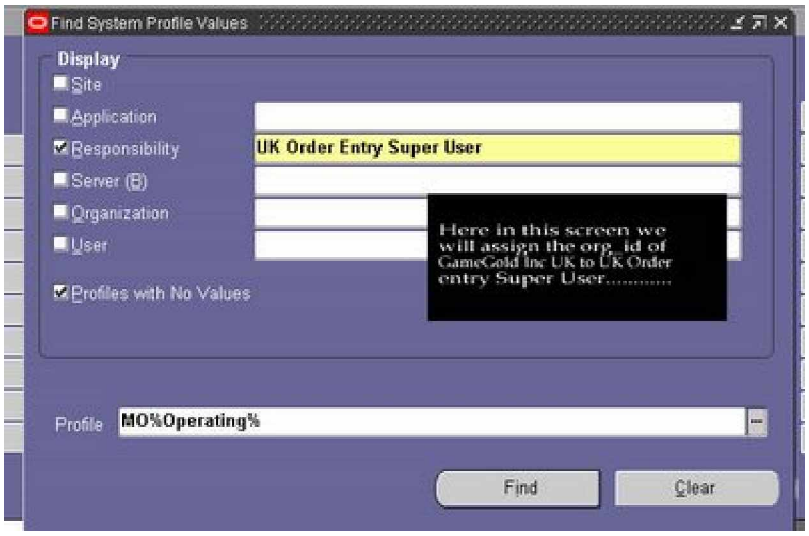
Request Group
Name: OE Concurrent Programs
Application: Order Entry


Navigate to System Administrator Responsibility and select the Menu /Profile/System.

Here we will assign profile option for Operating Unit.

Mind you all this is one time setup done by your functional person. This is just for your knowledge

Navigator - System Administrator
Functions
Documents
Profile:System
Update system profile options
+ Oracle Applications Manager
+ Concurrent
- Profile
System
Personal
+ Application
+ Install

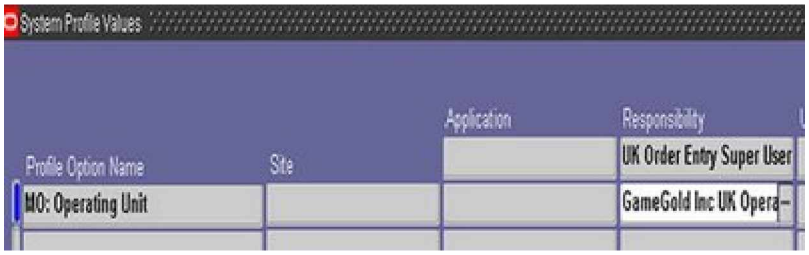


Responsibility [✓]: UK Order Entry Super User

Profiles with No Values [✓]

Profile: MO%Operating%

Here in this screen we will assign the org_id of GameGold Inc UK to UK Order entry Super User............



Profile Option Name	Site	Application	Responsibility
			UK Order Entry Super User
MO: Operating Unit			GameGold Inc UK Opera

Thanks,
Anil Passi

posted by Anil Passi at 12:14 AM

Comments on "ORG_ID and Multi Org In Oracle Apps"

Anonymous said ... (2:51 AM) :

Hi anil,

As we know org_id is all about the operating unit and organization_id is about the inventory org.How about the HR org.In HR tables organization_id is represents what?

Anil Passi said ... (2:54 AM) :

Hi Anonymous,

Whether it is an ORG_ID or Inventory Organization Id or HR Organization, all these Organizations are stored in one single table.
That table is named HR_ALL_ORGANIZATION_UNITS.

select organization_id, name from HR_ALL_ORGANIZATION_UNITS ;

Now, each such Organization record in HR_ALL_ORGANIZATION_UNITS can be assigned Classifications.

One snigle Organization can be assigned multiple classifications if required.

For example, if you attach it a classification of "Operating Unit", then it can be used as a ORG_ID, by the virtue of profile option "MO: Operating Unit"

If you attach that Organization record in
HR_ALL_ORGANIZATION_UNITS a classification of "HR Organization", then it becomes HR Org.

You can assign a classification of "Inventory Organization", hence flagging it as Inventory Warehouse.

Other possible classifications are
Business Group
GRE / Legal Entity
Project Expenditure/Event Organization
Project Invoice Collection Organization
Project Task Owning Organization
Asset Organization
MRP Organization

There are several more, to find the complete list fo classifications run below SQL
```
SELECT hl.meaning, hl.lookup_code
FROM hr_lookups hl
WHERE hl.lookup_type = 'ORG_CLASS'
AND hl.enabled_flag = 'Y'
AND trunc(SYSDATE) BETWEEN nvl(hl.start_date_active
,trunc(SYSDATE)) AND
nvl(hl.end_date_active
,trunc(SYSDATE))
ORDER BY meaning
```

Anonymous said ... (11:01 PM) :

Anil,
I am new to Oracle apps and trying to learn as much as I can. I like your blog and learn a lot from it.
I understand (I think) the value of org_id in filtering the result.
I need, however to be able to run a query for ALL org_id(s) in my organization without setting it as: begin fnd_client_info.set_org_context("org_id"). Is it possible? If so how? Is there a table that has all defined org_ids that I can use to loop throught, possibly?
Thanks,
Eugene

Anil Passi said ... (1:09 AM) :

Hi Eugene

Simply run your query using _ALL table. I mean, use the table which ends with _ALL

By doing so, you will be retrievnig data straight from the tables.

Thanks,
Anil

Smrati Saxena said ... (3:45 AM) :

Hi Anil,
Awesome Information.......
Really appreciate your effort of sharing knowledge with all of us.
I would like to know the difference and relationship between org_id and organization_id.
As when we write queries for report, it realy confuses us that waht stands for what...and as i am new to apps..donn have much knwledge of all these..
Thanks in advace,
Smrati

Anonymous said ... (3:46 AM) :

Hi Anil,
Awesome Information.......
Really appreciate your effort of sharing knowledge with all of us.
I would like to know the difference and relationship between org_id and organization_id.
As when we write queries for report, it realy confuses us that waht stands for what...and as i am new to apps..donn have much knwledge of all these..
Thanks in advace,
Smrati

Anil Passi said ... (5:37 AM) :

Hi Smrati,

organization_id is the Unique Id of the Organization that you have defined.

Lets take an example,
1. Your company name is Smrati Inc
2. Smrati inc has various departments like IT, Payables, Purchasing Dept etc.

In such case, you will define 4 records in hr_all_organization_units table
record 1:- smrati inc [id=101]
record 2:-payables dept [id=102]
record 3:-it dept [id=103]
record 4:-purchasing dept [id=104]

However, only Smrati Inc will be registered as an operating unit in this case.

Hence your org_id will be 101, i.e. hr_all_organization_units.organization_id of Smrati Inc.

Thanks,
Anil Passi

Anonymous said ... (10:44 PM) :

Hi Anil,
Great Thanks for the explanation!!..
However i still have one doubt...
There must be so many Operating Units in an organization...so how to find which all inventory organizations comes under which operating unit..
Like in ur example..where this relationship is defined that 102,103 and 104 comes under 101..
Appreciate if u can explin bit more on this..
Thanks..

Anonymous said ... (11:30 AM) :

Anil,

I have been working as an apps developer for a year now...I find your blog is full of real life situations which an apps consultant faces everyday. I appreciate the service you are doing..Kudos to you!!

Imran Khan

Anonymous said ... (9:28 AM) :

Hi Anil,

After reading multi org structure, I am struggling to co-relate this to real life scenario. I mean if I have a

Global Compnay called ABC and this Compnay now want to implement Oracle Apps. So for this Company which all will be the probable Business Group, Legal Entity, Balancing Entity,Operating Unit, Inventory Org

Anil Passi said ... (9:38 AM) :

Think of these as below:-

1. Inventory:- Each of your physical warehouse
2. Operating Unit :- Think of this units within different taxation rules, accounting rules
3. Business Group
Think of it, Mittal Steel and Arcelor have merged. If you were to integrate their systems, you will have two different business groups.
4. Mittal Steel UK and Arcelor steel UK will both belong to same Legal Entity

Anonymous said ... (6:17 PM) :

Anil, Excellent write ups. Keep up the good work...

Gayathri said ... (4:11 AM) :

Hi Anil

Excellent way in which u explain things in teh simplest way!!

THat makes a lot of difference to people who read the article.

I appreciate ur dedication and knowledge sharing attitude

Regards
Gayathri

Sanjay said ... (12:52 AM) :

Hi Anil

After having a look at the topic. I am very clear with multi-org situation. I have understood the relation between ORGANISATION_ID and BUSUNESS_GROUP_ID as you have explained to Smrati in your comment section.
I have a doubt Anil, I just want to clear that what is the difference/Similarity between ORG_ID and ORGANISATION_ID?

Anil Passi said ... (4:16 PM) :

to simply explain, org_id is the organization_id of the operating unit.

Ah, now lets explain this.

You have three records in hr_all_organization_units
1. Reliance Petrochemical
2. Reliance Petro - Finance Deparment
3. Reliance Petro - Sales Deparment

In this case, lets assume the organization_id of these three records are 1000,1001,1002

In this case, lets say "Reliance Petrochemical" is defined as MultiOrg Operating Unit.

