

*A Guide to Language  
Fundamentals*

**2nd Edition**  
**Covers Oracle9i**



# Oracle PL/SQL Language

*Pocket Reference*



**O'REILLY®**

*Steven Feuerstein,  
Bill Pribyl & Chip Dawes*

SECOND EDITION

---

# Oracle PL/SQL Language

## *Pocket Reference*

*Steven Feuerstein, Bill Pribyl, and  
Chip Dawes*

**O'REILLY®**

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

## Bulk Binds

You can use collections to improve the performance of SQL operations executed iteratively by using *bulk binds*. Bulk binds reduce the number of context switches between the PL/SQL engine and the database engine. Two PL/SQL language constructs implement bulk binds: FORALL and BULK COLLECT INTO.

The syntax for the FORALL statement is:

```
FORALL bulk_index IN lower_bound..upper_bound
    [SAVE EXCEPTIONS]
    sql_statement;
```

*bulk\_index* can be used only in the *sql\_statement* and only as a collection index (subscript). When PL/SQL processes this statement, the whole collection, instead of each individual collection element, is sent to the database server for processing. To delete all the accounts in the collection *inactives* from the table *ledger*, do this:

```
FORALL i IN inactives.FIRST..inactives.LAST
    DELETE FROM ledger WHERE acct_no = inactives(i);
```

The default is for Oracle to stop after the first exception encountered. Use the keywords SAVE EXCEPTIONS to tell Oracle that processing should continue after encountering exceptions. The cursor attribute %BULK\_EXCEPTIONS stores a collection of records containing the errors. These records have two fields, EXCEPTION\_INDEX and EXCEPTION\_CODE, which contain the FOR ALL iteration during which the exception was raised, as well as the SQLCODE for the exception. If no exceptions are raised, the SQL%BULK\_EXCEPTION.COUNT method returns 0. For example:

```
DECLARE
    TYPE NameList IS TABLE OF VARCHAR2(32);
    name_tab NameList := NameList('Pribyl'
        , 'Dawes', 'Feuerstein', 'Gennick'
        , 'Pribyl', 'Beresniewicz', 'Dawes', 'Dye');
    error_count NUMBER;
```

```

bulk_errors EXCEPTION;
PRAGMA exception_init(bulk_errors, -24381);
BEGIN
  FORALL indx IN name_tab.FIRST..name_tab.LAST SAVE
  EXCEPTIONS
    INSERT INTO authors (name) VALUES (name_tab(indx));
    -- authors has pk index on name
  EXCEPTION
    WHEN others THEN
      error_count := SQL%BULK_EXCEPTIONS.COUNT;
      DBMS_OUTPUT.PUT_LINE('Number of errors is ' ||
        error_count);
      FOR indx IN 1..error_count LOOP
        DBMS_OUTPUT.PUT_LINE('Error ' || indx || '
          occurred during '||'iteration ' ||
            SQL%BULK_EXCEPTIONS(indx).ERROR_INDEX);
        DBMS_OUTPUT.PUT_LINE('Error is ' ||
          SQLERRM(-SQL%BULK_EXCEPTIONS(indx).ERROR_CODE));
      END LOOP;
END;
/

```

```

Number of errors is 2
Error 1 occurred during iteration 5
Error is ORA-00001: unique constraint (.) violated
Error 2 occurred during iteration 7
Error is ORA-00001: unique constraint (.) violated

```

The syntax for the BULK COLLECT INTO clause is:

```
BULK COLLECT INTO collection_name_list;
```

where *collection\_name\_list* is a comma-delimited list of collections, one for each column in the SELECT. Collections of records cannot be a target of a BULK COLLECT INTO clause. However, Oracle does support retrieving a set of typed objects and “bulk collecting” them into a collection of objects.

The BULK COLLECT INTO clause can be used in SELECT INTO, FETCH INTO, or RETURNING INTO statements. For example:

```

DECLARE
  TYPE vendor_name_tab IS TABLE OF
    vendors.name%TYPE;

```

```

TYPE vendor_term_tab IS TABLE OF
    vendors.terms%TYPE;
v_names vendor_name_tab;
v_terms vendor_term_tab;
BEGIN
    SELECT name, terms
        BULK COLLECT INTO v_names, v_terms
        FROM vendors
        WHERE terms < 30;
    ...
END;
```

The next function deletes products in an input list of categories, and the SQL RETURNING clause returns a list of deleted products:

```

FUNCTION cascade_category_delete (categorylist clist_t)
RETURN prodlist_t
IS
    prodlist prodlist_t;
BEGIN
    FORALL aprod IN categorylist.FIRST..categorylist.LAST
        DELETE FROM product WHERE product_id IN
            categorylist(aprod)
            RETURNING product_id BULK COLLECT INTO prodlist;
    RETURN prodlist;
END;
```

You can use the SQL%BULK\_ROWCOUNT cursor attribute for bulk bind operations. It is like an associative array containing the number of rows affected by the executions of the bulk bound statements. The *n*th element of SQL%BULK\_ROWCOUNT contains the number of rows affected by the *n*th execution of the SQL statement. For example:

```

FORALL i IN inactives.FIRST..inactives.LAST
    DELETE FROM ledger WHERE acct_no = inactives(i);
FOR counter IN inactives.FIRST..inactives.LAST
LOOP
    IF SQL%BULK_ROWCOUNT(counter) = 0
    THEN
        DBMS_OUTPUT.PUT_LINE('No rows deleted for ' ||
            counter);
    END IF;
```

```
END LOOP;
```

You cannot pass `SQL%BULK_ROWCOUNT` as a parameter to another program, or use an aggregate assignment to another collection. `%ROWCOUNT` contains a summation of all `%BULK_ROWCOUNT` elements. `%FOUND` and `%NOTFOUND` reflect only the last execution of the SQL statement.