14

Oracle Supplied Packages

Objectives

After completing this lesson, you should be able to do the following:

- Write dynamic SQL statements using DBMS_SQL and EXECUTE IMMEDIATE
- Describe the use and application of some Oracle server-supplied packages:
  - DBMS_DDL
  - DBMS_JOB
  - DBMS_OUTPUT
  - UTL_FILE
  - UTL_HTTP and UTL_TCP

Using Supplied Packages

Oracle-supplied packages:

- Are provided with the Oracle server
- Extend the functionality of the database
- Enable access to certain SQL features normally restricted for PL/SQL

Using Native Dynamic SQL

Dynamic SQL:

- Is a SQL statement that contains variables that can change during runtime
- Is a SQL statement with placeholders and is stored as a character string
- Enables general-purpose code to be written
- Enables data-definition, data-control, or session-control statements to be written and executed from PL/SQL
- Is written using either DBMS_SQL or native dynamic SQL

ORACLE

Execution Flow

SQL statements go through various stages:

- Parse
- Bind
- Execute
- Fetch

Note: Some stages may be skipped.

Using the DBMS_SQL Package

The DBMS_SQL package is used to write dynamic SQL in stored procedures and to parse DDL statements. Some of the procedures and functions of the package include:

- OPEN_CURSOR
- PARSE
- BIND_VARIABLE
- EXECUTE
- FETCH_ROWS
- CLOSE_CURSOR

Using DBMS_SQL

```
CREATE OR REPLACE PROCEDURE delete_all_rows
   (p_tab_name IN VARCHAR2, p_rows_del OUT NUMBER)
 IS
   cursor_name   INTEGER;
 BEGIN
   cursor_name := DBMS_SQL.OPEN_CURSOR;
   DBMS_SQL.PARSE(cursor_name, 'DELETE FROM '||p_tab_name,
                  DBMS_SQL.NATIVE );
   p_rows_del := DBMS_SQL.EXECUTE (cursor_name);
   DBMS_SQL.CLOSE_CURSOR(cursor_name);
END;
/
```

Use dynamic SQL to delete rows

```
VARIABLE deleted NUMBER
EXECUTE  delete_all_rows('employees', :deleted)
PRINT deleted
```

```
PL/SQL procedure successfully completed.

                    DELETED
                                         109
```

Using DBMS_SQL

```
CREATE OR REPLACE PROCEDURE delete_rows
  (p_tab_name IN VARCHAR2, p_rows_del OUT NUMBER) IS
     cursor_name INTEGER;
BEGIN
     cursor_name := DBMS_SQL.OPEN_CURSOR;
     DBMS_SQL.PARSE (cursor_name, 'DELETE FROM '||p_tab_name ||
                        ‘ WHERE employee_id >= 207 ‘, DBMS_SQL.NATIVE );
     p_rows_del := DBMS_SQL.EXECUTE (cursor_name);
     DBMS_SQL.CLOSE_CURSOR(cursor_name);
END;
/
```

Use dynamic SQL to delete rows

```
VARIABLE deleted NUMBER
EXECUTE delete_rows('employees', :deleted)
PRINT deleted
```