Hence, all the tables that have column org_id, will have a value of 1000 in org_id column, assuming the transaction was created for "Reliance Petrochemical"

another way, just translate column org_id to organization_id.
Think that there is no such thing as org_id. In your mind replace each occurance of org_id with organization_id, for which the master table is hr_all_organization_units

thanks,
anil

Anonymous said ... (2:32 AM) :

Anil,

Excellent job !!!!

Thanks for sharing your knowldge...

Saran

vemula said ... (3:41 AM) :

Hi Anil,
Iam a fresher to Oracle apps.But you made the "operating unit" concept very easy.Always think of freshers and post your articles. This makes the concept very clear and complete.
Thanks,
Surya.

vivek said ... (2:35 AM) :

Hi Anil,

Wonderful Job !!
Keep it up.

Thanks
Vivek

reddy said ... (8:20 AM) :

Hi Anil,
Now i am very clear about ORG_ID and multi org in oracle apps.
Very much thank you.
Here i have one doubt.

For each _all table , Oracle provides a correspondong view without _all. For examples create or replace view xx_invoices_all as select * from xx_invoices_all where org_id=fnd _profile.value('org_id').

In this explanation the view should not have _all but in your example the view is contain _all.
Could you please clarify this.
Thanks in advance.

Anonymous said ... (8:21 AM) :

Hi Anil,
Now i am very clear about ORG_ID and multi org in oracle apps.
Very much thank you.
Here i have one doubt.

For each _all table , Oracle provides a correspondong view without _all. For examples create or replace view xx_invoices_all as select * from xx_invoices_all where org_id=fnd _profile.value('org_id').

In this explanation the view should not have _all but in your example the view is contain _all.
Could you please clarify this.
Thanks in advance.

Anil Passi said ... (9:22 AM) :

You are right, sorry for the typo

It should be
create or replace view xx_invoices as select * from xx_invoices_all where org_id=fnd _profile.value('org_id')

Thanks
anil

Anonymous said ... (1:28 AM) :

Hi Anil,

Thanks for your effort of sharing knowledge.

Could you please explain why Oracle provide so many types of ORG.

Rgds
NK

subrat said ... (9:35 AM) :

Appreciate all your valuable topics.It helps a lot lot....

Best regards
Subrat

Anonymous said ... (11:24 AM) :

Can u explain me the concept of multi-org?

Anil Passi said ... (1:07 PM) :

On olden days, when Oracle ERP was launched, each Operating Unit within a Global company had to implement their own Oracle ERP.

This was done because we had one installation per organization.

Latter in Mid-1990s Oracle introduced Multi-org model , which has been explained in this article.
Each Organization specific transaction is attached an Operating Unit. This is done by means of OrgId, which is explained in this article.

Please feel free to ask any questions on this if anything is unclear.

Thanks,
Anil Passi

shankar said ... (12:20 AM) :

hi anil,
first i would like to thank you for your wonderful job.this is my doubt.
with reference to your 'reliance petrochemical ' - example , do you mean to say that

entries for org_id (column in other tables) will be found only if a organization is declared as operating unit?
or is it just a alias name given to the organization_id in the case of operating unit?
can you shed more light on the 'reliance example' as to what exactly happens in the back end (along with the column names and table names)?
thanx in advance.

Anand said ... (2:41 AM) :

Hi anil,
I like the way you share your knowledge.I would like to leave a suggestion here.
All your explaination are terrific.
my suggestion is this"Will it be better if you leave a HYPER LINK to the screenshot right adjacent to the explaination instead of screenshots at the bottom as it might reach the dumb people like me very easily".

Thanks and Regards
Anand.R

Anil Passi said ... (2:47 AM) :

You have a valid point, but I disagree with the "dumb" remark.

Thanks for your valuable suggestion

Anil

Anonymous said ... (12:30 AM) :

Hi Anil,

You are excellent in explaining the concept of multi org.

But i have a small doubt regarding the multi-org.

When an installation can be said that it has got multi org structure?????...as of now i am thinking that if an installation has got more than 1 bussiness group only multiorg structure is available...is that true????????

Shaik said ... (5:11 AM) :

Hi, Anil,

Just today i have seen ur blog and the answers u have given to questions,
Ur excellent in explaning the things to the ground level that a lay man gets understand easily,

thanks a lot.,
Shaik Ghouse

shireen said ... (5:41 AM) :

Hi Anil,

Your site is really helpful, It made the concept of procure to pay n order to cash very easy..which was otherwise very difficult..I am planning to take up oracle financials fundamental exam,I have queries on workflow, can u please tell explain what are business event system functions ??n is the workflow architecture same as business event system architecture??

Anil Passi said ... (10:38 AM) :

Hi Shireen

The Business events engine is a part of workflow engine, as said by Oracle. However for all practical reasons you can think of Business Events to be a totally different entity that Workflows.

I have some examples on Business Events on http://oracle.anilpassi.com

Thanks,
Anil Passi

Tejprakash said ... (8:51 AM) :

Hi Anil, Great Explanations. I do have a query? What would be the scene if the Organization is Defined as GRE/Legal Entity,OU,& Inventory Organization at the same time.What will be the Org_id & Organization_id in that scenario.
Thanks
Tej Prakash Tiwari

Anil Passi said ... (8:55 AM) :

Hi Tej

The org_id and organization_id will still be the same, because there will be just one record in hr_all_organization_units table.
but this record will be a foreign key to multiple classifications

Thanks
Anil

Anonymous said ... (12:16 PM) :

Hi Anil,

Thanks for this an other wonderful post from your cap i was able to understand the org_id aka multiorg concept except for one thing ie., what exactly does this condition **'where org_id = fnd_profile.value('ORG_ID')'**.

Thanks and i appreciate your help,
Jay.

Anil Passi said ... (12:57 PM) :

Hi Jay

Lets do some further Q&A

The purpose of multi org view?

The purpose of multi-org view is to partition the data based upon the value of current org_id.

How does this partitioning work?

The MO view has a where clause that filters the records relevant to to the current org_id

How does Oracle know the current Org Id

Org Id is setup as a profile option. Short name of profile option is ORG_ID. To get the current profile value assigned to the user, simply do fnd_profile.value('ORG_ID')

Note: A client info session variable named org_id is also populated by Oracle, within a package procedure named fnd_global.initialize

When is fnd_global.initialize called?
This is called as soon as a user logs onto a responsibility

Thanks,
Anil Passi

Anonymous said ... (6:30 PM) :

Thanks so much for the earlier reply..I am a fresher, I have completed functional training recently, I am an MBA(Finance and marketing) with graduation in B.E(Instrumentation and electroncs), I have completed financials functional training, I have the knowldege of c-language, SQL/PL SQL.In the same institute where i underwent functional training are offering 2months of technical training, i would get a good discount if i join it.Please suggest me if it will be useful if join it as a functional consultant...

Anil Passi said ... (1:00 AM) :

It appears you wish to be Functional consultant. If so, then basic SQL knowledge is enough for you to carry out your functional duties in project. After having done MBA, there is no much value for you to go back to technical programming job. Hence you do not require Apps technical training.

Anyway, I did my BE in instrumentation and process control too.

Thanks,
Anil

Anonymous said ... (7:29 AM) :

Hi Anil, Thank you so much for suggesting me , U got it right that i am interested in Fucnctional.And you saying there is no value in going back to technical has given me very good justification to be in functional...thanks again..

Anonymous said ... (7:57 AM) :

Hi!Could you please tell me how important is oracle certification for freshers...and how to prepare best for the exams..

Anil Passi said ... (8:20 AM) :

Certification increases your probability of getting the job, surely.

As this shows your prospective employer
1. You are commited to the cause of learning
2. you add to the value of the organization and having a lot of certified professionals will them win more projects

What is the subject matter you wish to certify on?
Thanks
Anil

Anonymous said ... (3:31 AM) :

Certification in oracle 11i financials functional(In AP which includes three exams fundamental, Gl and AP)

Anonymous said ... (10:48 AM) :

hi!Please let me know how to prepare for the above exam...

Anil Passi said ... (11:32 AM) :
The best way to prepare is to run end to end processes in these respective modules.

The exam contents are listed on Oracle website

Please see
http://education.oracle.com/pls/web_prod-plq-dad/show_desc.redirect?redir_type=13&group_id=58&p_org_id=1080544 =US&media=1

and
http://www.oracle.com/global/us/education/certification/ebus11i_appteccourses.html

How will you set the MOAC org context in a OA Framework screen. Iam desigining a OA screen in R12 based on a view object. But it does not return any rows since the org id is not set.

Please let me know how do i need to set the org id ..
I think it can be done in
Controller of the page , but how ?

R12 MOAC should be picked up automatically from the MOAC Setup

Please read the below links for details to ensure that this has been setup on your system.

When OAF forms the WebAppsContext, the MOAC code will be executed too.

R12 Multi Org Basics
R12 Multi Org Details

stephen said ... (11:24 AM) :

Can two Operating Units within the same set of books have different key flex-fields for inventory items? Similar question for asset id's?

My understanding was that all KFF's and DFF's are shared but your lucid article makes me wonder if org_id enables non Accounting KFF's to be different for each org.

gandhi said ... (2:16 AM) :

Hi need help.
How can we find out the operating unit of a particular company (GL Code combinations segment1).
I'm looking for the table links.

Thanks a lot in advance for the help.

blesson said ... (12:30 PM) :

"Invoice screens will use ap_invoices in their query and not AP_INVOICES_ALL."

Could you please explain what is the difference bte ap_invoices and AP_INVOICES_ALL w.r.t above comments.

