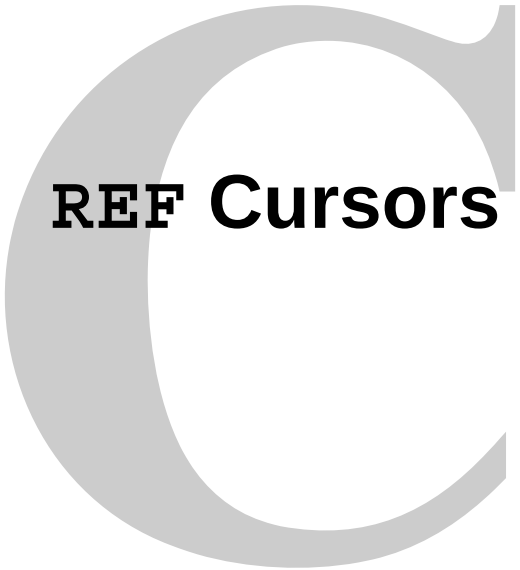

REF Cursors

Cursor Variables

- Cursor variables are like C or Pascal pointers, which hold the memory location (address) of an item instead of the item itself.
- In PL/SQL, a pointer is declared as `REF X`, where `REF` is short for `REFERENCE` and `X` stands for a class of objects.
- A cursor variable has the data type `REF CURSOR`.
- A cursor is static, but a cursor variable is dynamic.
- Cursor variables give you more flexibility.

Why Use Cursor Variables?

- You can use cursor variables to pass query result sets between PL/SQL stored subprograms and various clients.
- PL/SQL can share a pointer to the query work area in which the result set is stored.
- You can pass the value of a cursor variable freely from one scope to another.
- You can reduce network traffic by having a PL/SQL block open (or close) several host cursor variables in a single round trip.

Defining REF CURSOR Types

- Define a REF CURSOR type:

```
Define a REF CURSOR type
TYPE ref_type_name IS REF CURSOR [RETURN
return_type];
```

- Declare a cursor variable of that type:

```
ref_cv ref_type_name;
```

- Example:

```
DECLARE
TYPE DeptCurTyp IS REF CURSOR RETURN
departments%ROWTYPE;
dept_cv DeptCurTyp;
```

Using the OPEN-FOR, FETCH, and CLOSE Statements

- The OPEN-FOR statement associates a cursor variable with a multirow query, executes the query, identifies the result set, and positions the cursor to point to the first row of the result set.
- The FETCH statement returns a row from the result set of a multirow query, assigns the values of select-list items to corresponding variables or fields in the INTO clause, increments the count kept by %ROWCOUNT, and advances the cursor to the next row.
- The CLOSE statement disables a cursor variable.

An Example of Fetching

```
DECLARE
   TYPE EmpCurTyp IS REF CURSOR;
   emp_cv   EmpCurTyp;
   emp_rec  employees%ROWTYPE;
   sql_stmt VARCHAR2(200);
   my_job   VARCHAR2(10) := 'ST_CLERK';
BEGIN
   sql_stmt := 'SELECT * FROM employees
                WHERE job_id = :j';
   OPEN emp_cv FOR sql_stmt USING my_job;
   LOOP
      FETCH emp_cv INTO emp_rec;
      EXIT WHEN emp_cv%NOTFOUND;
      -- process record
   END LOOP;
   CLOSE emp_cv;
END;
/
```