Error Management in Oracle PL/SQL

Steven Feuerstein
PL/SQL Evangelist, Quest Software
steven.feuerstein@quest.com
PL/SQL Obsession - www.ToadWorld.com/SF

How to benefit most from this training

- Watch, listen, ask questions, focus on concepts and principles.
- Download and use any of my training materials:

PL/SQL Obsession **http://www.ToadWorld.com/SF**

- Download and use any of my scripts (examples, performance scripts, reusable code) from the same location: the demo.zip file.

filename_from_demo_zip.sql

- You have my permission to use *all* these materials to do internal trainings and build your own applications.
  - But remember: they are not production ready.
  - You must test them and modify them to fit your needs.

And some other incredibly fantastic and entertaining websites for PL/SQL

Error Management in PL/SQL

- Defining exceptions
- Raising exceptions
- Handling exceptions
- Best practices for error management

- Let's start with some quizzes to test your knowledge of the basic mechanics of exception handling in PL/SQL.

Quiz: When strings don't fit...(1)

- What do you see after running this block?

```
DECLARE
   aname VARCHAR2(5);
BEGIN
   BEGIN
      aname := 'Big String';
      DBMS_OUTPUT.PUT_LINE (aname);
   EXCEPTION
      WHEN VALUE_ERROR
      THEN
         DBMS_OUTPUT.PUT_LINE ('Inner block');
   END;
   DBMS_OUTPUT.PUT_LINE ('What error?');
EXCEPTION
   WHEN VALUE_ERROR
   THEN
      DBMS_OUTPUT.PUT_LINE ('Outer block');
END;
```

excquiz1.sql

Quiz: When strings don't fit...(2)

- What do you see after running this block?

```
DECLARE
   aname VARCHAR2(5);
BEGIN
   DECLARE
      aname VARCHAR2(5) := 'Big String';
   BEGIN
      DBMS_OUTPUT.PUT_LINE (aname);

   EXCEPTION
      WHEN VALUE_ERROR
      THEN
         DBMS_OUTPUT.PUT_LINE ('Inner block');
   END;
   DBMS_OUTPUT.PUT_LINE ('What error?');
EXCEPTION
   WHEN VALUE_ERROR
   THEN
      DBMS_OUTPUT.PUT_LINE ('Outer block');
END;
/
```

excquiz2.sql

Quiz: SQL in PL/SQL

```
DECLARE
   v_totsal NUMBER;
   v_ename emp.ename%TYPE;
BEGIN
   SELECT SUM (sal) INTO v_totsal
     FROM emp WHERE deptno = -15;

   DBMS_OUTPUT.PUT_LINE (
      'Total salary: ' || v_totsal);

   SELECT ename INTO v_ename
     FROM emp
    WHERE sal =
       (SELECT MAX (sal)
          FROM emp WHERE deptno = -15);

   DBMS_OUTPUT.PUT_LINE (
      'The winner is: ' || v_ename);
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.PUT_LINE ('Outer block');
END;
/
```

- What do you see after running this block?

excquiz4.sql

Utter confusion of reused exceptions

```
CREATE OR REPLACE PROCEDURE who_did_that (
   emp_in IN emp.empno%TYPE)
IS
   v_ename          emp.ename%TYPE;
   line             VARCHAR2 (1023);
   fid              UTL_FILE.file_type;
   list_of_names    DBMS_SQL.varchar2s;
BEGIN
   SELECT ename INTO v_ename FROM emp
    WHERE empno = emp_in;

   DBMS_OUTPUT.put_line (v_ename);
   fid := UTL_FILE.fopen ('c:\temp', 'notme.sql', 'R');
   UTL_FILE.get_line (fid, line);
   UTL_FILE.get_line (fid, line);

   IF list_of_names (100) > 0
   THEN
      DBMS_OUTPUT.put_line ('Pos value at 100');
   END IF;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.put_line ('Who did that?');
END who_did_that;
/
```

- How do you where the exception was raised?

excquiz6.sql

Quiz: An Exceptional Package

```
PACKAGE valerr
IS
  FUNCTION
    get RETURN VARCHAR2;
END valerr;

PACKAGE BODY valerr
IS
   v VARCHAR2(1) := 'abc';
   FUNCTION get RETURN VARCHAR2 IS
   BEGIN
      RETURN v;
   END;
BEGIN
   DBMS_OUTPUT.PUT_LINE (
     'Before I show you v...');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE (
     'Trapped the error!');
END valerr;
```

```
SQL> EXECUTE p.l (valerr.get);
```

- I create the valerr package and then execute the command below. What is displayed on the screen?