Anil Passi said ... (4:05 PM) :

In Oracle 11i, when a Payables user for GE-UK logs in, they must not be able to view invoices of GE-US.

AP_INVOICES_ALL will contain the invoices of all the organizations, i.e. GE-UK, GE-US etc.

Hence the screen queries ap_innvoices, which is a view on top of ap_invoices_all.

select * from ap_invoices
is equivalent to
select * from ap_invoices_all where org_id = fnd_profile.value('ORG_ID')

Thanks,
Anil Passi

SURYA said ... (5:02 AM) :

HI ANIL,
WHAT EXACT THIS CCID MEANS
AND HOW CAN WE DEFINE IT

Value set

http://getappstraining.blogspot.com
Question: What is value set?
Answer: It is a set of values

Question: Why do we need value sets?
Answer: You do not always want a user to enter junk free text into all the fields. Hence, Oracle Apps uses value set to validate that correct data is being entered in the fields in screen.

Question: Is value set attached to all the fields that require validations?
Answer : A big NO

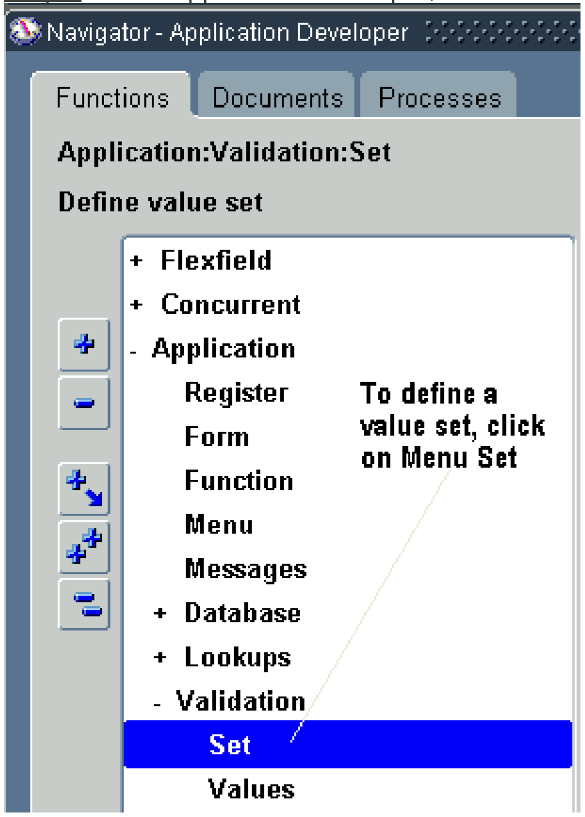
Question: Then where lies the usage of value sets?
Answer: Broadly speaking, value sets are attached to **segments in Flexfields**. You can argue that value sets are also attached to **parameters of concurrent program**(but in reality oracle treats parameters as Descriptive Flexfields)

Question: Any examples?
Answer: For the namesake, lets add a Parameter to the concurrent program that we defined in "Concurrent Program Trainin
Lesson link". Lets add a parameter named "cost centre", the values to this parameter must be restricted to one of the three values, i.e. HR, SEC, IT.

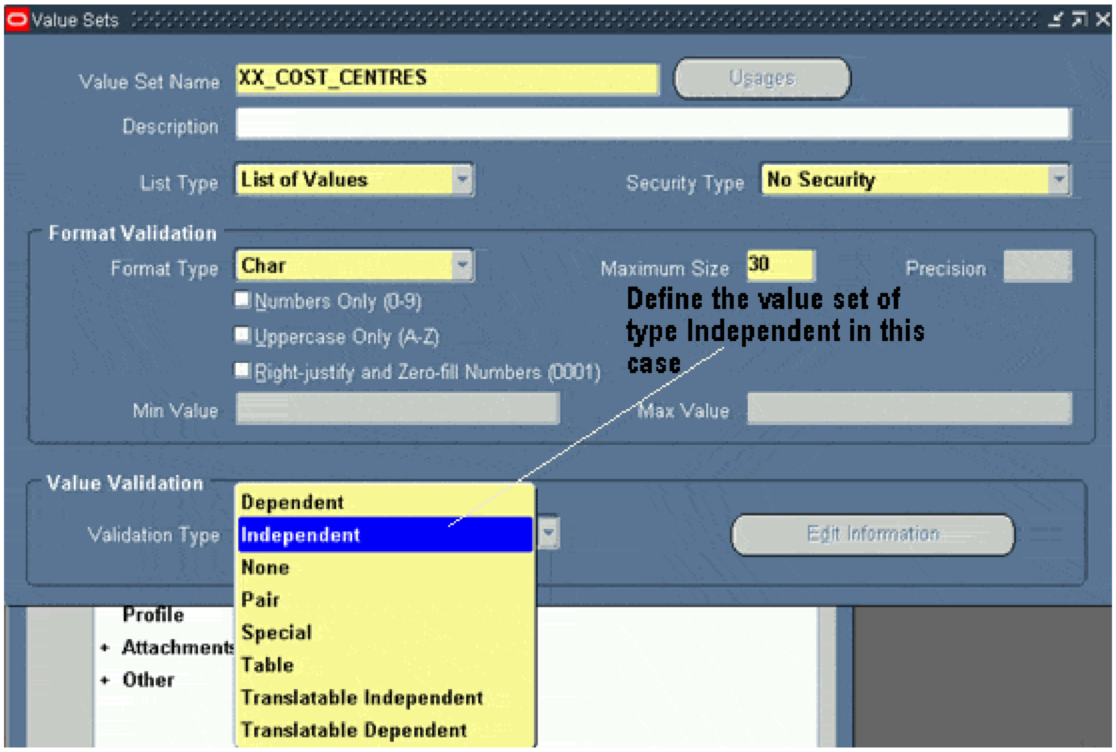
At the time of submission of the concurrent program the user should be able to pick a cost centre from a list. This is where value set gets used.

Lets now define a simple value set as described above.

Step 1. Go to Application Developer, and select menu /Validation/Set



Step 2. Now define a value set of type Independent. We will cover the other most widely used Type "Table" latter.

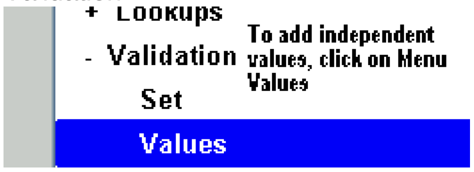


Value Set Name: XX_COST_CENTRES
Description:
List Type: List of Values
Security Type: No Security
Format Type: Char
Maximum Size: 30
Precision:
☐ Numbers Only (0-9)
☐ Uppercase Only (A-Z)
☐ Right-justify and Zero-fill Numbers (0001)
Min Value:
Max Value:
Validation Type: Independent

Dependent
Independent
None
Pair
Special
Table
Translatable Independent
Translatable Dependent

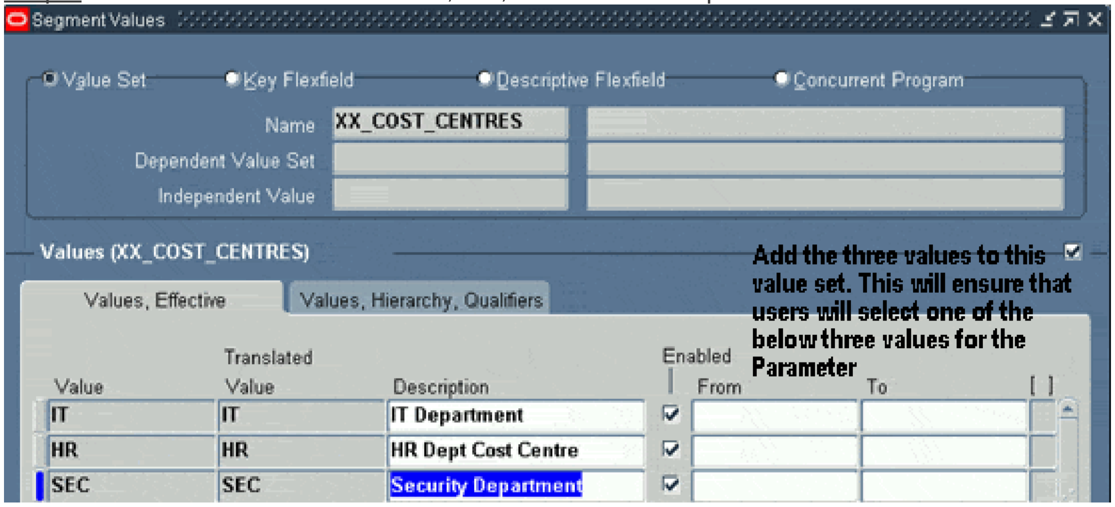
Define the value set of type Independent in this case

Step 3. Now, lets add three independent values to the value set for this Cost Centre list. Hence click on menu Values within Validation



To add independent values, click on Menu Values

Step 4. Here we add the values for IT, HR, SEC to this independent value set.