Using the EXECUTE IMMEDIATE Statement

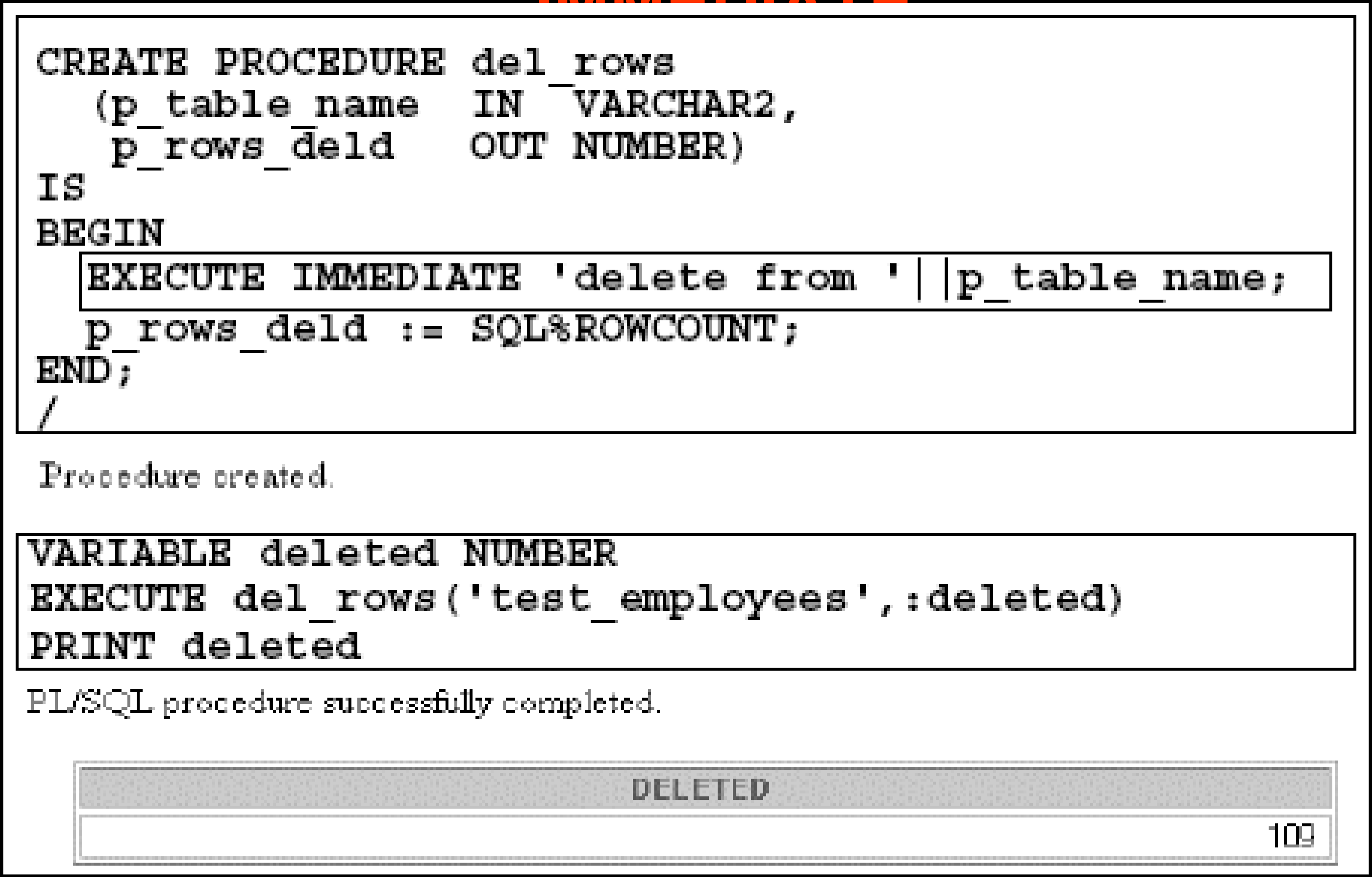
Use the EXECUTE IMMEDIATE statement for native dynamic SQL with better performance.

```
EXECUTE IMMEDIATE dynamic_string
        [INTO {define_variable
                 [, define_variable] ... | record}]
        [USING [IN|OUT|IN OUT] bind_argument
                 [, [IN|OUT|IN OUT] bind_argument] ... ];
```

- INTO is used for single-row queries and specifies the variables or records into which column values are retrieved.
- USING is used to hold all bind arguments. The default parameter mode is IN.

ORACLE

Dynamic SQL Using EXECUTE IMMEDIATE



```
CREATE PROCEDURE del_rows
   (p_table_name  IN  VARCHAR2,
    p_rows_deld   OUT NUMBER)
IS
BEGIN
   EXECUTE IMMEDIATE 'delete from '||p_table_name;
  p_rows_deld := SQL%ROWCOUNT;
END;
/
```

```
Procedure created.
```

```
VARIABLE deleted NUMBER
EXECUTE del_rows('test_employees',:deleted)
PRINT deleted
```

```
PL/SQL procedure successfully completed.
```

```
                    DELETED
                                              109
```

