Writing Sweet PL/SQL !

Contact Info:
Bill Coulam
bcoulam@yahoo.com

Our Agenda Today

- History, best use and future of PL/SQL
- Anti-Patterns
- Patterns
- Practices


Future of PL/SQL
Best Uses of PL/SQL
Brief History of PL/SQL

Brief History of PL/SQL

- Similar to Ada and Pascal
- Oracle added to provide a procedural option around execution of SQL
- v1 with Oracle 6 in 1991.
- v2 added stored routines
- v2.1 dynamic SQL
- v2.3 added arrays
- v8 bulk binds, autonomous tx
- v9 record-based DML, CASE
- v10 regexp, conditional compilation
- v11 function cache

Best Uses of PL/SQL

- Processing close to the data = fast!
- Backend, timed or event-driven transformations, calculations or processing of large data
- Consumption or production of data-filled files
- Ensuring data access layer is free of SQL injection
- When a trigger is the right choice
- Ensuring business logic is kept in one place
- Data access: DBA review, tuned, instrumented, monitored, easy to modify and redeploy

Future of PL/SQL

- NoSQL is the buzzword of the day, but its uses are limited
- PL/SQL like COBOL & C: too useful, too widespread, too ingrained to disappear soon
- Adopted by IBM in DB2 9.7
- PostresSQL made their pgPLSQL look alike
- Oracle APEX is pure PL/SQL
- Very likely Oracle will add to MySQL

What is an Anti-Pattern?

“Something that looks like a good idea, but which backfires badly when applied.” - Jim Coplien

"Anti-patterns are Negative solutions that present more problems than they address." - antipatterns.com

By understanding how NOT to do something, we can prevent it from happening again and recovering when it was mistakenly applied.

Anti-Pattern 1

Too Many Cooks in the Kitchen

Anti-Pattern 2

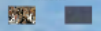
The Dung Beetle Ball

Anti-Pattern 3

Stonehenge

Anti-Pattern 4

Rabbits and Tribbles

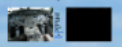


Anti-Pattern 5

Mona Lisa at the Mall

Anti-Pattern 6

Flying Blind



Anti-Patterns of Poor PL/SQL	Patterns of Great PL/SQL
Too Many Cooks in the Kitchen	Stick to a Standard
The Dung Beetle Ball	Keep it Simple
Stonehenge	Document the Interface
Rabbits and Tribbles	Everything Needs a Home
Mona Lisa at the Mall	Version the Source Code
Flying Blind	Instrument to Illuminate