Name: XX_COST_CENTRES
Dependent Value Set:
Independent Value:

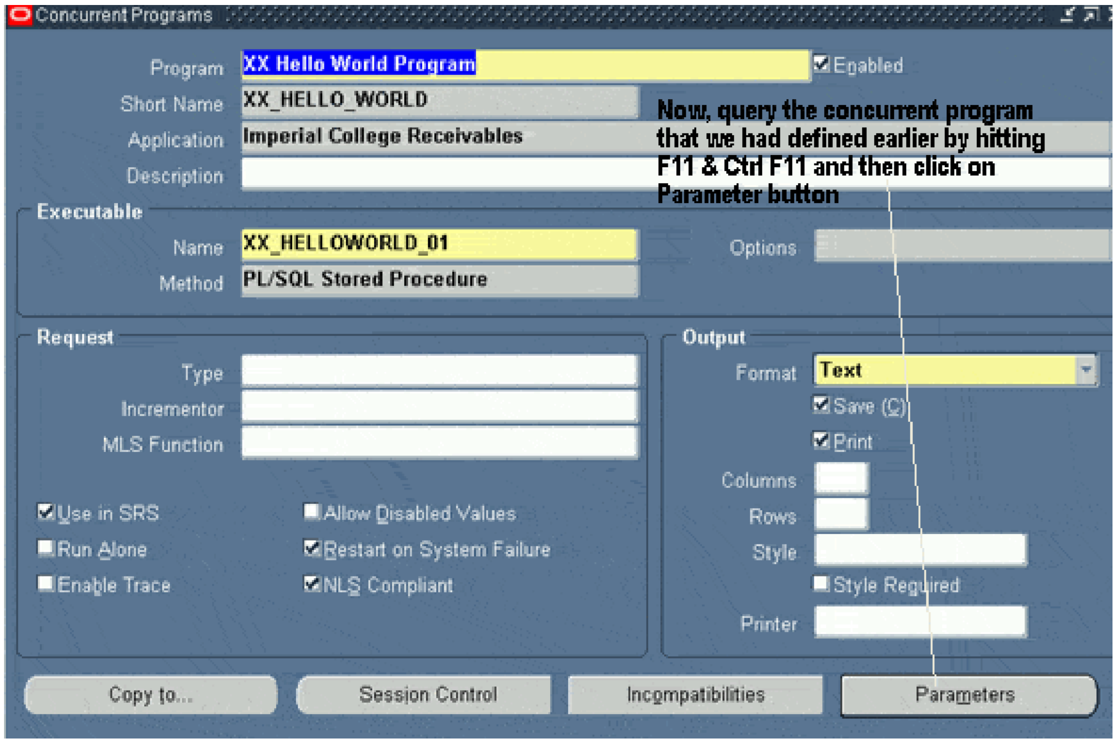
Values (XX_COST_CENTRES)

Value	Translated Value	Description	Enabled	From	To
IT	IT	IT Department	✓		
HR	HR	HR Dept Cost Centre	✓		
SEC	SEC	Security Department	✓		

Add the three values to this value set. This will ensure that users will select one of the below three values for the Parameter

“CONTROL-S” to save the data

Step 5. Now let us go back to Concurrent Program that we created in earlier training lesson and Click on Parameters


Concurrent Programs

Program: XX Hello World Program ☑ Enabled

Short Name: XX_HELLO_WORLD

Application: Imperial College Receivables

Description:

Executable

Name: XX_HELLOWORLD_01

Method: PL/SQL Stored Procedure

Options:

Request

Type:

Incrementor:

MLS Function:

☑ Use in SRS ☐ Allow Disabled Values

☐ Run Alone ☑ Restart on System Failure

☐ Enable Trace ☑ NLS Compliant

Output

Format: Text

☑ Save (C)

☑ Print

Columns:

Rows:

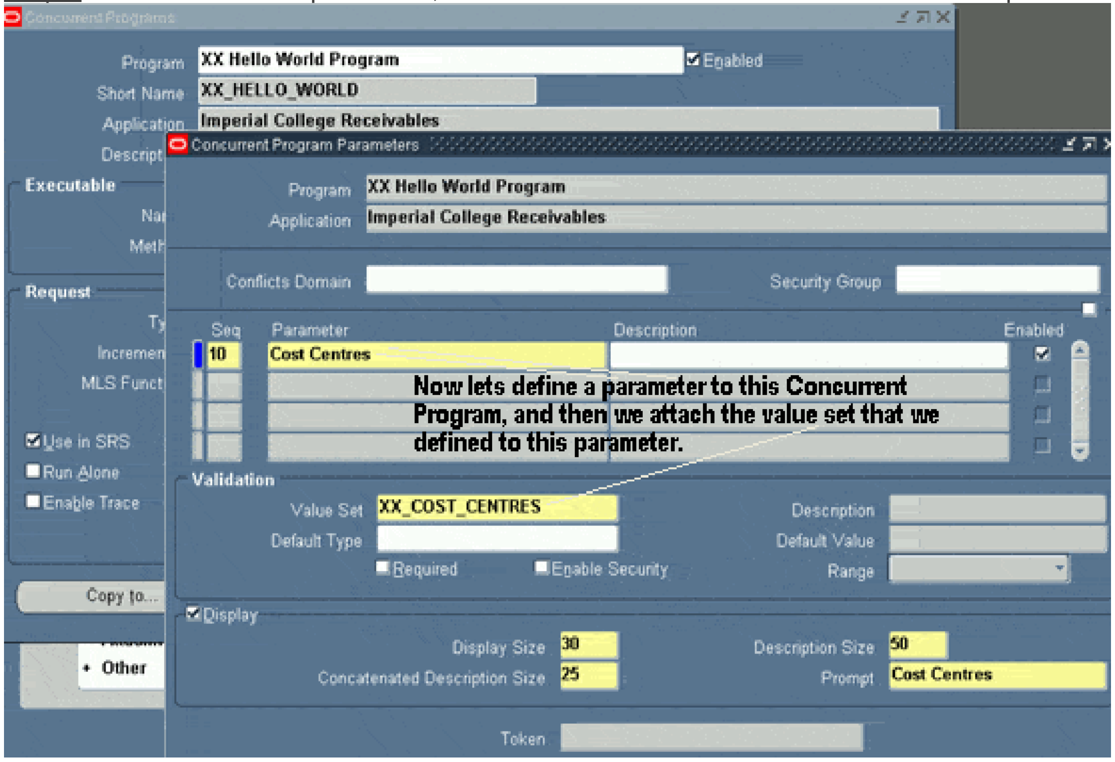
Style:

☐ Style Required

Printer:

Now, query the concurrent program that we had defined earlier by hitting F11 & Ctrl F11 and then click on Parameter button

Step 6. Now lets create a parameter, and attach the value set that we created to this parameter.


Concurrent Programs

Program: XX Hello World Program ☑ Enabled

Short Name: XX_HELLO_WORLD

Application: Imperial College Receivables

Concurrent Program Parameters

Program: XX Hello World Program

Application: Imperial College Receivables

Conflicts Domain: Security Group:

Seq	Parameter	Description	Enabled
10	Cost Centres		☑

Now lets define a parameter to this Concurrent Program, and then we attach the value set that we defined to this parameter.

Validation

Value Set: XX_COST_CENTRES

Default Type:

☐ Required ☐ Enable Security

Description:

Default Value:

Range:

☑ Display

Display Size: 30

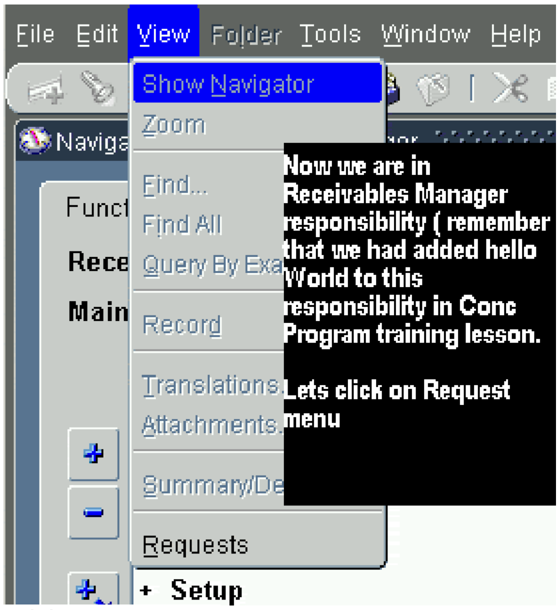
Concatenated Description Size: 25

Description Size: 50

Prompt: Cost Centres

Token:

Step 7.
Now to test this, lets go to receivables manager and click on Requests.

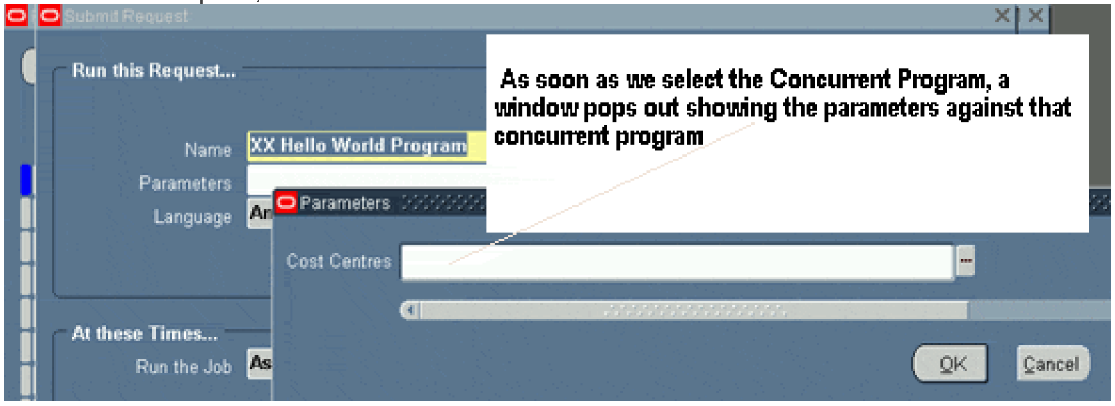


Now we are in Receivables Manager responsibility (remember that we had added hello World to this responsibility in Conc Program training lesson.