Dynamic SQL Using EXECUTE IMMEDIATE

```
CREATE TABLE EMP_T AS SELECT * FROM EMPLOYEES;
CREATE PROCEDURE del_rows
   (p_table_name IN VARCHAR2, p_rows_deld OUT NUMBER) IS
BEGIN
   EXECUTE IMMEDIATE 'delete from '||p_table_name;
   p_rows_deld := SQL%ROWCOUNT;
END;
/
VARIABLE deleted NUMBER
EXECUTE del_rows(‘EMP_T',:deleted)
PRINT deleted
```

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',‘HR',‘DEL_ROWS')
```

  - ANALYZE_OBJECT (object_type, owner, name, method) (COMPUTE, ESTIMATE, DELETE)

```
DBMS_DDL.ANALYZE_OBJECT('TABLE',‘HR',‘EMP_T','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

Submitting Jobs

You can submit jobs by using DBMS_JOB.SUBMIT.

Available parameters include:

- JOB OUT BINARY_INTEGER
- WHAT IN VARCHAR2
- NEXT_DATE IN DATE DEFAULT SYSDATE
- INTERVAL IN VARCHAR2 DEFAULT 'NULL'
- NO_PARSE IN BOOLEAN DEFAULT FALSE

```
CREATE OR REPLACE PACKAGE job_pack IS
PROCEDURE add_job (p_jobid IN jobs.job_id%TYPE,
                   p_jobtitle IN jobs.job_title%TYPE);
PROCEDURE upd_job (p_jobid IN jobs.job_id%TYPE,
                   p_jobtitle IN jobs.job_title%TYPE);
PROCEDURE del_job (p_jobid IN jobs.job_id%TYPE);
FUNCTION q_job (p_jobid IN jobs.job_id%TYPE)
    RETURN VARCHAR2;
END job_pack;
/
CREATE OR REPLACE PACKAGE BODY job_pack IS
PROCEDURE add_job (p_jobid IN jobs.job_id%TYPE,
                   p_jobtitle IN jobs.job_title%TYPE)  IS
BEGIN
  INSERT INTO jobs (job_id, job_title) VALUES (p_jobid, p_jobtitle);
END add_job;
…
```

```
PROCEDURE upd_job (p_jobid IN jobs.job_id%TYPE,
                        p_jobtitle IN jobs.job_title%TYPE) IS
BEGIN
  UPDATE jobs  SET job_title = p_jobtitle  WHERE job_id = p_jobid;
  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20202,'No job updated.');
  END IF;
END upd_job;
PROCEDURE del_job (p_jobid IN jobs.job_id%TYPE) IS
BEGIN
   DELETE FROM jobs   WHERE job_id = p_jobid;
   IF SQL%NOTFOUND THEN
      RAISE_APPLICATION_ERROR (-20203,'No job deleted.');
   END IF;
END del_job;
FUNCTION q_job (p_jobid IN jobs.job_id%TYPE) RETURN VARCHAR2 IS
   v_jobtitle jobs.job_title%TYPE;
BEGIN
   SELECT job_title   INTO v_jobtitle   FROM jobs   WHERE job_id = p_jobid;
   RETURN (v_jobtitle);
END q_job;
END job_pack;
/
```

Submitting Jobs

Use DBMS_JOB.SUBMIT to place a job to be executed in the job queue.

```
VARIABLE jobno NUMBER
BEGIN
    DBMS_JOB.SUBMIT (
    job => :jobno,
    what => 'JOB_PACK.ADD_JOB(''EDU'',''EDUCATION'');',
    next_date => TRUNC(SYSDATE + 1),
    interval => 'TRUNC(SYSDATE + 1)'
    );
    COMMIT;
END;
/
PRINT jobno
```

Changing Job Characteristics

- DBMS_JOB.CHANGE: Changes the WHAT, NEXT_DATE, and INTERVAL parameters
- DBMS_JOB.INTERVAL: Changes the INTERVAL parameter
- DBMS_JOB.NEXT_DATE: Changes the next execution date
- DBMS_JOB.WHAT: Changes the WHAT parameter

```
BEGIN
  DBMS_JOB.CHANGE(1, NULL, TRUNC(SYSDATE+1)+6/24, 'SYSDATE+4/24');
END;
```