Anti-Pattern 1

Too Many Cooks in the Kitchen

- Everyone's got their hand in the pot
- Everyone thinks they're a great chef; no SPOA
- Everyone adds favorite spices, ruining the dish
- Eventually the code is so tangled, decorated and inconsistent, that it sours, becoming too costly to keep it in its current state.

```
CREATE OR replace PACKAGE BODY STR
/*****************************************************************************
Bill Coulam (bcoulam@dbartisans.com)

Developer    DATE      Comments
============ ========= ============================================================
bcoulam      1997Dec30 Creation
SHAWN NIELSEN 19980305   Added str_to_bool
Daws, Frank  August 8, 2002  Added get_diacritic_list
EricH   2004 MAR 15    TICKET 8840: Created get_diacritic_map
BarnesHoward 2006Oct30 TICKET 9984: created foreign_to_ascii
BarnesHoward 2008May18 TICKET 10334: fixed foreign_to_ascii to not raise exceptions if the user passes in nothing
bcoulam      2012May22 Added TRIM to last element assignment in parse_list()
*****************************************************************************/
AS

FUNCTION bool_to_str(boolean_value                  IN   BOOLEAN)    RETURN VARCHAR2
IS
BEGIN
   IF (boolean_value) THEN RETURN 'TRUE';
   ELSIF (boolean_value IS NULL) THEN RETURN 'NULL';
   ELSE RETURN 'FALSE';
   END IF;
END bool_to_str;

function str_to_bool(                                   -- start of function
 p_str varchar2                                         -- incoming string parameter
) return boolean                                        -- return TRUE or FALSE
is
BEGIN                                                   -- start of block
   if (lower(p_str) in ('true','y','yes','t','1'))      -- convert parameter to lowercase and compare to likely postive valu
   then

    return true;

   elsif (lower(p_str) in ('false','n','no','f','0'))   -- convert parameter to lowercase and compare to likely negative val
   then

    return false;

   elsif (p_str IS null)                                -- if parameter is empty, that's OK, return empty silently
   then

    return null;
```

```
/*****************************************************************************/
FUNCTION get_diacritic_list
RETURN VARCHAR2
IS
BEGIN
   RETURN 'ÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏ';
END get_diacritic_list;

      FUNCTION get_diacritic_map
      return VARCHAR2
      IS
      BEGIN
         RETURN 'DNOOOOOxOUUUUUYPBaaaaaaaceeeeiiiionooooo-ouuuuuypyAAAAAAACEEEEIIII';
          end get_diacritic_map;
/*****************************************************************************/

                    ------------------------------------
FUNCTION foreign_to_ascii (
   i_str    IN VARCHAR2
)  RETURN    VARCHAR2
IS
BEGIN
/** Check to make sure the i_str parameter has a value; it's OK if it does not, but
    don't return anything either. It's not an error to not have any value.
**/
   IF (i_str IS NULL) THEN
   -- TICKET 10334: BEGIN, howardb 2008May18 03:04pm
      RETURN NULL;
   -- TICKET 10334: END, howardb 2008May18 03:07pm
   END IF;

/**
TICKET 8840: BEGIN EricH -- Sorta removing while we figure out if it's gonna work
    as a function instead of a hard-coded literal
   RETURN TRANSLATE(i_str, get_diacritic_list, 'DNOOOOOxOUUUUUYPBaaaaaaaceeeeiiiionooooo-ouuuuuypyAAAAAAACEEEEIIII');
TICKET 8840: END EricH
**/
   RETURN TRANSLATE(i_str,
          get_diacritic_list
         ,/** TICKET 8840 BEGIN EricH **/ get_diacritic_map /** TICKET 8840 END EricH **/);
END foreign_to_ascii;

/*****************************************************************************/
FUNCTION nonascii_to_ascii (
```

```
FUNCTION format_to_width
(
   i_str         IN VARCHAR2,
   i_width       IN INTEGER DEFAULT cnst.pagewidth,
   p_allow_wrap IN VARCHAR2 DEFAULT 'Y'
) RETURN VARCHAR2
IS
   v_ostr VARCHAR2(32767); -- o[riginal]str[ing]
   v_nstr VARCHAR2(32767); -- n[ew]str[ing]
   v_piece VARCHAR2(32767);
   l_length PLS_INTEGER := 0;
   l_curr_pos PLS_INTEGER := 1; -- current position
   l_width PLS_INTEGER := 0;
   l_next_para PLS_INTEGER := 0;
   l_next_space PLS_INTEGER := 0;
   l_next_dash PLS_INTEGER := 0;
   DASH CHAR(1) := CHR(45);
   PARA  CHAR(1) := CHR(126);
BEGIN
   -- Check odd conditions first where the string needs no processing
   IF (i_str IS NULL) THEN
      RETURN NULL;
   END IF;

   l_length := LENGTH(i_str);

   -- convert paragraphs so they can be turned back into paragraphs after
   -- LF and CR replacement.
   v_ostr := REPLACE(
               REPLACE(
                  REPLACE(
                     REPLACE(
                        REPLACE(i_str, LF||LF, PARA)
                     , CR||LF, PARA)
                  , LF||CR, PARA)
               , LF, SP)
            , CR, SP);

   ...
   END format_to_width;

   -- Many more routines snipped to shorten image
```

Solution

Stick to a Standard

- Establish a programming and style standard
- Ensure the standard is adhered to
- Use templates, automation and formatting tools to make it easy

- Establish a programming and style standard
- Ensure the standard is adhered to
- Use templates, automation and formatting tools to make it easy

www.toadworld.com/Experts/StevenFeuersteinsPLSQLObsession/tabid/153/Default.aspx

Most Visited | Pandora | Gmail | Yahoo! Mail | Amazon | Jira | LiveBookmarks | Oracle | Tech | Natural | Edutainment | Manliness | News and Weather | LDS Church | Religion | (2) Google+ | Home | Oracle Optimizer

Steven Feuerstein's
PL/SQL Obsession

I, Steven Feuerstein, am **obsessed** with the PL/SQL language. How else to explain my life since 1994: ten books written about the Oracle PL/SQL language, visits to some 20 countries to train developers on how best to utilize PL/SQL, winner of the coveted Oracle Magazine "PL/SQL Developer of the year award" twice, PL/SQL Evangelist for Quest Software since 2001, writing daily quizzes for the PL/SQL Challenge since April 2010, recorded over 27 hours of video training on PL/SQL at the PL/SQL Channel....yes, my friends, this is *definitely* an obsession.

And you, dear visitor, can *benefit* from that obsession right here in Toad World, where I have collected together lots of great resources on PL/SQL. Plus, you might even get some answers to burning questions that have troubled readers of my books for years, such as: "How do you pronounce your last name?" Oh, all right. The answer is (drum roll, please):**FOYER-STEEN**. And "How can I get in touch with Steven?" Just send me an email!