- **Key to remember: even if package initialization fails, Oracle marks the package as *initialized*.**

valerr.pkg
valerr2.pkg

Defining Exceptions

- The EXCEPTION is a limited type of data.
  - Has just two attributes: code and message.
  - You can RAISE and handle an exception, but it cannot be passed as an argument in a program.
- Give names to error numbers with the EXCEPTION_INIT PRAGMA.

```
CREATE OR REPLACE PROCEDURE upd_for_dept (
   dept_in   IN   employee.department_id%TYPE
 , newsal_in IN   employee.salary%TYPE
)
IS
   bulk_errors   EXCEPTION;
   PRAGMA EXCEPTION_INIT (bulk_errors, -24381);
```

Raising Exceptions

- RAISE raises the specified exception by name.
  - RAISE; re-raises current exception. Callable only within the exception section.
- RAISE_APPLICATION_ERROR
  - Communicates an application specific error back to a non-PL/SQL host environment.
  - Error numbers restricted to the -20,999 - -20,000 range.

Using RAISE_APPLICATION_ERROR

```
RAISE_APPLICATION_ERROR (
   num binary_integer
 , msg varchar2
 , keeperrorstack boolean default FALSE
);
```

- Communicate an error number and message to a non-PL/SQL host environment.
  - The following code from a database triggers shows a typical (and problematic) usage of RAISE_APPLICATION_ERROR:

```
IF :NEW.birthdate > ADD_MONTHS (SYSDATE, -1 * 18 * 12)
THEN
   RAISE_APPLICATION_ERROR
      (-20070, 'Employee must be 18.');
END IF;
```

Handling Exceptions

- The EXCEPTION section consolidates all error handling logic in a block.
  - But only traps errors raised in the executable section of the block.
- Several useful functions usually come into play:
  - SQLCODE and SQLERRM
  - DBMS_UTILITY.FORMAT_CALL_STACK
  - DBMS_UTILITY.FORMAT_ERROR_STACK
  - DBMS_UTILITY.FORMAT_ERROR_BACKTRACE
- The DBMS_ERRLOG package
  - Quick and easy logging of DML errors

SQLCODE and SQLERRM

- SQLCODE returns the error code of the most recently-raised exception.
  - You cannot call it inside an SQL statement (even inside a PL/SQL block).
- SQLERRM is a generic lookup function: return the message text for an error code.
  - And if you don't provide an error code, SQLERRM returns the message for SQLCODE.
- But....SQLERRM might truncate the message.
  - Very strange, but it is possible.
  - So Oracle recommends that you *not* use this function.

sqlerrm.sql

DBMS_UTILITY error functions

- Answer the question "How did I get here?" with DBMS_UTILITY.FORMAT_CALL_STACK.
- Get the full error message with DBMS_UTILITY.FORMAT_ERROR_STACK
  - It will not truncate your error message.
  - And it might even return a stack!
- Find line number on which error was raised with DBMS_UTILITY.FORMAT_ERROR_BACKTRACE
  - Introduced in Oracle10g Release 2, it returns the full stack of errors with line number information.
  - Formerly, this stack was available only if you let the error go unhandled.

callstack.sql & callstack.pkg
errorstack.sql
backtrace.sql

More on the BACKTRACE function

- When you re-raise your exception (RAISE;) or raise a different exception, subsequent BACKTRACE calls will point to *that* line.
  - So before a re-raise, call BACKTRACE and store that information to avoid losing the original line number.
- The BACKTRACE does not include the error message, so you will also want to call the FORMAT_ERROR_STACK function.

backtrace.sql
bt.pkg

Continuing Past Exceptions

- What if you want to continue processing in your program even if an error has occurred?
- Three options...
  - Use a nested block
  - FORALL with SAVE EXCEPTIONS
  - DBMS_ERRLOG

continue_past_exception.sql

Exception handling and FORALL

- When an exception occurs in a DML statement....
  - That statement is rolled back and the FORALL stops.
  - All (previous) successful statements are *not* rolled back.
- Use the SAVE EXCEPTIONS clause to tell Oracle to continue past exceptions, and save the error information for later.
- Then check the contents of the pseudo-collection of records, SQL%BULK_EXCEPTIONS.
  - Two fields: ERROR_INDEX and ERROR_CODE