Lets click on Request menu

Click on Request,

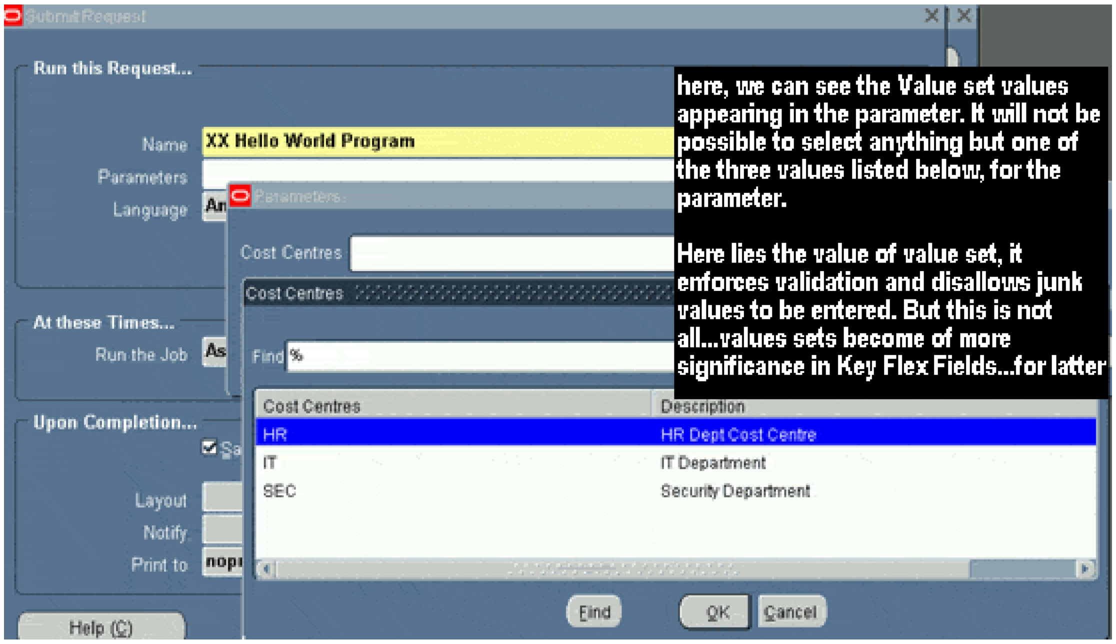
Step 8.
Submit New Request, and then click on OK.



Name: XX Hello World Program

As soon as we select the Concurrent Program, a window pops out showing the parameters against that concurrent program

Step 9
Now, we can see the values defined in the value set here.



XX Hello World Program

here, we can see the Value set values appearing in the parameter. It will not be possible to select anything but one of the three values listed below, for the parameter.

Here lies the value of value set, it enforces validation and disallows junk values to be entered. But this is not all...values sets become of more significance in Key Flex Fields...for latter

Find %

Cost Centres	Description
HR	HR Dept Cost Centre
IT	IT Department
SEC	Security Department

ihave uderstand why oracle consider **parameter as discriptive flexfield**

what i understand is because pop up window appear when u click on parameter or after selecting concurrent program

...
When you submit a concurrent program to run, Oracle lets you enter values into its Parameters.

This happens becasue Oracle pops-up a small window that contains all the parameters for the concurrent program being submitted.

That pop-up wnidow is nothing but a descriptive flexfield screen.

Not much emphasis is needed on this topic, as Conc program parameters being Descriptive Flexfields is something that gets taken care of internally by Oracle.

i am not uderstand oracle treat parameter as discriptive field , what it means
please advise

See this link, its an article contributed by Rakesh

http://getappstraining.blogspot.com/2006/12/value-sets-in-oracle-apps.html

If your filteration is based on profile option, then I suggest neatest way would be to create a database view which performs the filteration of data[based on profile option etc]. You can then use the database view in the SQL Query of table type value set.

: from where can I read in detail about all the types of value sets?? Oracle Documentation???. I have access to dev environment and know how to create dependent and independent value set but would like to try my hands on other types as well. Also, is it possible to restrict the values of the value set based on Organization (other than the Business Group). Suppose I have two Org under by BG - Org A, and Org B. Both the Orgs have their own Functions (e.g - IT & Technical in Org A) and (HR & Sales in Org B). While creating the position which uses Functions as one of the value sets, is it possible to restrict the value sets based on the Organization if I am using security profile based on Organizations? Thanks Mayank

The steps when using View for Table Type Value set are exactly the same as that for creating VS of table type on table.

Instead of table name , you will specify view name in table name field
see links
http://getappstraining.blogspot.com/2006/12/value-sets-in-oracle-apps.html

http://photos1.blogger.com/x/blogger/1078/3631/1600/874155/valueset_07.gif

Could you please give me an example of creating a table dependent value set using a databse view (or let me know where can I read about in detail). Thanks a lot for your help as always!

.
I would like to ask one question.
Is there any way by which we can pass multiple values for a parameter using value set.

I am afraid, by standard practice, you will only pass one value per valueset.

Yes, you can indeed concatenate strings with a delimiter to pass multiple values in one parameter....however there is no reason to take such approach. Please define one parameter per value being passed to concurrent program

i have some confussion...What is the procedure, if we want to create values in independent and dependent valuesets based on database table (for example po_headers_all (po_header_id) as INDEPENDENT value set and po_lines_all (po_line_id) as DEPENDENT valueset)

in order to achieve your requirement, in the table type value set of po_lines_all, include a where clause such as WHERE po_header_id = :$FLEX$.PO_HEADER_ID_VS_NAME

I have to create a value set having Maximum_SIZE =>6,Precision=>2 ,MIN_VALUE=>0.00 and MAX_VALUE=>100.00.

When i am submitting my request then in the Parameter window, if i enter 12.35 then program accept as it is but if i enter 12.356 then it shows an error message stating that U CAN ENTER UPTO 2 DECIMAL VALUES.
OR
After 2 decimal point, it will not allow us to enter any value.

Is it possible with value set?

Please refer to this link to find info about Special Value Set

http://getappstraining.blogspot.com/2006/12/value-sets-in-oracle-apps.html

I have defined independent valueset (number)with LOV as '1,5,10.15,20,25,30,35,40,45' but while running the report it is showing/sort by first digit/character not by its value as '1,10,15,20,25,30,35,40,45,5', I want to print as I mentioned above, could you please help me in this regard.

hii yaar ..
i hav this prob while creating a value set .. wwhich of table type .. here is the query try to helpme out ..

```
(SELECT DISTINCT transfer_batch,FROM_WAREHOUSE,TO_WAREHOUSE,SCHEDULED_RE
LEASE_DATE FROM ic_xfer_mst )

WHERE
TRUNC(SCHEDULED_RELEASE_DATE) BETWEEN
TO_DATE(:$FLEX$.FROM_DATE,'DD/MM/YYYY') AND
TO_DATE(:$FLEX$.TO_DATE,'DD/MM/YYYY')
AND FROM_WAREHOUSE=nvl(:$FLEX$.DV_WHSE,'%')
ORDER BY TRANSFER_BATCH
```

Custom Message not displaying when validated from Valueset Special
written by Ram Babu , October 08, 2007

Hi Anil,

I followed the steps:

1)From Message Form created message with language US, type ERROR, Message Text and submitted concurrent program Generate Messages.

2)Called this message through a valueset-type Special and event:validate with conditions.

3)Assigned this valueset to a concurrent program parameters to validate.

4) bounced the apache server.
Issue is the Message Name is not calling the actual message text from message form, instead displaying Message Name at runtime.

eg: fnd_message.set_name('ONT','TEST_MSG');
fnd_message.raise_error;

Showing message TEST_MSG at runtime instead of actual message text.
please suggest how to overcome from the issue.

thanks in advance,
R

How to gray out field?
Hi Anil,

I've three parameters TYPE,Project and Organization in a report.
Type is either Project or Organization.
If I select Project in Type field the Organization field grays out and Project field gets enabled. Similarly if Organization is selected then Project field grays out and Organization field gets enabled.
Please let me know how can a field be grayed out depending on the values of another field.

How do i disable a parameter2 on submit request, depending on a lov selected value of some other parameter1. Parameter2 is not a LOV. It is a user enterable field.

You will have to attach a value set to both those parameter. Without that, it isn't possible to have dependent segments in Descriptive Flexfields

Need Help In Diableing Or Enabling The Parameters
written by Hari Babu , November 28, 2007

Hi,
I have tow parameters. Depending on the first parameter value the scond parameter should be in disabled or enabled. How can we achieve this?

Help needed in selecting distinct values in a Value set
written by Sandip Jadhav , December 15, 2007

Hi Pradip,

This is regarding distinct values in LOV.

At the time of creating LOV, in table name write query like this e.g. (SELECT DISTINCT element_name,reporting_name FROM PAY_ELEMENT_TYPES_F_TL) A.

Then in Name and ID, select values like this - A.element_name, A.reporting_name
Write in where clause if u need.

can you please help me in creating table validated value sets.

