PARTICIPANT GUIDE

PROD
EDITION 1.0
JUL 95

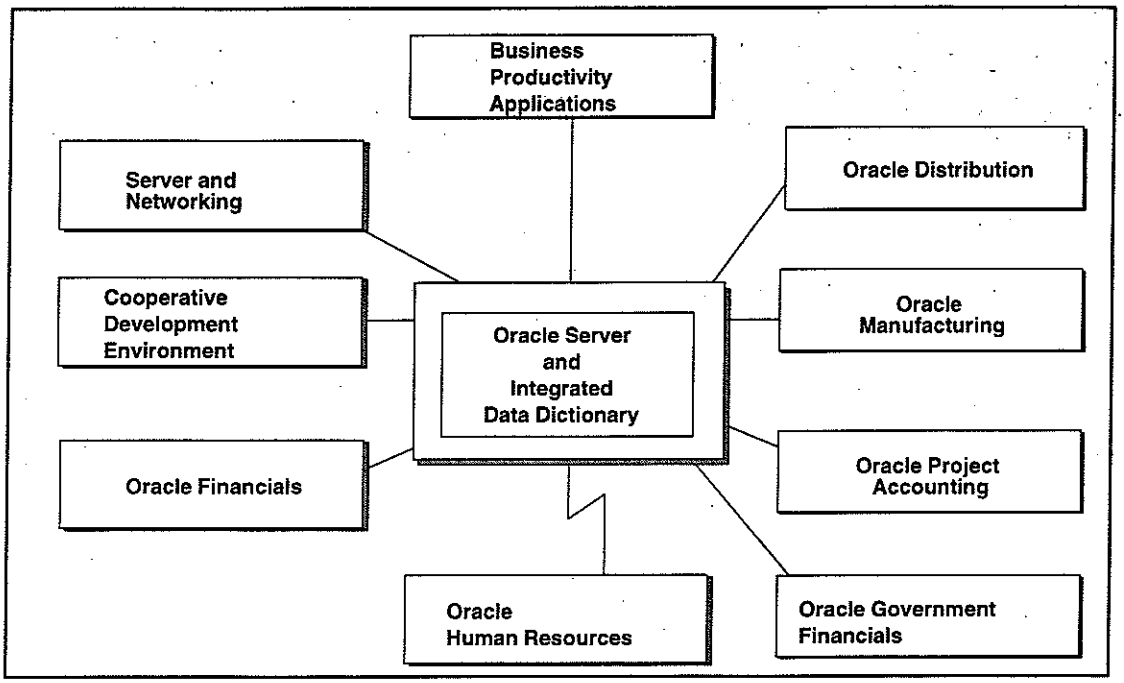
M02486

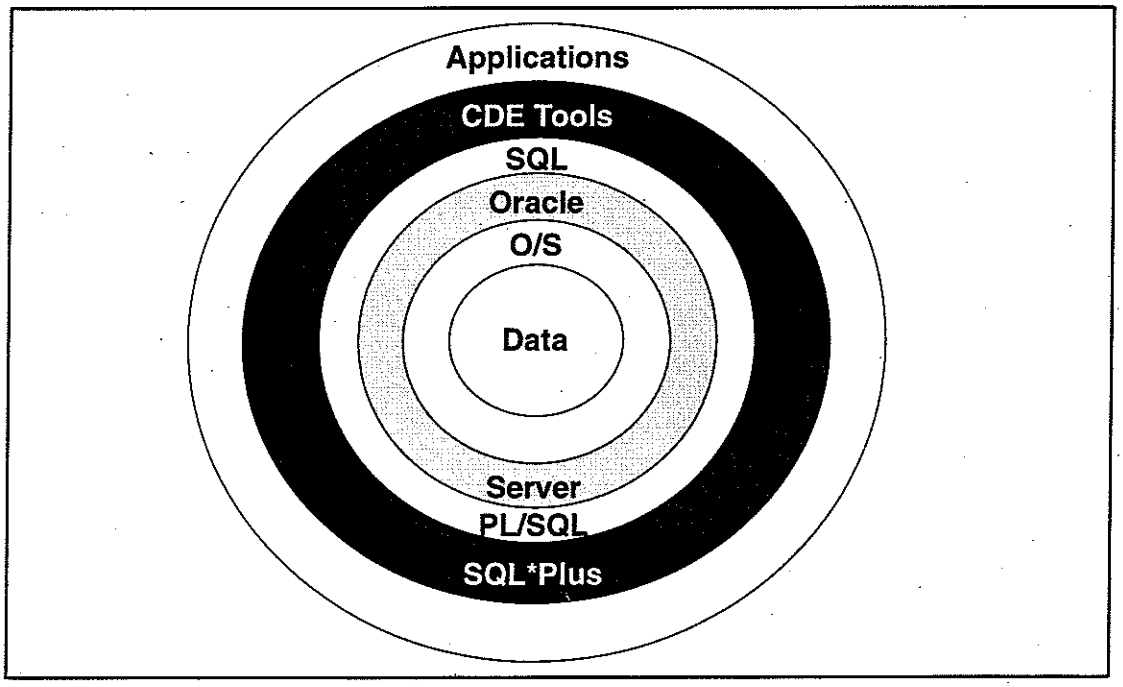
V7 DEVELOP WITH DATABASE PROCEDURES 7.2

Oracle PL/SQL programming
steve feuerstein
O'Reilly & Associates, Inc
ISBN 1-56592-142-9