FORALL with SAVE EXCEPTIONS

- Add SAVE EXCEPTIONS to enable FORALL to suppress errors at the *statement* level.

```
CREATE OR REPLACE PROCEDURE load_books (books_in IN book_obj_list_t)
IS
  bulk_errors EXCEPTION;
  PRAGMA EXCEPTION_INIT ( bulk_errors, -24381 );
BEGIN
  FORALL indx IN books_in.FIRST..books_in.LAST
    SAVE EXCEPTIONS
    INSERT INTO book values (books_in(indx));
EXCEPTION
   WHEN bulk_errors THEN
      FOR indx in 1..SQL%BULK_EXCEPTIONS.COUNT
      LOOP
         log_error (SQL%BULK_EXCEPTIONS(indx).ERROR_CODE);
      END LOOP;
END;
```

Allows processing of all statements, even after an error occurs.

Iterate through pseudo-collection of errors.

bulkexc.sql

DBMS_ERRLOG (new in Oracle10gR2)

- Use this package to enable row-level error logging (and exception suppression) for DML statements.
  - Compare to FORALL SAVE EXCEPTIONS, which suppresses exceptions at the *statement* level.
- Creates a log table to which errors are written.
  - Lets you specify maximum number of "to ignore" errors.
- Better performance than trapping, logging and continuing past exceptions.
  - Exception handling is *slow*.

dbms_errlog.sql
dbms_errlog_helper.pkg
dbms_errlog_vs_save_exceptions.sql

Best practices for error management

- No application code in the exception section.
- Avoid suppression of exceptions.
  - Especially WHEN OTHERS THEN NULL;
- Do not hard-code error numbers, messages and error logging mechanics.
- Set standards for error mgt code before you start building your application.
- Make sure all developers rely on a single error management utility.

No application code in the exc. section

- Two common scenarios cause this:
  - Deliberately raised exceptions
  - "Unfortunate" exceptions
- Two ways to avoid this problem:
  - Encapsulate the code that raises an exception
  - Give the caller of your code the option to decide if it is an exception or simply a different data condition.

exec_ddl_from_file_badnews.sql
raise_if_ndf.sql

Avoid suppression of exceptions

- The "I don't care" exception section
- Maybe you really don't care, but whenever you trap and do not re-raise an exception, you should....
  - Consider logging the error for later analysis
  - At least include a comment explaining why an exception should be ignored.

```
EXCEPTION
   WHEN OTHERS
   THEN
      NULL;
END;
```

```
/*
Remove table if it exists,
otherwise just keep going.
*/
BEGIN
   EXECUTE IMMEDIATE
      'DROP TABLE temptab';
EXCEPTION
   WHEN OTHERS
   THEN
      NULL;
END;
```

Do not hard-code error codes, messages and error logging mechanics

```
WHEN NO_DATA_FOUND
THEN
   l_code := SQLCODE; l_errm := SQLERRM;
   INSERT INTO errlog VALUES ( l_code
             , 'No company for id ' || TO_CHAR ( v_id )
             , 'fixdebt', SYSDATE, USER );
WHEN DUP_VAL_ON_INDEX
THEN
   RAISE_APPLICATION_ERROR (-20984, 'Company with name already exists');
WHEN OTHERS
THEN
   l_code := SQLCODE; l_errm := SQLERRM;
   INSERT INTO errlog
      VALUES (l_code, l_errm, 'fixdebt', SYSDATE, USER );
   RAISE;
END;
```

- Let's find all the hard-codings....
- What a mess of code!

If you must use RAISE_APPLICATION_ERROR...

- Avoid hard-coding error numbers and messages.
  - And don't pull those -20NNN numbers out of thin air!
- Instead, build a repository of errors and generate code that offers names for errors for developers to use.
- Better yet: don’t use RAISE_APPLICATION_ERROR at all.

msginfo.*

Rely on a single error mgt utility

- Objectives of this utility:
  - Everyone writes error management code in the same way.
  - Minimize time spent by application developers in the exception section.
  - Make it easy to gather comprehensive system and application *clues* as to the source of the problem.
- How do we accomplish this?
  - Fix the limitations of the EXCEPTION datatype.
  - Provide procedures and functions that do the work for you.
- I will use the Quest Error Manager as the model for such a utility.