PL/SQL Education

- Practical Best PL/SQL Video Series
- Trainings, Seminars, and Presentations
- PL/SQL Standards
- PL/SQL Puzzles & Quizzes
- PL/SQL Tips
- Steven Feuerstein's PL/SQL Books

Blogs

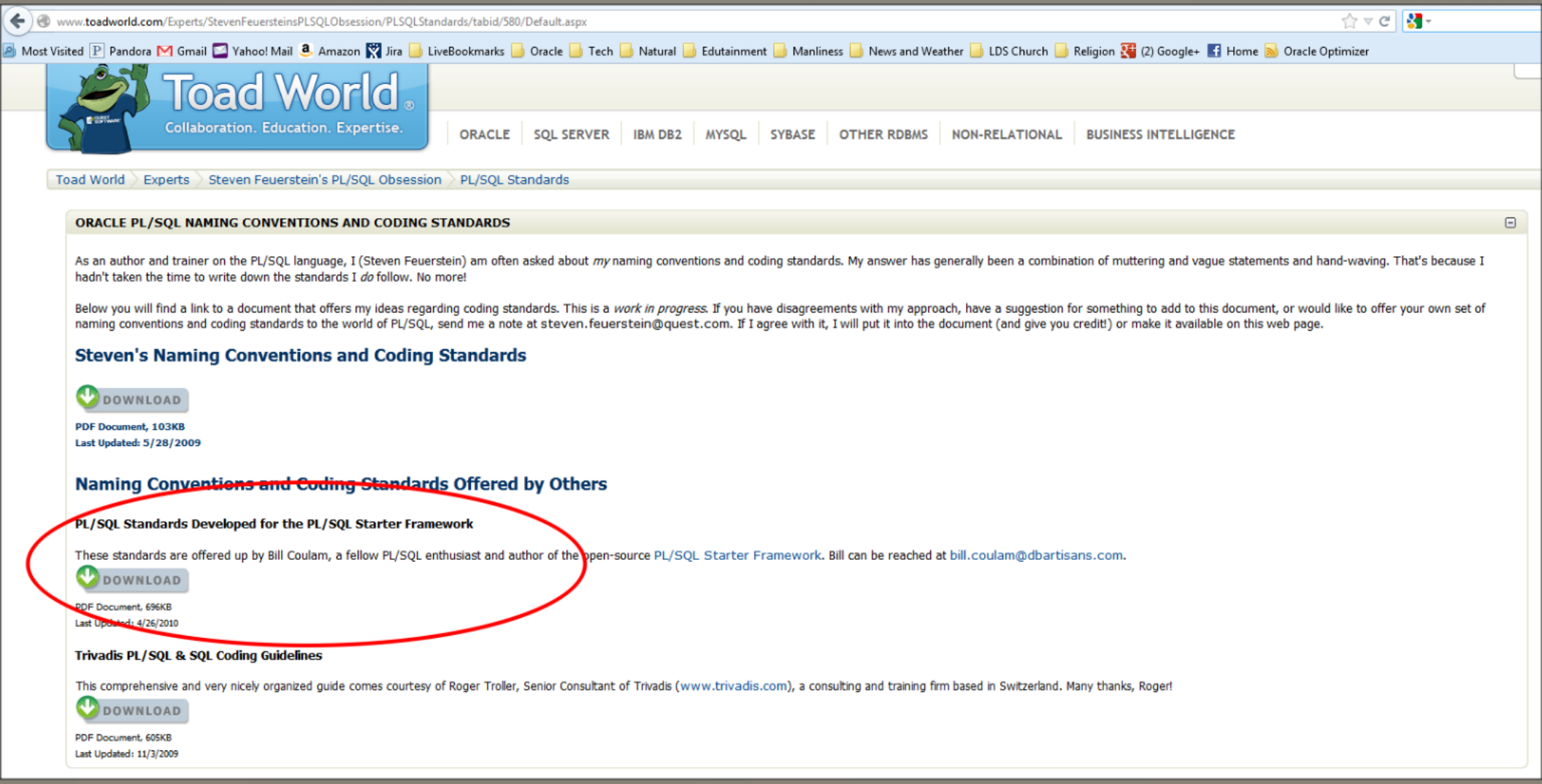
- PL/SQL Obsession Blog
- Real Automated Code Testing for Oracle Blog
- Steven's Personal Blog

Steven-related Products & Utilities

- Quest Code Tester for Oracle
- Quest Error Manager
- Quest CodeGen Utility
- Companion Files for Training (DEMO.ZIP)
  Updated December 12, 2011