Running, Removing, and Breaking Jobs

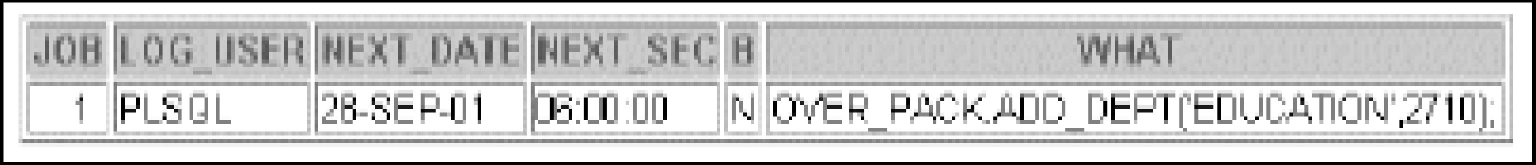
- DBMS_JOB.RUN: Runs a submitted job immediately
- DBMS_JOB.REMOVE: Removes a submitted job from the job queue
- DBMS_JOB.BROKEN: Marks a submitted job as broken, and a broken job will not run

```
EXECUTE  DBMS_JOB.RUN (1)
EXECUTE DBMS_JOB.BROKEN(1,TRUE)
EXECUTE DBMS_JOB.REMOVE(1)
```

Viewing Information on Submitted Jobs

- Use the DBA_JOBS dictionary view to see the status of submitted jobs.

```
SELECT job, log_user, next_date, next_sec,
broken, what
FROM DBA_JOBS;
```



JOB	LOG_USER	NEXT_DATE	NEXT_SEC	B	WHAT
1	PLSQL	28-SEP-01	06:00:00	N	OVER_PACK.ADD_DEPT('EDUCATION',2710);

- Use the DBA_JOBS_RUNNING dictionary view to display jobs that are currently running.