The Quest Error Manager solution

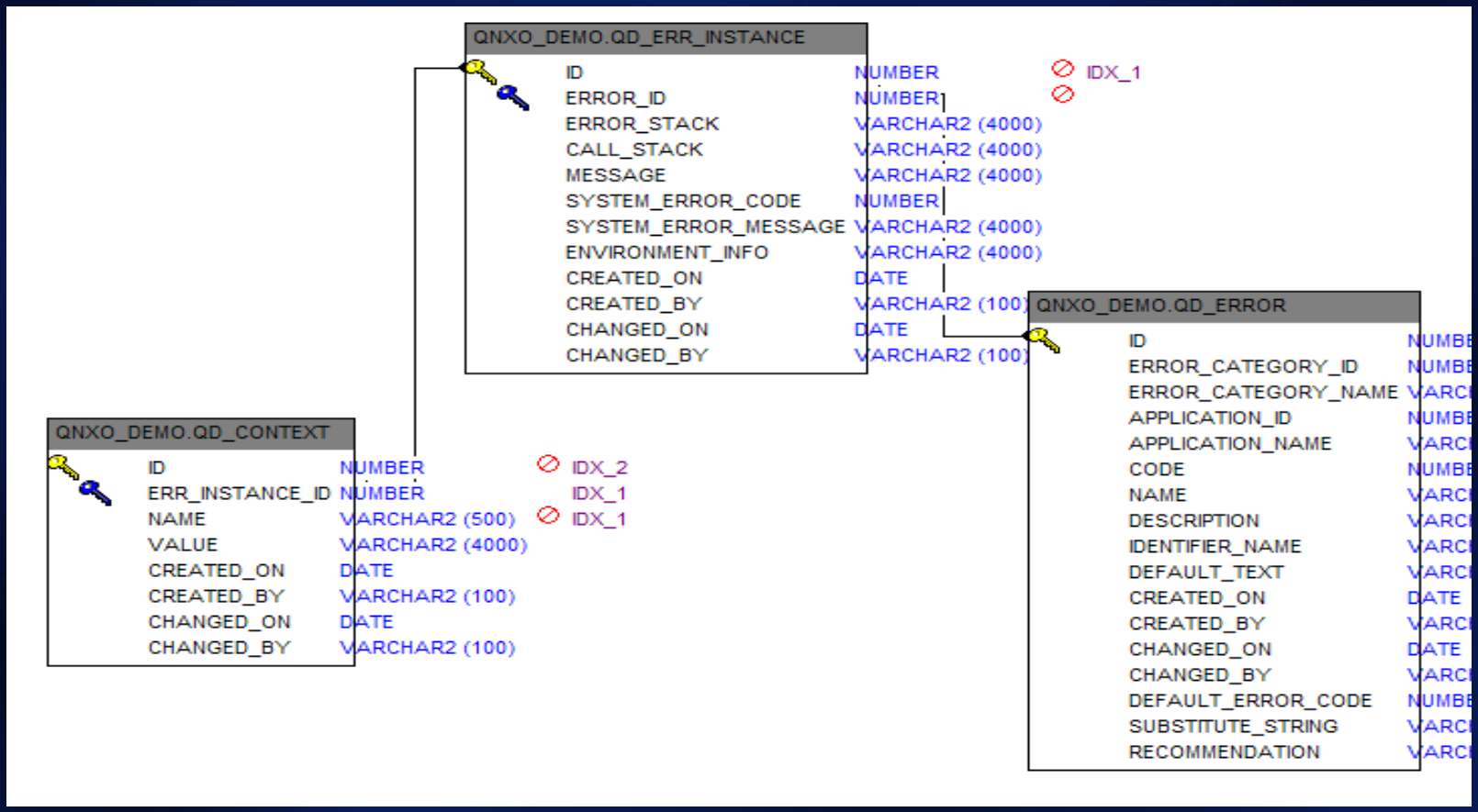
- This freeware utility (see URL below) implements all of the ideas mentioned above.
  - Error instance table-based repository, with ability to store as much context as needed.
- High-level API that allows all developers to easily…
  - Raise and handle exceptions
  - Communicate error information back to the host environment.