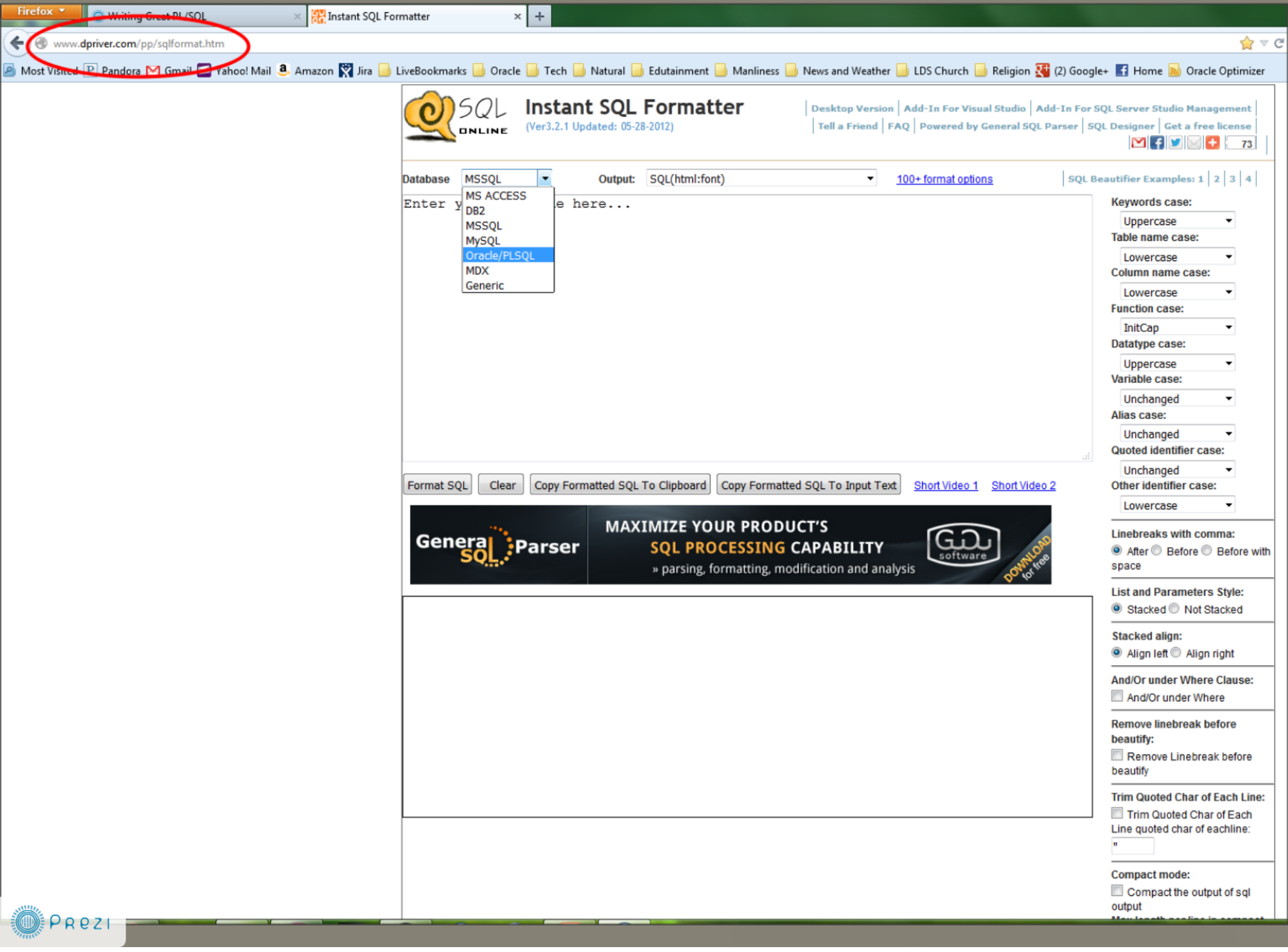
Websites

- StevenFeuerstein.com
- The PL/SQL Challenge
- The PL/SQL Channel
- I Love PL/SQL and ...


www.toadworld.com/Experts/StevenFeuersteinsPLSQLObsession/PLSQLStandards/tabid/580/Default.aspx
Most Visited Pandora Gmail Yahoo! Mail Amazon Jira LiveBookmarks Oracle Tech Natural Edutainment Manliness News and Weather LDS Church Religion (2) Google+ Home Oracle Optimizer
Toad World
Collaboration. Education. Expertise.
ORACLE SQL SERVER IBM DB2 MYSQL SYBASE OTHER RDBMS NON-RELATIONAL BUSINESS INTELLIGENCE
Toad World Experts Steven Feuerstein's PL/SQL Obsession PL/SQL Standards
ORACLE PL/SQL NAMING CONVENTIONS AND CODING STANDARDS
As an author and trainer on the PL/SQL language, I (Steven Feuerstein) am often asked about my naming conventions and coding standards. My answer has generally been a combination of muttering and vague statements and hand-waving. That's because I hadn't taken the time to write down the standards I do follow. No more!
Below you will find a link to a document that offers my ideas regarding coding standards. This is a work in progress. If you have disagreements with my approach, have a suggestion for something to add to this document, or would like to offer your own set of naming conventions and coding standards to the world of PL/SQL, send me a note at steven.feuerstein@quest.com. If I agree with it, I will put it into the document (and give you credit!) or make it available on this web page.
Steven's Naming Conventions and Coding Standards
DOWNLOAD
PDF Document, 103KB
Last Updated: 5/28/2009
Naming Conventions and Coding Standards Offered by Others
PL/SQL Standards Developed for the PL/SQL Starter Framework
These standards are offered up by Bill Coulam, a fellow PL/SQL enthusiast and author of the open-source PL/SQL Starter Framework. Bill can be reached at bill.coulam@dbartisans.com.
DOWNLOAD
PDF Document, 696KB
Last Updated: 4/26/2010
Trivadis PL/SQL & SQL Coding Guidelines
This comprehensive and very nicely organized guide comes courtesy of Roger Troller, Senior Consultant of Trivadis (www.trivadis.com), a consulting and training firm based in Switzerland. Many thanks, Roger!
DOWNLOAD
PDF Document, 605KB
Last Updated: 11/3/2009