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

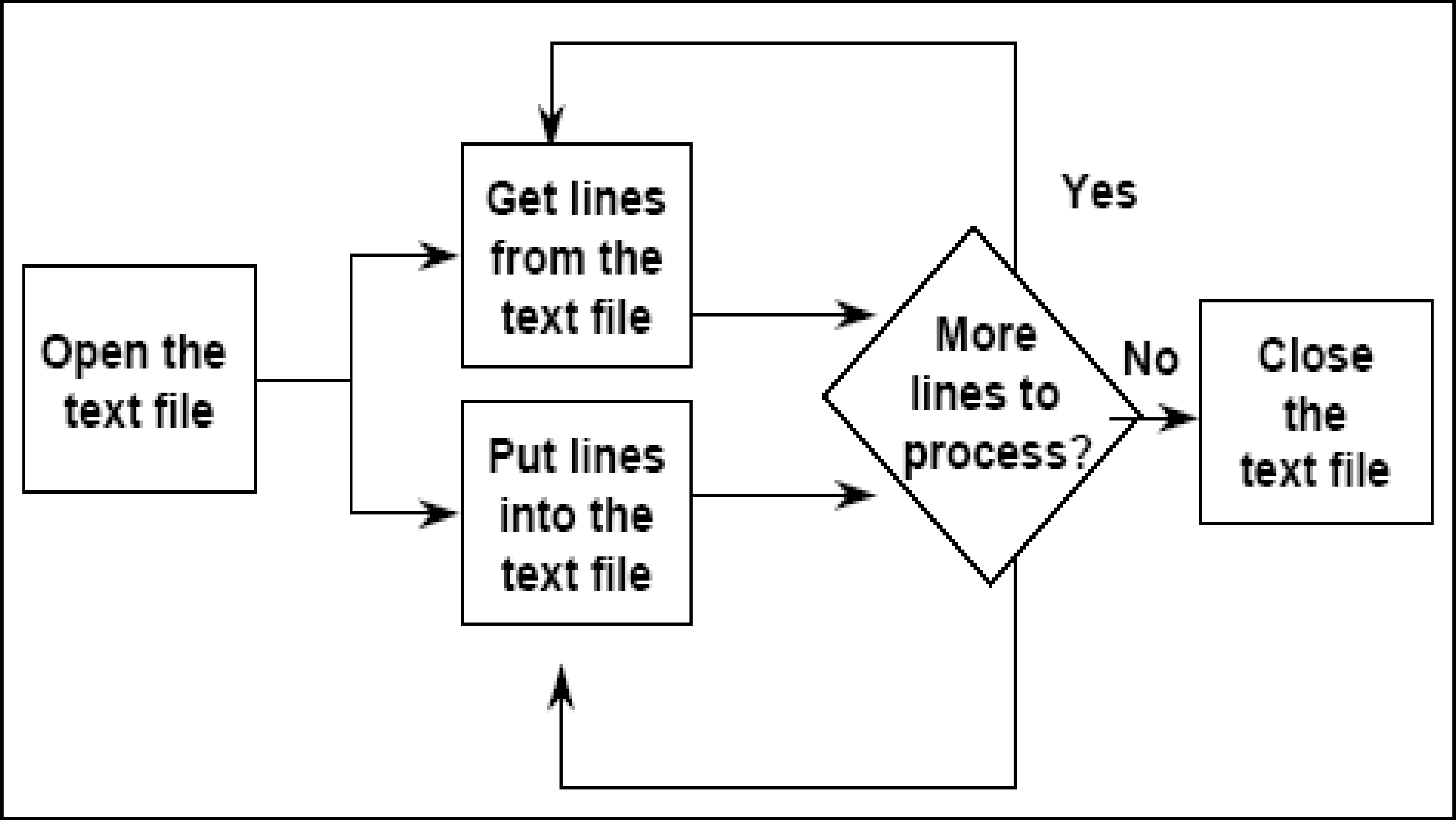
- UTL_FILE Oracle-supplied package:
  – Provides text file I/O capabilities
  – Is available with version 7.3 and later
- The DBMS_LOB Oracle-supplied package:
  – Provides read-only operations on external BFILES
  – Is available with version 8 and later
  – Enables read and write operations on internal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

```
CREATE OR REPLACE PROCEDURE cross_avgsal
                         (p_filedir IN VARCHAR2, p_filename1 IN VARCHAR2) IS
v_fh_1 UTL_FILE.FILE_TYPE;
CURSOR cross_avg IS
SELECT last_name, department_id, salary FROM employees outer
WHERE salary > (SELECT AVG(salary) FROM employees inner
                    GROUP BY outer.department_id)
ORDER BY department_id;
BEGIN
    v_fh_1 := UTL_FILE.FOPEN (p_filedir, p_filename1, 'w');
    UTL_FILE.PUTF (v_fh_1,'Employees with more than average salary:\n');
    UTL_FILE.PUTF (v_fh_1, 'REPORT GENERATED ON %s\n\n', SYSDATE);
    FOR v_emp_info IN cross_avg LOOP
         UTL_FILE.PUTF(v_fh_1, '%s %s \n',
         RPAD(v_emp_info.last_name, 35, ' '),
         LPAD(TO_CHAR(v_emp_info.salary, '$99,999.00'), 12, ' '));
    END LOOP;
    UTL_FILE.NEW_LINE (v_fh_1);
    UTL_FILE.PUT_LINE (v_fh_1, '*** END OF REPORT ***');
    UTL_FILE.FCLOSE (v_fh_1);
END cross_avgsal;
/
EXECUTE cross_avgsal('D:\ORACLE\UTL_FILE', 'sal_rptxx.txt')
```

File Processing Using 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

Exceptions Specific to the UTL_FILE Package

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

The FOPEN and IS_OPEN Functions

```
FUNCTION FOPEN
   ( location IN VARCHAR2,
     filename IN VARCHAR2,
     open_mode IN VARCHAR2)
RETURN UTL_FILE.FILE_TYPE;
```

```
FUNCTION IS_OPEN
   (file_handle IN FILE_TYPE)
RETURN BOOLEAN;
```