Fix the limitations of EXCEPTION

- When an error occurs....
  - Sure, it's nice to know what the error code is.
  - But what I care most about is what *caused* this particular error to be raised.
- Think in terms of *instances* of an error.
  - What caused this error?
  - What were the application-specific values or context in which the error occurred?
- So the challenge becomes: how do I get hold of and save all that critical application information?

ERD for error definition tables

- Error instance - one row for every error raised/logged
- Error context - name-value pairs per instance
- Error repository - static error definitions



Quest Error Manager API

- REGISTER_ERROR
  - Register error instance, return handle.
- RAISE_ERROR
  - Register the error, and then re-raise the exception to stop the calling program from continuing.
- ADD_CONTEXT
  - Add a name-value pair to an error instance.
- GET_ERROR_INFO
  - Retrieve information about latest (or specified) error.
- Plus...tracing, substitutes for DBMS_OUTPUT, assertions.

Applying the Error Manager API

- The idea is to let the application developers specify the information that only *they* can provide.
  - Everything else is left to the utility.
- All information is written out to the tables.
- The front end or support team can then retrieve the data.
  - Provide better information to the users.
  - Make it easier to figure out what went wrong and how to fix the problem.

qem_demo.sql

Error Management Summary - 1

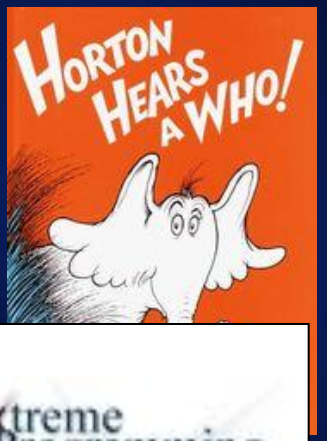
- Exceptions raised in the declaration section always escape unhandled.
  - Consider assigning default values in the executable section instead.
- Call DBMS_UTILITY functions whenever you are logging errors and tracing execution.
- Consider using DBMS_ERRLOG when executing a large number of DML statements.
  - But don't use the ERR$ table "as is".

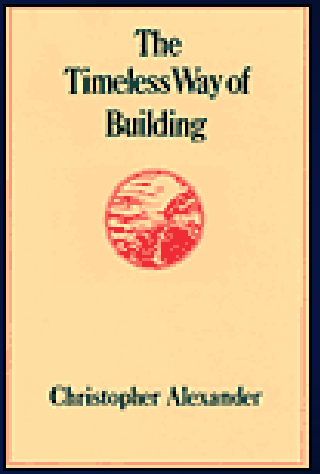
Error Management Summary - 2

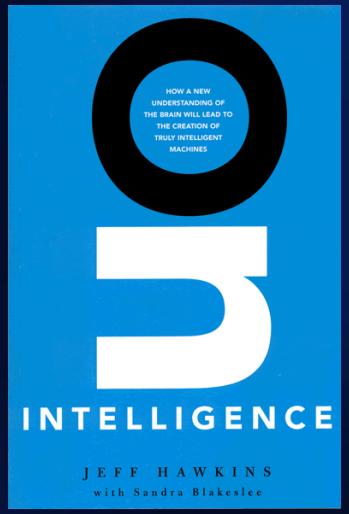
- Set standards before you start coding
  - It's not the kind of thing you can easily add in later
- Use standard infrastructure components
  - Everyone and all programs need to handle errors the same way
  - Quest Error Manager offers a good example.
- Don't accept the limitations of Oracle's current implementation.
  - You can do lots to improve the situation.

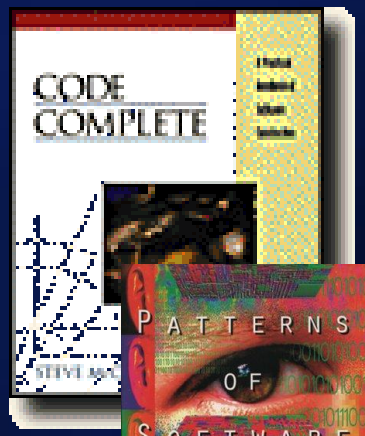
Acknowledgements and Resources

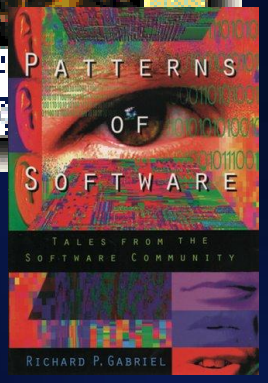
- **Very few of my ideas are truly original. I have learned from every one of these books and authors – and you can, too!**