-->in the edit information we have id,meaning and value in the colums frame which columns do we need to specify can you please illustrate with an exapmle.

can i change the value of the parameter on each run of concurrent program
written by sumit verma , February 12, 2008

i want to attech one value set to a parameter which takes two values say A and B on each run of the concurrent program. this value should change on the ever next run of the progarm. tat is if for now it is running for A the next it should run for B. how can i degine this value set. And values A and B are constant type

error while running the concurrent program
written by jyotsna , March 04, 2008

I followed the above steps mentioned in your tutorial.
I was able to run it without thevalue sets ok. Record got inserted in the table. But when I run this request

with value set, why do I get this error ?
I get this error :

```
XX_HELLOWORLD module: XX Hello World
------------------------------------------------------------------------

Current system time is 04-MAR-2008 14:10:24

------------------------------------------------------------------------

**Starts**04-MAR-2008 14:10:25
ORACLE error 6550 in FDPSTP

Cause: FDPSTP failed due to ORA-06550: line 1, column 7:
PLS-00306: wrong number or types of arguments in call to 'XX_REGISTER_USER_PRC'
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored.
```

dynamic report parameter
written by saurav , March 13, 2008

> I have 2 parameters.
>
> 1)1st parameter has independent value set and it will have only 2 possible values 'week' or 'month'. My 1st value set name is xx_period_type
>
> 2)2nd parameter has value set which should display the
> 'weekno-year'of last 5 years if parameter1 value passed is 'week'.
>
> 3)2nd parameter value set should display the 'month-year' of last 5 years if parameter1 value passed is 'month'.
>
> My code for 2nd list of value is :-
>
> (SELECT DECODE(:parameter1,'week',(TO_CHAR(SYSDATE -((LEVEL-1)*7) ,'ww-YYYY')),TO_CHAR(ADD_MONTHS(SYSDATE,-LEVEL 1),'Month-YYYY')) week_month FROM dual CONNECT BY LEVEL< DECODE(:parameter1,'week',52,12))
>
> This code is working fine from sqlplus on oracle client and even if I hardcode 'week'at the place of :parameter1 the value set is working fine in concurrent program,but
> when I used :$FLEX$.xx_period_type at the place of :parameter1
> it is giving error APP-FND-00005:INCORRECT ARGUMENTS WERE PASSED TO USER EXIT #MESSAGE_TOKEN in value set.

- How to gray out field?

I've three parameters TYPE,Project and Organization in a report.
Type is either Project or Organization.
If I select Project in Type field the Organization field grays out and Project field gets enabled. Similarly if Organization is selected then Project field grays out and Organization field gets enabled.
Please let me know how can a field be grayed out depending on the values of another field.

Have a look at http://oracle.anilpassi.com/sm...ields.html

http://oracle.anilpassi.com/hr...br100.html
http://getappstraining.blogspo...-apps.html

In Oracle Apps Some tables ends with TL,VL,what is the significance of the TL and VL
http://getappstraining.blogspo...-apps.html

http://oracle.anilpassi.com

HOW WILL U REGISTER THE REPORT. APPS
What is incompatibility?
What is compulsory parameter when you create report?
what was the structure of the accounting flexfield in previous project
What happens when u complete an invoice
What Credit memo / Debit Memo
What is set of books
which is the parameter set to validate keyflexfield value
What are the factors for application developer responsibility.
What are the Default Types for Parameters. What is the use of each one of it.
What are the differences between executable and concurrent program
Is it mandatory to use API's (like Fnd INIT, FND Flex, FND Exit) in forms and reports when working in Apps environment?

What other responsibilities Application Developer can contain?

How the rows retrive from error tables, bad files?
What the required columns of po_headers,po_lines and also tell me the derived columns.
What is meaning of action columns in po_headers_interafce table and what r the types of actions.

1.what kind of document you got as for your assignment? what are the specification & design of that document?
2.what document you submit after completion of your task & specifications and design?
3.what ar the processes & phases you gone throuth during development?
4.which document you submit in testing and debugging phase?
5.how you go about your work when it is given to you?
6.what functional document you have given for analysis of requirements?

.How do u test the reports in oracle application?
Where are APIs located on my system?
Where can I find information on an API, including what it does and its parameters?

Do APIs validate Flexfield data?

5. How can I use the CREATE_USER_ACCOUNT_API?
The delivered processes are:
User Account Employee Extract Process
Data Pump Engine
Data Pump Batch Exceptions Report
For complete details, please see the Batch Creation of User Accounts White Paper on Metalink, at Top Tech Docs > ERP > HRMS > Human

6. Oracle Web ADI includes the following features:

- What is date tracking? How does Oracle keep it?
- What is Fast-Formula?
- Who is giving you specification?

GE Industries

- Is it mandatory to use API's (like Fnd INIT, FND Flex, FND Exit) in forms and reports when working in Apps environment?

- What other responsibilities Application Developer can contain?

12. What is a lookup in oracle apps

.31. How to debug programs oracle applications
33. How to make menu for specific concurrent programs.

What are the server side and client side concepts in PL/SQL APIs
In KFF only independent Value Sets can be used for different segments and all segments of type Varchar2.

- What is Rollup Groups in ACCOUNTING KFF.
- .How to open new periods ? and how to create a new period.

What is a must parameter in report
How to attach flex fields in report & forms?
6) Parameter passing in reports

GESOFT QUESTIONS.

who use different set of books.
5) What is a calendar ? Types ?
Calendar is accuonting period defined as per organizational needs.
Types: Accounting and Workday calendar.
1) Which project have u done regarding implementation of apps ?
2) What is the location of that organization ?
3) How many operating units and legal entities ?
4) What is essential set ups for a fresh implementation of apps ?
5) What needs to be done for upgradation of 11.5.7 to 11.5.9 ?
6) What is lot and serial control ? Where is it defined ?
9) What is difference between india localization set up and normal set up ?
Taxations - Outside India nornally all taxes are in one form - VAT.
IN India various taxes and surcharges come into picture.
po interface steps,tables and columns .
.31>Lets suppose if i want to see the data of the table. And i write a select statemant
but it didnt diaplsy the data.so what i should do to see the data in sql*plus environment.
37>Tell me the steps of po interface.
38>What the required columns of po_headers,po_lines and also tell me the derived columns.
.40>What is meaning of action columns in po_headers_interafce table and what r the types of actions.
48>Types of purchase documents.
49>can i have row level and statement trigger at same time.
50>tell me the code what u will write in format trigger.
Q. What r the validation and name the columns for u have done the validation, which interface u, has develop?
Q. What is Entered_dr and entered_cr?
Q. How will get total number of records in database in single select statement?
Q. How u will debug Ur reports?
nterview Questions Of Tata Honey Well, Pune Dated: 03/06/04
Q 5:-->Select 1 from emp? what will be the O/P?
Q 6-->Different b/w count(1) and Count(*)?
Q 14:-->command for compiling custom.pll? where you willl compile it?

Q 15:-->Which outbound interface u have done?
different b/w API's for US and UK?
Q 27:-->difference between stored procedure or stored functions?
.
Q 29:-->can we use procedure name in select stmt and can we use function name in select stmt?
Q 30: **If i have a function and using it in select stmt this function is inserting value in table so,is it possible to use it in select stmt?**
Q 31:-->thru API u have to eliminate Duplicate
Q 32:-->thru a procedure, how u will identify Duplicate? (ans hint: using PL/SQL)
Q 33:-->where is the data file located?
Q 34:-->what type of error you encounter in report?
Q 51:-->can we delete Concurrent program?
16 which tables contain org id
4. validations on gl interface
9. requirement of application short name in reports
12. Is it mandatory for parameter to have a valueset
15. how to avoid number of rows while uploading data from flat file to staging table
21. how to check the total number of records in plsql table
How do you handle exceptions in reports?
If you have a procedure in a remote database which has an error, in which database would you write the exception to handle the error?
What is the difference between Lexical parameters and User parameters in reports?
While using a cursor, will 'too_many_rows' exception occur or not?

Can you use an insert statement in a report?
resume_based:
how do you migrate the reports,
how do you enable the DFF in AP

Multiorg concept
Levels,
Set of books :Components
Diff between org_id,Organization_id?
10.suppose u have two fields in your report parameter form(apps,concurrent program(execute method:oracle report)..
deptno: ------
empno : --------
if i give deptno 10 then all the empno belongs to that deptno has to be displayed.
if i give deptno 20 then all the empno belongs to that deptno has to be displayed.

tell me about this(how can u implement this)?
11. Another solution for the above problem?
12.is any table in po(purchase order) belogs to ap(accounts payable)?
13.

```
declare
function fun_name is
b varchar2(10);
begin
select ename  into b form emp where empno=121;
update emp set  name='spider'  where empno=1213;

exception
when no_data_found then
 dbms_output.put_line('spider exception');
when others then
 dbms_output.put_line('spider exception others');
```

```
end fun_name;
begin

end;
```

what is major error in this program?(don`t say variable not declared...) ?

Report Customization
1) Report Name:- Distribution Set Listing (AP)
Customization :- Display the Distribution list as 'like' the given distribution set name
Short Name :- APXGDGDL
Parameter :- Active/ Inactive Set, Effective Date
Table:-
AP_distribution_sets,
Ap_distribution_set_lines
Gl_code_combination

Solution:-
Add one parameter (Distribution set name) in the parameter list. See the parameter listing which is required for the report with the help of report name and short name. The fpt the report from apps/--appl/ap/11.5.0/reports/US to your directory.