- Establish a programming and style standard
- Ensure the standard is adhered to
- Use templates, automation and formatting tools to make it easy

- TOAD Formatter
- PL/SQL Developer Beautifier
- SQL Developer Formatter
- SQL Detective's Clear SQL
- Instant SQL Formatter
- etc. (one in most every tool now)


Firefox
Writing Great PL/SQL
Instant SQL Formatter
www.dpriver.com/pp/sqlformat.htm
Most Visited
Pandora
Gmail
Yahoo! Mail
Amazon
Jira
LiveBookmarks
Oracle
Tech
Natural
Edutainment
Manliness
News and Weather
LDS Church
Religion
(2) Google+
Home
Oracle Optimizer
SQL ONLINE
Instant SQL Formatter
(Ver3.2.1 Updated: 05-28-2012)
Desktop Version
Add-In For Visual Studio
Add-In For SQL Server Studio Management
Tell a Friend
FAQ
Powered by General SQL Parser
SQL Designer
Get a free license
73
Database
MSSQL
MS ACCESS
DB2
MSSQL
MySQL
Oracle/PLSQL
MDX
Generic
Output:
SQL(html:font)
100+ format options
SQL Beautifier Examples: 1 2 3 4
Enter y e here...
Keywords case:
Uppercase
Table name case:
Lowercase
Column name case:
Lowercase
Function case:
InitCap
Datatype case:
Uppercase
Variable case:
Unchanged
Alias case:
Unchanged
Quoted identifier case:
Unchanged
Other identifier case:
Lowercase
Format SQL
Clear
Copy Formatted SQL To Clipboard
Copy Formatted SQL To Input Text
Short Video 1
Short Video 2
General SQL Parser
MAXIMIZE YOUR PRODUCT'S
SQL PROCESSING CAPABILITY
» parsing, formatting, modification and analysis
GSP software
DOWNLOAD for free
Linebreaks with comma:
After Before Before with space
List and Parameters Style:
Stacked Not Stacked
Stacked align:
Align left Align right
And/Or under Where Clause:
And/Or under Where
Remove linebreak before beautify:
Remove Linebreak before beautify
Trim Quoted Char of Each Line:
Trim Quoted Char of Each Line quoted char of eachline:
"
Compact mode:
Compact the output of sql output
Prezi

```
CREATE OR REPLACE PACKAGE BODY str
/****************************************************************************
%author
Bill Coulam (bcoulam@dbartisans.com)

<pre>
Artisan      Date      Comments
============ ========= ============================================================
bcoulam      1997Dec30 Creation
snielsen     1998Mar05 Added str_to_bool
fdaws        2002Aug08 Added get_diacritic_list
ehansen      2004Mar15 Added get_diacritic_map
hbarnes      2006Oct30 Added foreign_to_ascii and nonascii_to_ascii
hbarnes      2008May18 Fixed foreign_to_ascii to not raise exception on null
                       parameter.
bcoulam      2012May22 Added TRIM to last element assignment in parse_list()

****************************************************************************/
AS

------------------------------------------------------------------------------
--                PACKAGE CONSTANTS, VARIABLES, TYPES, EXCEPTIONS
------------------------------------------------------------------------------


------------------------------------------------------------------------------
--                          PRIVATE FUNCTIONS AND PROCEDURES
------------------------------------------------------------------------------


------------------------------------------------------------------------------
                            PUBLIC FUNCTIONS AND PROCEDURES
------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
FUNCTION bool_to_str(i_bool_val IN BOOLEAN) RETURN VARCHAR2
IS
BEGIN
   IF (i_bool_val) THEN
      RETURN 'TRUE';
   ELSIF (i_bool_val IS NULL) THEN
      RETURN 'NULL';
   ELSE
      RETURN 'FALSE';
   END IF;
END bool_to_str;

--------------------------------------------------------------------------------
FUNCTION str_to_bool(i_str IN VARCHAR2) RETURN BOOLEAN
IS
BEGIN
   IF (LOWER(i_str) IN ('true','y','yes','t','1')) THEN
      RETURN TRUE;
   ELSIF (LOWER(i_str) IN ('false','n','no','f','0')) THEN
      RETURN FALSE;
   ELSIF (i_str IS NULL) THEN
      RETURN NULL;
   ELSE
      RAISE_APPLICATION_ERROR(-20000, 'str_to_bool does not support ['||i_str||']');
   END IF;
END str_to_bool;

--------------------------------------------------------------------------------
FUNCTION get_diacritic_list RETURN VARCHAR2
IS
```

```
BEGIN
   RETURN 'ÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏ';
END get_diacritic_list;

-------------------------------------------------------------------------------
FUNCTION get_diacritic_map RETURN VARCHAR2
IS
BEGIN
   RETURN 'DNOOOOOxOUUUUUYPBaaaaaaaceeeeiiiionooooo-ouuuuuypyAAAAAAACEEEEIIII';
END get_diacritic_map;

-------------------------------------------------------------------------------
FUNCTION foreign_to_ascii (
   i_str    IN VARCHAR2
)  RETURN    VARCHAR2
IS
BEGIN
   IF (i_str IS NULL) THEN
      RETURN NULL;
   END IF;

   RETURN TRANSLATE(i_str, get_diacritic_list, get_diacritic_map);
END foreign_to_ascii;

-------------------------------------------------------------------------------
FUNCTION nonascii_to_ascii (
   i_str    IN VARCHAR2
) RETURN VARCHAR2
IS
BEGIN
   RETURN ASCIISTR(i_str);
END nonascii_to_ascii;
```

```
--------------------------------------------------------------------------------
FUNCTION format_to_width
(
   i_str         IN VARCHAR2,
   i_width       IN INTEGER DEFAULT cnst.pagewidth,
   i_allow_wrap IN VARCHAR2 DEFAULT 'Y'
) RETURN VARCHAR2
IS
   l_ostr typ.t_maxvc2; -- o[riginal]str[ing]
   l_nstr typ.t_maxvc2; -- n[ew]str[ing]
   l_piece typ.t_maxvc2;
   l_length PLS_INTEGER := 0;
   l_curr_pos PLS_INTEGER := 1; -- current position
   l_width PLS_INTEGER := 0;
   l_next_para PLS_INTEGER := 0;
   l_next_space PLS_INTEGER := 0;
   l_next_dash PLS_INTEGER := 0;
   DASH CHAR(1) := CHR(45);
   PARA  CHAR(1) := CHR(126);
BEGIN
   -- Check odd conditions first where the string needs no processing
   IF (i_str IS NULL) THEN
      RETURN NULL;
   END IF;

   l_length := LENGTH(i_str);

   -- convert paragraphs so they can be turned back into paragraphs after
   -- LF and CR replacement.
   l_ostr := REPLACE(
               REPLACE(
                  REPLACE(
```