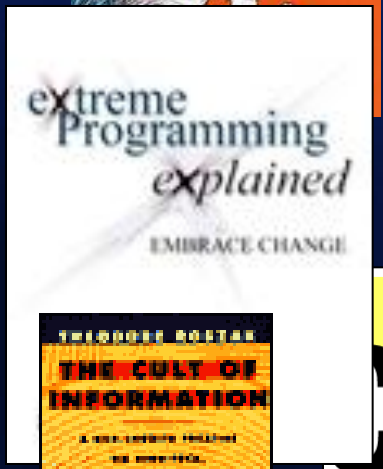


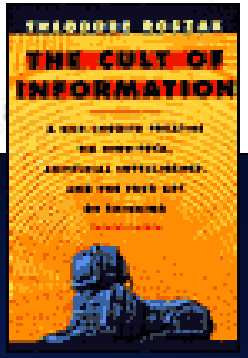


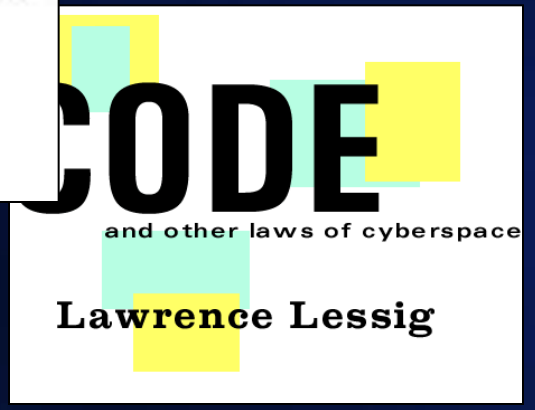


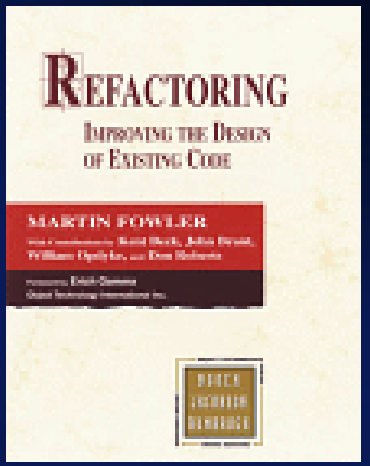












A guide to my mentors/resources

- **Horton Hears a Who** - if you have small children and you have not read this book to them, please do so immediately! A story of basic human compassion.
- **A Timeless Way of Building** – a beautiful and deeply spiritual book on architecture that changed the way many developers approach writing software.
- **On Intelligence** – a truly astonishing book that lays out very concisely a new paradigm for understanding how our brains work.
- **Peopleware** – a classic text on the human element behind writing software.
- **Refactoring** – formalized techniques for improving the internals of one's code without affect its behavior.
- **Code Complete** – another classic programming book covering many aspects of code construction.
- **The Cult of Information** – thought-provoking analysis of some of the down-sides of our information age.
- **Patterns of Software** – a book that wrestles with the realities and problems with code reuse and design patterns.
- **Extreme Programming Explained** – excellent introduction to XP.
- **Code and Other Laws of Cyberspace** – a groundbreaking book that recasts the role of software developers as law-writers, and questions the direction that software is today taking us.

Some Free PL/SQL Resources

- Oracle Technology Network PL/SQL page

  http://www.oracle.com/technology/tech/pl_sql/index.html

- OTN Best Practice PL/SQL

  http://www.oracle.com/technology/pub/columns/plsql/index.html

- Oracle documentation – complete, online, searchable!

  http://tahiti.oracle.com/

- PL/SQL Obsession - my on-line portal for PL/SQL developers

  http://www.ToadWorld.com/SF

- Quest Pipelines

  http://quest-pipelines.com/

- I Love PL/SQL and...help improve the PL/SQL language!

  http://ILovePLSQLAnd.net

- PL/Vision

  http://quest-pipelines.com/pipelines/dba/PLVision/plvision.htm