Using UTL_FILE

```
CREATE OR REPLACE PROCEDURE sal_status
    (p_filedir IN VARCHAR2, p_filename IN VARCHAR2) IS
    v_filehandle UTL_FILE.FILE_TYPE;
    CURSOR emp_info IS
    SELECT last_name, salary, department_id
    FROM employees  ORDER BY department_id;
    v_newdeptno employees.department_id%TYPE;
    v_olddeptno employees.department_id%TYPE := 0;
BEGIN
    v_filehandle := UTL_FILE.FOPEN (p_filedir, p_filename,'w');
    UTL_FILE.PUTF (v_filehandle,
                        'SALARY REPORT: GENERATED ON %s\n', SYSDATE);
    UTL_FILE.NEW_LINE (v_filehandle);
    FOR v_emp_rec IN emp_info LOOP
        v_newdeptno := v_emp_rec.department_id;
…
```

```
 IF v_newdeptno <> v_olddeptno THEN
   UTL_FILE.PUTF (v_filehandle,
                        'DEPARTMENT: %s\n',v_emp_rec.department_id);
 END IF;
 UTL_FILE.PUTF (v_filehandle,' EMPLOYEE: %s earns: %s\n',
                         v_emp_rec.last_name, v_emp_rec.salary);
 v_olddeptno := v_newdeptno;
END LOOP;
 UTL_FILE.PUT_LINE (v_filehandle, '*** END OF REPORT ***');
 UTL_FILE.FCLOSE (v_filehandle);
EXCEPTION
 WHEN UTL_FILE.INVALID_FILEHANDLE THEN
            RAISE_APPLICATION_ERROR (-20001, 'Invalid File.');
 WHEN UTL_FILE.WRITE_ERROR THEN
            RAISE_APPLICATION_ERROR (-20002, 'Unable to write to file');
END sal_status;
/
```

The UTL_HTTP Package

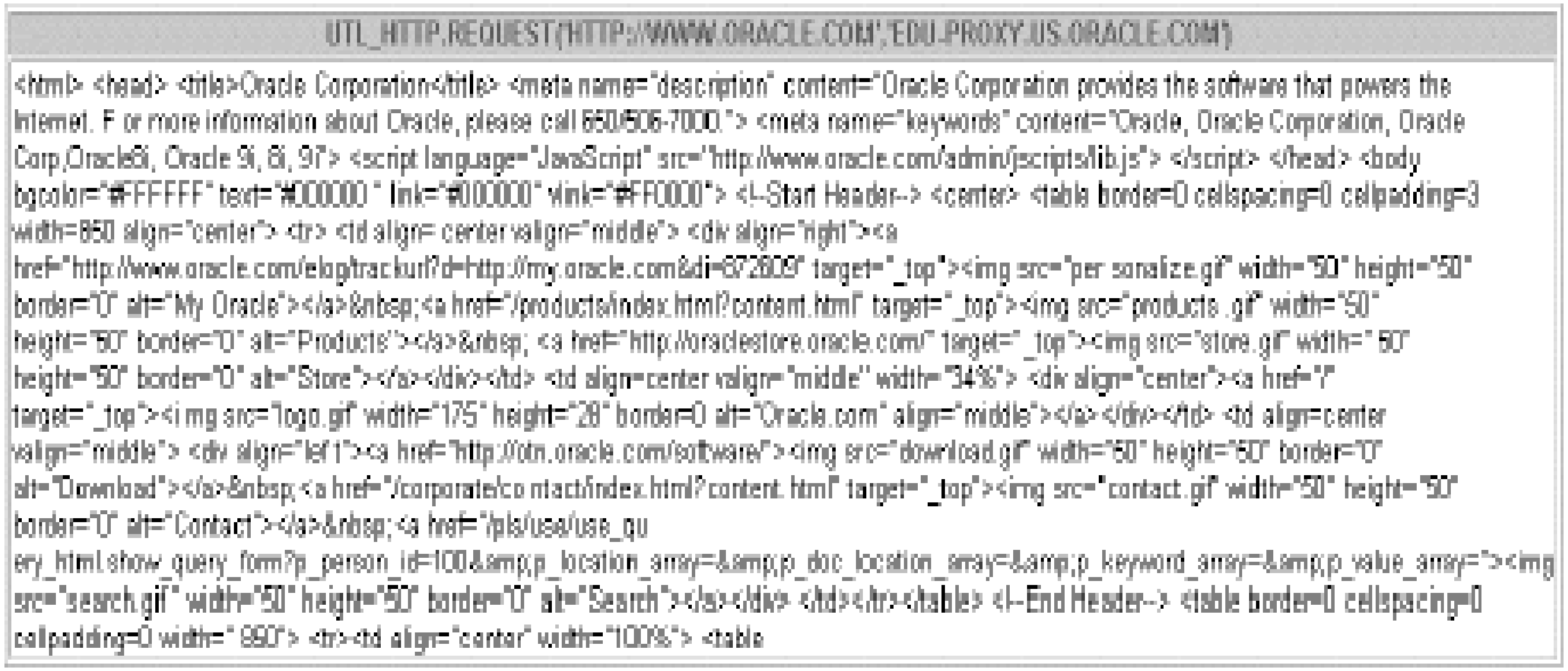
The UTL_HTTP package:

- 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_HTTP Package

```
SELECT UTL_HTTP.REQUEST('http://www.oracle.com',
                          'edu-proxy.us.oracle.com')

FROM DUAL;
```



```
UTL_HTTP.REQUEST('HTTP://WWW.ORACLE.COM','EDU-PROXY.US.ORACLE.COM')
<html> <head> <title>Oracle Corporation</title> <meta name="description" content="Oracle Corporation provides the software that powers the
Internet. F or more information about Oracle, please call 650/506-7000."> <meta name="keywords" content="Oracle, Oracle Corporation, Oracle
Corp,Oracle8i, Oracle 9i, 8i, 9i"> <script language="JavaScript" src="http://www.oracle.com/admin/jscripts/lib.js"> </script> </head> <body
bgcolor="#FFFFFF" text="#000000 " link="#000000" vlink="#FF0000"> <!-Start Header--> <center> <table border=0 cellspacing=0 cellpadding=3
width=850 align="center"> <tr> <td align= center valign="middle"> <div align="right"><a
href="http://www.oracle.com/elog/trackurl?d=http://my.oracle.com&di=672809" target="_top"><img src="per sonalize.gif" width="50" height="50"
border="0" alt="My Oracle"></a>&nbsp;<a href="/products/index.html?content.html" target="_top"><img src="products .gif" width="50"
height="50" border="0" alt="Products"></a>&nbsp; <a href="http://oraclestore.oracle.com/" target="_top"><img src="store.gif" width=" 50"
height="50" border="0" alt="Store"></a></div></td> <td align=center valign="middle" width="34%"> <div align="center"><a href="/"
target="_top"><i mg src="logo.gif" width="175" height="28" border=0 alt="Oracle.com" align="middle"></a></div></td> <td align=center
valign="middle"> <div align="lef t"><a href="http://otn.oracle.com/software/"><img src="download.gif" width="50" height="50" border="0"
alt="Download"></a>&nbsp;<a href="/corporate/co ntact/index.html?content. html" target="_top"><img src="contact.gif" width="50" height="50"
border="0" alt="Contact"></a>&nbsp;<a href="/pls/use/use_qu
ery_html.show_query_form?p_person_id=100&amp;p_location_array=&amp;p_doc_location_array=&amp;p_keyword_array=&amp;p_value_array="><img
src="search.gif" width="50" height="50" border="0" alt="Search"></a></div> </td></tr></table> <!-End Header--> <table border=0 cellspacing=0
cellpadding=0 width=" 990"> <tr><td align="center" width="100%"> <table
```

```
SET SERVEROUTPUT ON
DECLARE
        x UTL_HTTP.HTML_PIECES;
BEGIN
        x := UTL_HTTP.REQUEST_PIECES('http://www.yahoo.com/',100,
                                                        'http://mail.yahoo.com');
        DBMS_OUTPUT.PUT_LINE(x.COUNT || ' pieces were retrieved.');
        DBMS_OUTPUT.PUT_LINE('with total length ');
        IF x.COUNT < 1 THEN
           DBMS_OUTPUT.PUT_LINE('0');
        ELSE
          DBMS_OUTPUT.PUT_LINE((2000*(x.COUNT - 1))+LENGTH(x(x.COUNT)));
        END IF;
END;
/
```

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

Oracle-Supplied Packages

Other Oracle-supplied packages include:

- **DBMS_ALERT**
- **DBMS_APPLICATION_INFO**
- **DBMS_DESCRIBE**
- **DBMS_LOCK**
- **DBMS_SESSION**
- DBMS_SHARED_POOL
- DBMS_TRANSACTION
- DBMS_UTILITY

Summary

In this lesson, you should have learned how to:

- Take advantage of the preconfigured packages that are provided by Oracle
- Create packages by using the catproc.sql script
- Create packages individually.

Practice 14 Overview

This practice covers using:

- DBMS_SQL for dynamic SQL
- DBMS_DDL to analyze a table
- DBMS_JOB to schedule a task
- UTL_FILE to generate text reports