Create one parameter in the user parameter list (ie MYDIST_SET_NAME)
Go the report builder navigator window click on the icon data model
Click on the query
Change the query as per your bind variable (ie. Dist_name like 'M11_Dist')
Save the report – ftp the rdf file.
Create concurrent program with the required parameter as per the original report also add your new parameter to that concurrent program attach that program to the oracle payable or your responsibility group.

This customize distribution set listing report shows the distribution set listing like the parameter entered by the user.

---

4) Report Name :- Invoice Register (AP)

Customization:-
- **Display the invoice register as per given supplier name.**
- **Use the LOV for selection of supplier**

Parameter:--
Cancelled Invoices only, Unapproved Invoices only.
Table :-
Ap_Invoices
Po_Vendors
Ap_Invoice_distribution
Ap_Batches
Ap_lookup_codes
Ap_tax_codes

Solution:-

See the parameter listing which is required for the report with the help of report name and short name.

The ftp the report from apps/..appl/ap/11.5.0/reports/US to your directory

Create one parameter in the user parameter list (ie., MY SUPPLIER)
Go the report builder navigator window click on the icon data model click on the query.

Change the query as per your bind variable (ie dist name like 'M11_PAY')
Save the report – ftp the rdf file.

Create a table type value set for supplier name from the table po_vendors_all or you can select predefined value set which contain the supplier name.

Create concurrent program with the required parameter as per original report also add your new parameter to that concurrent program add the value set for supplier name parameter.
Attach the program to the oracle payable or your responsibility group.

This customize Invoice register shows the parameter list with supplier name LOV

---

5) Report Name :- Final Payment Register (AP)
Customization :- Add one field status in the Header section
Short name:- APXPBFPR
Parameter :- Payment , Trace Switch
Table :-
Ap_Checks
Po_vendor_sites
Ap_bank_account
Ap_invoice_selection_criteria
Ap_lookup_codes
Fnd_Teritories

Solution:-
See the parameter listing which is required for the report with the help of report name and short name.
The ftp the report from apps/..appl/ap/11.5.0/reports/US to your directory
Go to the data model and click on the query.
In the select statement of ap_invoice_selection_criteria add status with alias
Go the report builder navigator window click on the icon Layout editor
Go the header section
Separate all the frames of header section, place the boilerplate text for the status, Add one field and give the source to the field as status
Error:- Frequency Below group.
Check the frame source or check the frame one in the another frame save the report – ftp the rdf file.
Create concurrent program with required parameter as per original report attach that program to the oracle payable or your responsibility group.

This Customize Final Payment Register Report shows the header section with the addition field STATUS

---

6) Report name: - Internal Requisition Status Report
Customization: - Add total for quantity ordered.
Short name:-POXRQSIN
Parameter:-
Table:-

Po_requisition_lines
Po_requisition_headers
Mtl_system_items
Hr_employees

Solution:-
See the parameter listing which is required for the report with the help of report name and short name.
Then ftp the report name from apps/..appl/inv/11.5.0/reports/US to your directory.
Create one summary column for the quantity order for sum
Go the report builder navigator window click on the icon layout add the total field.
Give the source of the summary column to that field.
Save the report – ftp the rdf file.
Create concurrent program with required parameter as per the original report
Attach that program to the oracle inventory or your responsibility group.
This customize internal requisition status report shows total for quantity order.

---

7) Report name: - AP-Invoice Audit Listing (APXINLST)

This report gives the details of all the booked invoices. Please add the following columns. Accounted for
Approval status
GL Date

Steps:--

- Download the original report through ftp to the local folder from /apps/..appl/ap/11.5.0/reports/US . The name of the report is APXINLST.rdf

- Save the original report as APXINLST_ORG.rdf

- Open APXINLST.rdf in report builder.

- Modify the query as follows .

```
SELECT
          i.invoice_num    c_invoice_number,
          i.invoice_date    c_invoice_date,
          i.invoice_currency_code c_currency_code,
          i.invoice_amount c_invoice_amount,
          lc.displayed_field  c_invoice_type,
          v.vendor_name c_vendor_name,
          v.segment1  c_vendor_number,
          i.gl_date c_gl_date,
          i.approval_status   c_approval_status,
         i.accts_pay_code_combination_id  c_code_Combination,
        &p_flexdata   c_flex_data,
        Gcc.chart_of_accounts_id  c_num
From
   Ap_invoices i,
   Po_vendors v,
   Ap_lookup_codes lc,
   Gl_code_combinations gcc
Where
   i.invoice_type_lookup_code=nvl( :p_invoice_type,i.invoice_type_lookup_code)
and i.vendor_id=v.vendor_id
and lc.lookup_code(+)= i.invoice_type_lookup_code
and lc.lookup_type(+)= 'INVOICE TYPE'
and i.invoice_amount >=:p_min_inv_amount
and i.invoice_date  >=:p_date_since
and gcc.code_combination_id=i.accts_pay_code_Combination_id
&c_order_by
```

Add a Formula column in Q_inv data block.

- C_DESC_ALL with datatype char(400) and write following PL/SQL

```
SRW.REFERENCE(:C_NUM);
SRW.REFERENCE(:C_FLEXDATA);
SRW.USER_EXIT('FND FLEXIDVAL
                    CODE="GL#'
                    NUM=":C_NUM"
                    APPL_SHORT_NAME="SQLGL"
                    DATA=":C_FLEXDATA"
                    DESCRIPTION=":C_DESC_ALL"
                    DISPLAY="ALL");
RETURN(:C_DESC_ALL);
```

- Add a field in layout editor and assign c_desc_all to its source property
- Add a field in layout editor and assign c_gl_date to its source property
- Add a field in layout editor and assign c_approval_status to its source property
- Save the report
- Transfer this report to /apps/..appl/ap/11.5.0/reports/US
- Run the report from oracle payables responsibility.

AP-Invoice History Report (APXINHIS.rdf)

This report gives all the details suppliers wise:
Pease add the following columns
Name of the bank
Bank account number
Payment Method—(EDI, Check etc)

<u>Steps:</u>

Download the original report through ftp to the local folder from /apps/..appl..
Save the original report as APXINHIS_ORG.rdf
Open APXINHIS.rdf in report builder.
Modify the query as follows

```
Select
        Po.vendor_name c_vendor,
        Pvs.vendor_site_code c_vendor_site,
        i.invoice_num  c_invoice_number,
        i.invoice_date c_transaction_date,
        i.invoice_lookup_code c_transaction_type,
        alc.dispalyed_field        c_transaction_type_field,
        i.payment_currency_code c_curr,
        decode(i.doc_sequence_value,NULL,i.voucher_num,
                i.doc_sequence_value) c_doc_sequence_number,
        f2.name  c_doc_sequence_name,

        decode(i.invoice_type_lookup_code,'PREPAYMENT',
        nvl(ap_utilities_pkg.ap_round_currency(i.original_prepayment_amount*
                    i.payment_cross_rate, i.payment_currency_code),0),
             nvl(nvl(i.pay_curr_invoice_amount,i.invoice_amount),0)) c_total_invoice_amt,
             ck.doc_sequence_value  c_doc_sequence_number,
            ck.check_number        c_transaction_number,
            f.name                        c_doc_sequence_name,
           aip.accounting_date c_transaction_date
          apd.line_type_lookup_code c_transaction_type,
          ck.currency_code  c_curr,
         -1*apd.amount  c_transaction_amt,
          Abb.bank_name c_bak_name,
        Aip.bank_account_num c_bank_account_num,
        i.payment_method_lookup_code c_payment_method

From
          Po_vendors pv,
          Po_vendor_sites_all pvs,
          Ap_invoices i,
          Ap_checks ck,
          Fnd_document_sequences f,
          Fnd_document_sequences f2,
          Ap_invoice_payments aip,
          Ap_payment_distributions apd,
          Ap_lookup_codes alc,
          Ap_bank_branches abb
Where
          Pv.vendor_id=pvs.vendor_id
  And  i.vendor_id=pv.vendor_id
 And  i.vendor_site_id=pvs.vendor_site_id
 And i.invoice_id=aip.invoice_id
 And i.invoice_type_lookup_code=alc.lookup_code
 And alc.lookup_type='INVOICE TYPE'
And apd.invoice_payment_id(+)=aip.invoice_payment_id
And ck.check_id(+)=aip.check_id
And f.doc_sequence_id(+)=ck.doc_sequence_id
And f2.doc_sequence(+)=i.doc_sequence_id
And decode(i.invoice_type_lookup_code,'PREPAYMENT',
          NVL(AIP.invoice_payment_type,'X'),1)
<>(i.invoice_type_lookup_code,'PREPAYMENT','PREPAY',2)
And aip.bank_num=abb.bank_num
&lp_vendor_id
&lp_vendor_site
&lp_invoices
&lp_invoices_number_from
&lp_invoices_number_to
&lp_doc_sequence_name
&lp_doc_sequence_number_from
&lp_doc_sequence_number_to
&lp_invoice_date_from
&lp_invoice_date_to
Order by
Pv.vendor_name,
Pvs.vendor_site_code,
i.invoice_num,
i.invoice_date,
i.invoice_type_lookup_code,
aip.accounting_date,
```

apd.line_type_lookup_code,

Modify the layout editor to add the new columns in it.