Anti-Pattern 2

The Dung Beetle Ball

Solution

Keep it Simple

- Routine no longer than a page+
- Routine does one thing and one thing well
- Don't repeat anything
  - Create constants and table-driven literals and parameters
  - Make vertical functions private to package body
  - Make public functions for common business logic
- Extract distinct SQL, business decision points and functions
  - Place in their own atomic routine

- Layer upon layer of functionality in a single proc
- No opportunity for reuse
- Lots of points of failure
- Raised exceptions are onerous
- Code is too long; difficult to grasp and modify
- The complexity attracts bugs
- The fixes pile up and accrete until the routine resembles a jungle, or a steaming heap
- Best pulled apart and re-written the right way

Solution

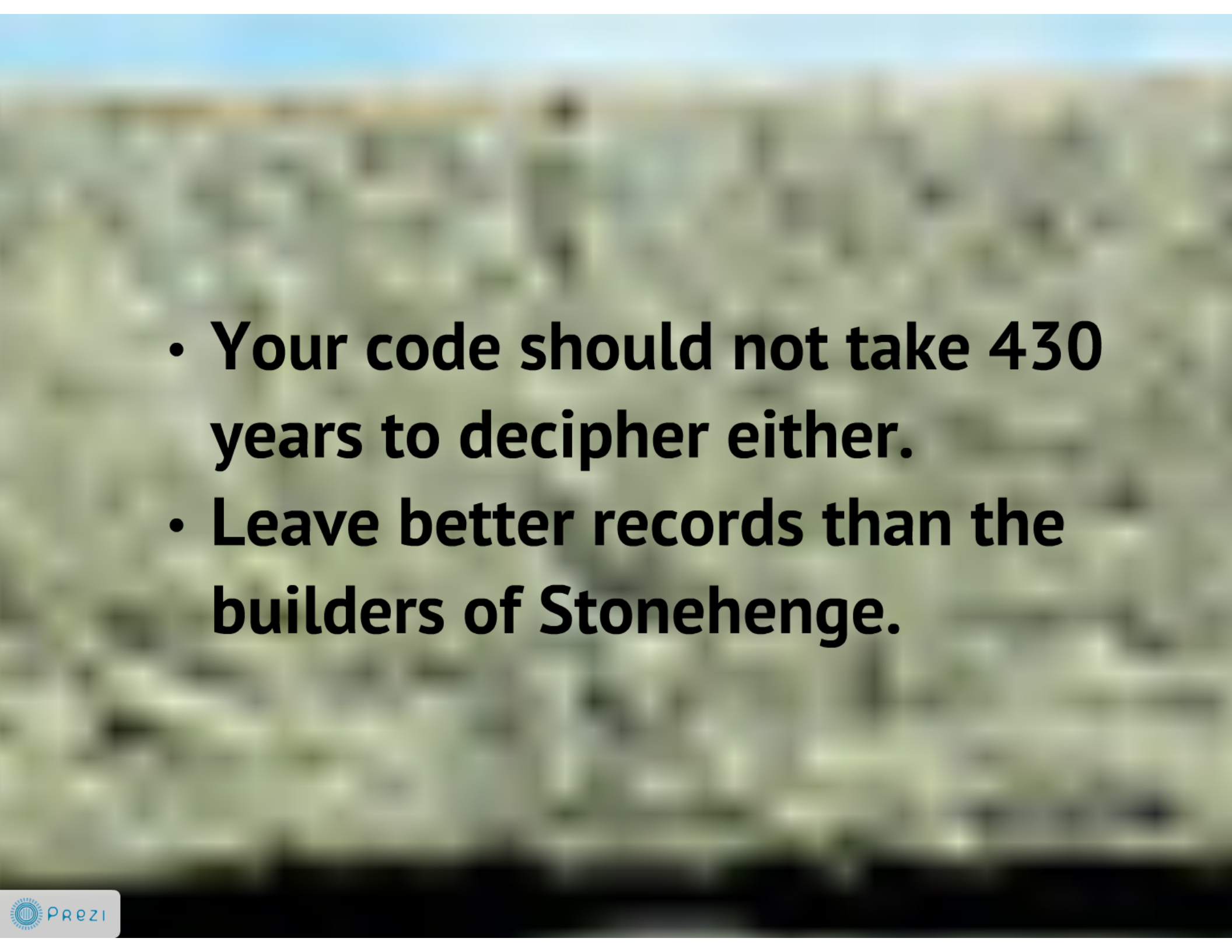
Keep it Simple

- Routine no longer than a page+
- Routine does one thing and one thing well
- Don't repeat anything
    - Create constants and table-driven literals and parameters
    - Make vertical functions private to package body
    - Make public functions for common business logic
- Extract distinct SQL, business decision points and functions
    - Place in their own atomic routine

Anti-Pattern 3

Stonehenge

- Studied scientifically for over 430 yrs
- There have been many theories put forth to explain its existence and usage.
- We're still not 100% sure everything its 4 phases of construction were used for.


• Your code should not take 430 years to decipher either.
• Leave better records than the builders of Stonehenge.

Solution

Document the Interface

- Encourage through templates
- Each package spec, routine, trigger, view and job should have a block explaining who, when and why.
- Document assumptions and test them in the code
- Document tricky parameters or usage, caveats, design notes, alternatives rejected and why

Anti-Pattern 4

Rabbits and Tribbles

- Starts innocently. Developer sees no harm in writing a few standalone routines. Then two of his friends follow suit, and they tell their two friends...
- Within a few short years, the codebase is a tangled, unnavigable mess of standalone routines with no owners.
- Legitimacy is a real concern, but they're all kept around "just in case."
- No organization to the enterprise code base
- Very little explanation as to who, when, why created
- No idea of relationship to other routines
- Blind to reuse of shared data, SQL and functions

Solution

Everything Needs a Home