ORACLE®

ORACLE


Business Productivity Applications
Server and Networking
Oracle Distribution
Cooperative Development Environment
Oracle Server and Integrated Data Dictionary
Oracle Manufacturing
Oracle Financials
Oracle Project Accounting
Oracle Human Resources
Oracle Government Financials


Applications
CDE Tools
SQL
Oracle
O/S
Data
Server
PL/SQL
SQL*Plus

Contents

Each of the predefined exceptions is listed in Table 8-1 along with its Oracle error number, the value returned by a call to SQLCODE, and a brief description. SQLCODE is a PL/SQL built-in function that returns the status code of the last-executed statement. SQLCODE returns zero if the last statement executed without errors. In most, but not all, cases, the SQLCODE value is the same as the Oracle error code.

Here is an example of how you might use the exceptions table. Suppose that your program generates an unhandled exception for error ORA–6511. Looking up this error, you find that it is associated with the CURSOR_ALREADY_OPEN exception. Locate the PL/SQL block in which the error occurs and add an exception handler for CURSOR_ALREADY_OPEN, as shown below:

```
EXCEPTION
   WHEN CURSOR_ALREADY_OPEN
   THEN
      CLOSE my_cursor;
END;
```

Of course, you would be even better off analyzing your code to determine proactively which of the predefined exceptions might occur. Then you could decide which of those exceptions you want to handle specifically, which should be covered by the WHEN OTHERS clause, and which would best be left unhandled.

Table 8-1. Predefined Exceptions in PL/SQL

Name of Exception Oracle Error/SQLCODE	Description
CURSOR_ALREADY_OPEN ORA–6511 SQLCODE= –6511	You tried to OPEN a cursor that was already OPEN. You must CLOSE a cursor before you try to OPEN or re-OPEN it.
DUP_VAL_ON_INDEX ORA–00001 SQLCODE= –1	Your INSERT or UPDATE statement attempted to store duplicate values in a column or columns in a row which is restricted by a unique index.
INVALID_CURSOR ORA–01001 SQLCODE= –1001	You made reference to a cursor that did not exist. This usually happens when you try to FETCH from a cursor or CLOSE a cursor before that cursor is OPENed.
INVALID_NUMBER ORA–01722 SQLCODE = –1722	PL/SQL executes a SQL statement that cannot convert a character string successfully to a number. This exception is different from the VALUE_ERROR exception, as it is raised only from within a SQL statement.
LOGIN_DENIED ORA–01017 SQLCODE= –1017	Your program tried to log onto the Oracle RDBMS with an invalid username-password combination. This exception is usually encountered when you embed PL/SQL in a 3GL language.
NO_DATA_FOUND ORA–01403 SQLCODE= +100	This exception is raised in three different scenarios: (1) You executed a SELECT INTO statement (implicit cursor) that returned no rows. (2) You referenced an uninitialized row in a local PL/SQL table. (3) You read past end of file with UTL_FILE package.

Table 8-1. Predefined Exceptions in PL/SQL (Continued)

Name of Exception Oracle Error/SQLCODE	Description
NOT_LOGGED_ON ORA–01012 SQLCODE= –1012	Your program tried to execute a call to the database (usually with a DML statement) before it had logged into the Oracle RDBMS.
PROGRAM_ERROR ORA–06501 SQLCODE= –6501	PL/SQL encounters an internal problem. The message text usually also tells you to "Contact Oracle Support."
STORAGE_ERROR ORA–06500 SQLCODE= –6500	Your program ran out of memory or memory was in some way corrupted.
TIMEOUT_ON_RESOURCE ORA–00051 SQLCODE= –51	A timeout occurred in the RDBMS while waiting for a resource.
TOO_MANY_ROWS ORA–01422 SQLCODE= –1422	A SELECT INTO statement returned more than one row. A SELECT INTO can return only one row; if your SQL statement returns more than one row you should place the SELECT statement in an explicit CURSOR declaration and FETCH from that cursor one row at a time.
TRANSACTION_BACKED_OUT ORA–00061 SQLCODE= –61	The remote part of a transaction is rolled back, either with an explicit ROLLBACK command or as the result of some other action.
VALUE_ERROR ORA–06502 SQLCODE= –6502	PL/SQL raises the VALUE_ERROR whenever it encounters an error having to do with the conversion, truncation, or invalid constraining of numeric and character data. This is a very general and common exception. If this same type of error is encountered in a SQL DML statement within a PL/SQL block, then the INVALID_NUMBER exception is raised.
ZERO_DIVIDE ORA–01476 SQLCODE= –1476	Your program tried to divide by zero.

Named Programmer-Defined Exceptions

The exceptions that PL/SQL has declared in the STANDARD package cover internal or system-generated errors. Many of the problems a user will encounter (or cause) in an application, however, are specific to that application. Your program might need to trap and handle errors such as "negative balance in account" or "call date cannot be in the past." While different in nature from "division by zero," these errors are still exceptions to normal processing and should be handled gracefully by your program.

One of the most useful aspects of the PL/SQL exception handling model is that it does not make any structural distinction between internal errors and application-specific errors. Once an exception is raised, it can and should be handled in the exception section, regardless of the type or source of error.