Save the report

Transfer this report to /apps/..appl/11.5.0/reports/US

Run the report from oracle payables responsibility.

Report name :- AP- Payables Trial Balance

This is the report for trail balances for suppliers. Give the details of the account that is shown in this. For e.g 01-000.2210-0000-0000 – Details of all the segements

Steps:-

- u Down load the original report through ftp to your local folder (APXTRBAL.rdf)
- u Save the original report as APXTRBAL_org.rdf
- u Open APXTRBAL.rdf in report builder.
- u Add a user parameters :- p_flexdata1 with data type char(400) and initial value GCC.Segment1||'-'||gcc.segemnt2
- u Change the query in Q_detail data block as:-

```
      Select & c_select_le legal_entity,
            &c_select_ou operating_unit,
            Pv.vendor_name supplier_name,
            Ai.invoice_num invoice_number,
            Ai.invoice_date invoice_date,
            Ai.invoice_currency_code invoice_curr,
            Decode(ai.invoice_currency_code,asp.base_currency_code,
            ai.invoice_amount,
            (ap_utilities_pkg.ap_round_currency((ai.invoice_amount * nvl(ai.exchage_rate,1)),
    asp.base_currency_code))) invoice_amount,
     Atb.remaining_amount remaining_amount,
    Atb.code_combination_id code_combination_id
    Atb.vendor_id third_party_id,
    Fnd_flex_ext.get_segs('SQLGL','GL#', :p_chart_of_accounts_id,atb.code_combination_id)
concat_segments,
    Ai.description invoice_description,
    Atb.invoice_id source_invoice_id,
    Atb.org_id org_id,
    Atb.set_of_books_id,
    gcc.chart_of_accounts_id c_num1,
    &p_flexdata1 c_flexdata1

    From
            &c_org_from_tables,
            Ap_trial_bal atb,
            Po_vendors pv,
            Ap_system_parameters_all asp,
            Ap_invoices_all ai,
            Gl_code_combinations gcc,
    Where
            Nvl(atb.org_id,-99)=nvl(ai.org_id,-99)
            And nvl(atb.org_id,-99)=nvl(asp.org_id,-99)
            And atb.vendor_id=pv.vendor_id
            And atb.invoice_id=ai.invoice_id
    And gcc.code_combination_id=atb.code_combination_id
    And atb.set_of_books_id=:p_set_of_books_id
    &p_supp_acc_where
    &c_multi_org_where
    &p_org_where
```

- Add the following SRW function in Before Report Trigger

```
            SRW.REFERENCE(:P_STRUCT_NUM);
            SRW.USER_EXIT(‘FND FLEXSQL
                                    CODE=”GL#”
                                    NUM=”:P_STRUCT_NUM”
                                    APPL_SHORT_NAME=”SQLGL”
                                    OUTPUT=”:P_FLEX_DATA1”
                                    TABLEALIAS=”GCC”
                                    MODE=”SELECT”
                                    DISPLAY=”ALL”
```

- Add a formula column in Q_Detail data block’s G_concat_segment group

```
            C_DISC_ALL1 with datatype char(400) and write following PL/SQL
            SRW.REFERENCE(:C_NUM1);
            SRW.REFERENCE(:C_FLEXDATA1);
            SRW.USER_EXIT(‘FND FLEXIDVAL
                                    CODE=”GL#”
                                    NUM=”:C_NUM1”
                                    APPL_SHORT_NAME=”SQLGL”
                                    DATA=”:C_FLEXDATA1”
                                    DESCRIPTION=”:C_DISC_ALL1”
                                    DISPLAY=”FA_COST_CTR”’);
            Return(:C_DISC_ALL1);
```

- Add a field in layout editor and assign C_DISC_ALL1 to its source property. This field should be in the same frame of contact_segemnt field.
- Save the report
- Transfer this report to /apps/..appl/…
- Run the report from oracle payables responsibility.

Parameters	prompt	Token
Set of Books id	Accounting Currency Value Set AP_SRS_ACCTG_CURRENCY	p_set_of_books_id
Accounting Date	As of Date Value Set FND_STANDARD_DATE	p_accounting_date
Supplier Name	Supplier Name Value Set AP_SRS_VENDOR_NAME	p_vendor_id
Accounting Flex Field	Liability Account Value set AP_SRS_FLEXFIELD.	p_account_id
Summarize Report	Summarize Report Value Set AP_SRS_YES_No_MAND	p_summarize
From date	Exclude Invoices	p_from_date

Prior to
Value set
FND_STANDARD_DATE

Customization :-
I make the ageing of the remaining amount column, I categories the amount in three different category from 0-30 days, 31-60 days, 61-above days pending days amount , I make a formula column in the group. From this i get the how many days the amount is pending.

Report name :- AP- Distribution Set Listing (APXGDGDL)

Give the details of account description with all the segments
Steps:-

- Download the original report through ftp to the local folder from /apps/../reports/US. The name of the report is APXGDGDL.rdf
- Save the original report as APXGDGDL_org.rdf
- Open APXGDGDL.rdf in report builder.
- Add two formula columns in Q_Dist_Set data block's G_Dist_Set_Dtl group.
    - c_desc_all with datatype char(400) and write the following Pl/sql

```
                SRW.REFERENCE(:c_chart_of_account_id);
                SRW.REFERENCE(:c_flexdata);
                SRW.USER_EXIT('FND FLEXIDVAL CODE="GL#'
                                    DATA=":C_FLEXDATA"
                                   APPL_SHORT_NAME="SQLGL"
                                   DESC RIPTION=":C_DESC_ALL"
                                  DISPLAY="ALL"
                  NUM=":C_CHART_OF_ACCOUNTS_ID");
            RETURN(:C_DESC_ALL);
```

- Add a field in layout editor and assign C_DESC_ALL to its source Property
- Save the Report
- Transfer this report to /apps/../reports/US
- Run the report from Oracle Payables responsibility.

Report name :- Purchase Order detail report (POXPOSTD) version 11.5.3

Customizations Done :-

u Changed from landscape format to portrait format.
u Added company's logo to the TOP left corner of PO report
u Whole layout to be adjusted to be fit in preformatted blank stationary.
u Running totals column line by line.
u Addition of some place holders columns to the report to show the deficiency columns for such report as per needs of the organizations.

Problems faced during this:
A layout problem with the printer was the major problem.
A layout problem with the low frequency error as the user parameter being passed was not matching.

Tables used:-

u po_lines_locations
u mtl_system_items
u mtl_Categories

- po_line_types
- hr_locations
- gl_set_of_books
- financials_system_parameters
- mtl_default_sets_view
- fnd_lookups
- fnd_currenices
- Po_headers
- Po_vendors
- Po_vendor_sites
- Per_people_f
- Po_lookup_codes
- Po_document_types

Po_line_locations:-

Po_line_locations contains information about purchase order shipment schedules and blanket agreement price breaks. You need one row for each schedule or price break you attach to a document line. There are seven types of documents that use schedules

- RFQ s
- Quotations
- Standard purchase orders
- Planned purchase orders
- Planned purchase order releases
- Blanket purchase orders
- Blanket purchase order releases

Report name :- Print Purchase order Report (POXPRPOP.rdf)

Customizations Done:-

- Added company's logo to the top left corner of PO(Bitmap.gif)
- Calculation pf page wise totals carried forward to next page
- Display of delivering to persons under the note to supplier
- Distribution lines are added to the quantity ordered(example if the total quantity ordered is 30 and it is distributed into 10 and 20 –split between requestors
- Additional of some placeholder columns in the column to show the deficiency columns for such reports as per needs of the organizations.

Problems faced:-

Taking an output file & then giving a print command was not yielding the right output. Print command was yielding the right print when submitted through print check box on purchase order approval form.

Tables Used

gl_set_of_books
financials_system_parameters
fnd_look_ups
po_headers_print
po_releases
po_lines_print
mtl_system_items
fnd_attached_docs_from_vl
po_line_locations_print
mtl_system_items_tl
po_distributions_print
fnd_attached_docs_from_vl

fnd_attached_docs_from_vl
financials_system_parameters
dual
po_action_history
po_distribution_print
fnd_currencies
fnd_atatched_docs_from_vl

Relations used in this report

Po_headers_print.release_id= po_release.release_id(+)
Po_headers.po_type!='RELEASE'
Gl_set_of_books.set_of_books_id=financial_system_parameters.set_of_books_id

Account Analysis GL Report (180 chars)

Report Name:	Account Analysis Report (180 chars)
Executable Name:	GLRJED
Executable File Name:	GLRJED
Application Name:	Oracle General Ledger
Responsibility:	General Ledger Vision Operation

Functionality of report:

This report prints the journal entry lines and beginning and ending balances of the accounts you request. For each journal entry line, the report prints the source, category, batch name, journal entry name, account, description, entry/line/source item reference information, and the debit or credit amount.

Parameters: 1. Type (Entry item or Line item or Source item)
2. Currency
3. Balance type
4. Starting/Ending period
5. Flex field From/to
6. Order by

Table Used:

u GL_PERIOD_STATUSES
u GL_CODE_COMBINATIONS
u GL_BALANCES

Customization Task:

Displaying description of flexfields in headings for min. and max. accounts.

Steps:

u Creating one user parameter P_FLEX_DATA_R with some initial value.

Creating one formula column and its Pl/Sql editor used FLEXIDVAL user exit and here used a parameter "Description" here specify name of that bind parameter which specifying the code combination value.