- Everything goes in a PL/SQL package
- If package becomes a dumping ground, the name was too generic.
    - Break up package into functional groups
- Do not put all literals in one package; keep them grouped in the package spec to which they are more closely aligned.
- Trigger and job code should also go in a package

Anti-Pattern 5

Mona Lisa at the Mall

- Keeping the Mona Lisa in a location publicly accessible is unconscionable; she's been through enough already. Could get defaced like this.
- The original belongs in a special, environmentally controlled and secure case.
- If the original were damaged like this, there's no going back.
- Database code objects are precious too. Keeping them or editing them in the database is an idea equally as bad as displaying da Vinci's masterpiece in the hall at the local mall.

- Keeping the Mona Lisa in a location publicly accessible is unconscionable; she's been through enough already. Could get defaced like this.
- The original belongs in a special, environmentally controlled and secure case.
- If the original were damaged like this, there's no going back.
- Database code objects are precious too. Keeping them or editing them in the database is an idea equally as bad as displaying da Vinci's masterpiece in the hall at the local mall.

Solution

Version the Source Code

- Install and configure a reliable source code control system (Subversion, Git, RCS, CVS, PVCS, VSS, etc.)
- Ensure it is used religiously.
  - Development and maintenance activities always begin as or with the source file, not the database object.
  - Some tools now default database object browsing to read-only mode. Have to force them in order to edit a DB object in-situ.

Anti-Pattern 6

Flying Blind

Which visual acuity would you prefer for your pilot ?

OR ?

Flying Blind

Which visual acuity would you prefer for your pilot ?

OR

?

Yet this is exactly how 90+% of DBAs deploy code: with zero insight into the innards of production code.

- No logging or monitoring
- No auditing
- No metrics
- No history
- No notifications
- No ability to debug on the fly

- When something goes wrong, it becomes a slow and costly, error-prone, misguided effort in heroics.
- It SHOULD be as simple as opening logs and change history, and spending a couple minutes reading what happened, who did it, when it happened, and what data (if any) was involved.
- If debugging is required, should be easy. Recompiling production PL/SQL to spit out DBMS_OUTPUT lines should be a badge of shame.

Solution

Resource Name	License	Purpose	Location & Notes
Google Code	Free	Library of libraries	http://code.google.com/hosting/search?q=label:plsql
Feuerstein PL/SQL Obsession	Free	Repository of all things SF and PL/SQL	http://www.toadworld.com/sf
QCGU (Quest CodeGen Utility)	Free	Full framework, Standards, Scripts, Template Factory, Code Generation, etc.	http://codegen.inside.quest.com/index.jspa Latest incarnation of Feuerstein's vast reservoir of experience. (successor of QXNO, PL/Vision, and PL/Generator.)
PL/SQL Starter	Free	Full framework. Standards.	http://sourceforge.net/projects/plsqlframestart
Simple Starter	Free	Logging, Timing, Auditing, Debugging, Error Handling, etc.	Simplified PL/SQL Starter to just logging, timing and auditing components (and the low-level packages they depend on). Designed to be used in one schema. Install and begin using in under a minute.
GED Toolkit	$120-$1200	Almost full framework	http://gedtoolkit.com Includes APEX UI to administer jobs and tables. Monitor processing.
PL/Vision	Free	Framework, API Generator, + more	http://toadworld.com/Downloads/PLVisionFreeware/tabid/687/Default.aspx Replaced by QXNO and then QCGU. Not supported.
Log4ora	Free	Logging	http://code.google.com/p/log4ora/ Fresh, full-featured logging library. Alerts. AQ. Easy to use. Good stuff.
ILO	Free	Timing and Tuning	http://sourceforge.net/projects/ilo From the sharp minds at Hotsos
Quest Error Manager	Free	Error Handling	http://www.toadworld.com/LinkClick.aspx?link=685&tabid=153 Included in QCGU. But offered separately as well. Not supported.
Plsql-commons	Free	Collection of utilities, including logging	http://code.google.com/p/plsql-commons
Log4oracle-plsql	Free	Logging	http://code.google.com/p/log4oracle-plsql Seems like an active project, but could not find code to download…
Log4PLSQL	Free	Logging	http://sourceforge.net/projects/log4plsql Popular, but aging and complex log4j analog in PL/SQL
Logger	Free	Logging	http://sn.im/logger1.4 Recently orphaned when Oracle decommissioned its samplecode site. Simple. Easy to use.
Orate	Free	Logging	http://sourceforge.net/projects/orate Never used it, but has been around a while. Still active.

Adopt an existing instrumentation library.

Automatically gain:

- **Dynamic debugging (invisible and little overhead to production; but can be activated by changing a column value)**
- **Change auditing**
- **Proactive monitoring and email notifications of anomalous events**
- **Recording of metrics**
- **"Tag" sessions and long operations**
- **Debug strings inherently comment the code as well!**

Practices of Great PL/SQL Programmers

- Get involved early
- Model right, then make it friendly
- Know and use your tools
- Only handle expected exceptions
- Peer review
- Do it in Bulk or a Single SQL
- Caller in charge of transaction
- There's this thing called "testing"...

Get Involved

- Get involved in early stages of project work
- Ask the questions of a data architect: length, uniqueness, longevity, audit, security, relationships, cardinality, etc.
- Prepare conceptual and logical models for review and project discussions
- Be aware of overall application architecture, hardware and human requirements

Model Right

- The data model is the foundation of the structure. Ensure that at least the data model is done right.
- Ignore those pleading for more friendly columns, quicker data access, elimination of joins.
- AFTER the model is correct, normalized, and reviewed, then make it friendly with views, materialized views, updateable views, virtual columns, etc.

Know Your Tools

- Take 10 minutes a day to explore the user guide of a favorite tool
- Learn and configure keyboard shortcuts
- DBAs can really benefit from model visualization, nameless macros, data generation, debuggers, data ETL, code/table/schema/model/database comparison tools, and DDL generators.

Exceptions

- Ban the use of WHEN OTHERS
  - Except when hiding the error is intentional
- Only write exception handling for expected exceptions.
  - Use a standard way of logging and re-raising
- Allow PL/SQL's default exception raising and transaction rollback to handle everything else

Another Head, Another Set of Eyes

- Pair programming is fantastic
- If not formally, quickly and informally get peer review for:
  - simple DML scripts
  - models
  - source code
  - design approach and assumptions
  - DB build manifest
  - Whenever you're stuck for more than 20-30 minutes

Do it in Bulk

- Quickest way to do something is not do it at all (ask yourself “Do I really need this?”)
- Next best is to do it in a single SQL statement
- Followed by doing it with Bulk PL/SQL features (collections, bulk bind, FORALL, and DML with collections of record type)

Transactions

- Let the client control the transaction
- If you are writing a backend job that begins life in the database, then you will write the driver and control COMMIT vs. ROLLBACK.
- Otherwise, leave it to database client software to decide.

Testing

- Document requirements and assumptions in the interface
- Write tests first, to the interface
- Then write body, re-testing all cases as you add/modify the code to reach requirements.
- Left with nice suite of re-usable tests
- Test with typical load and data quality


Get Involved
• Get involved in early stages of project work
• Ask the questions of a data architect: length, uniqueness, longevity, audit, security, relationships, cardinality, etc.
• Prepare conceptual and logical models for review and project discussions
• Be aware of overall application architecture, hardware and human requirements

Questions?

Contact me:

bcoulam@yahoo.com