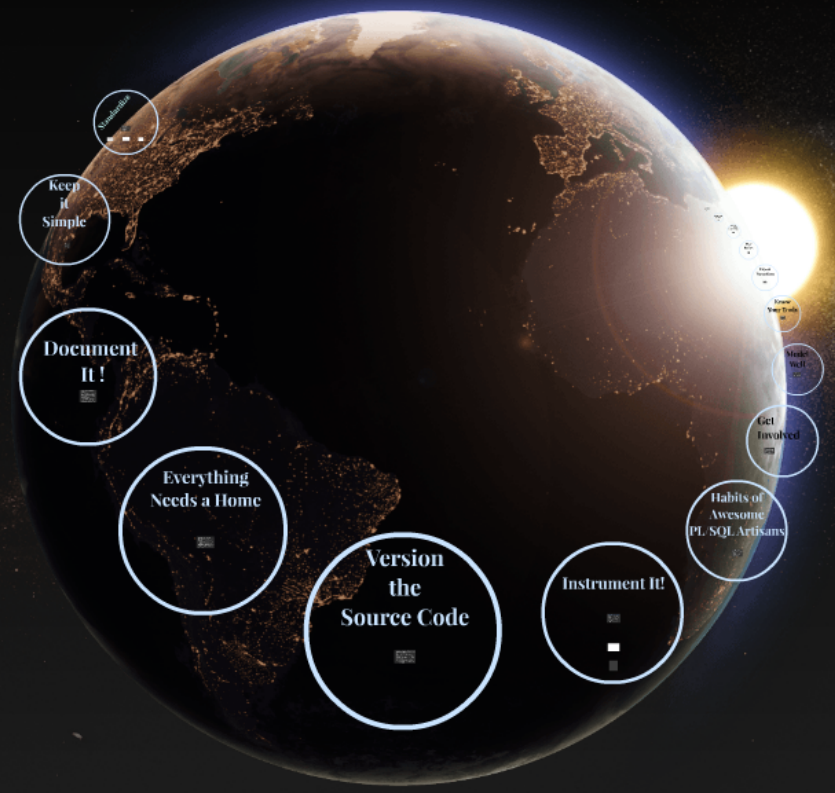
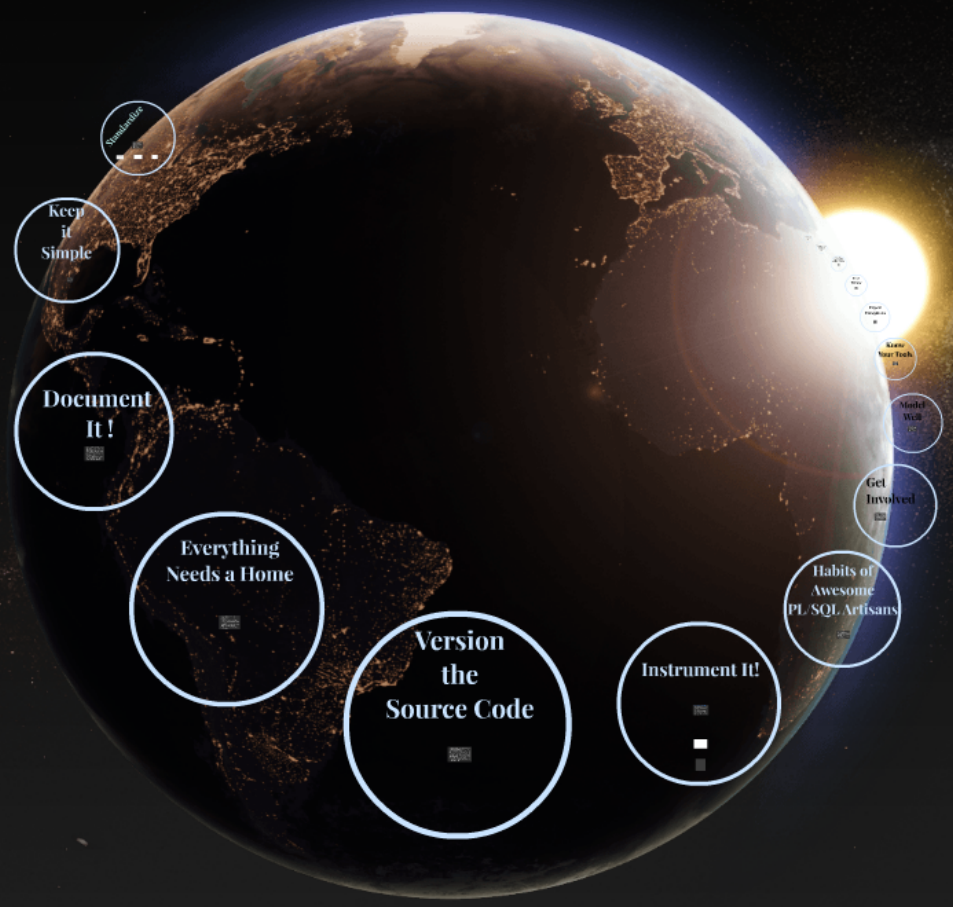
How to Write Awesome PL/SQL !



How to Write Awesome PL/SQL !



The Presenter

- Bill Coulam
- bcoulam@yahoo.com
- Programming C, Java and PL/SQL since 1995
- Andersen Consulting (PacBell, US West, AT&T) - San Francisco, Denver, Herndon VA
- New Global Telecom - Denver, CO
- The Structure Group - Houston, TX
- Church of Jesus Christ of Latter Day Saints - SLC, UT
- Speaking at RMOUG, IOUG and UTOUG since 2001
- Passionate about excellent modeling and programming practices

Involve users early and often

There's always another bug

Reduce and re-use.
Don't repeat.

Simplify.

Test with plenty of dirty data

Get another pair of eyes

Document It !

Write as if you'll contract amnesia next week

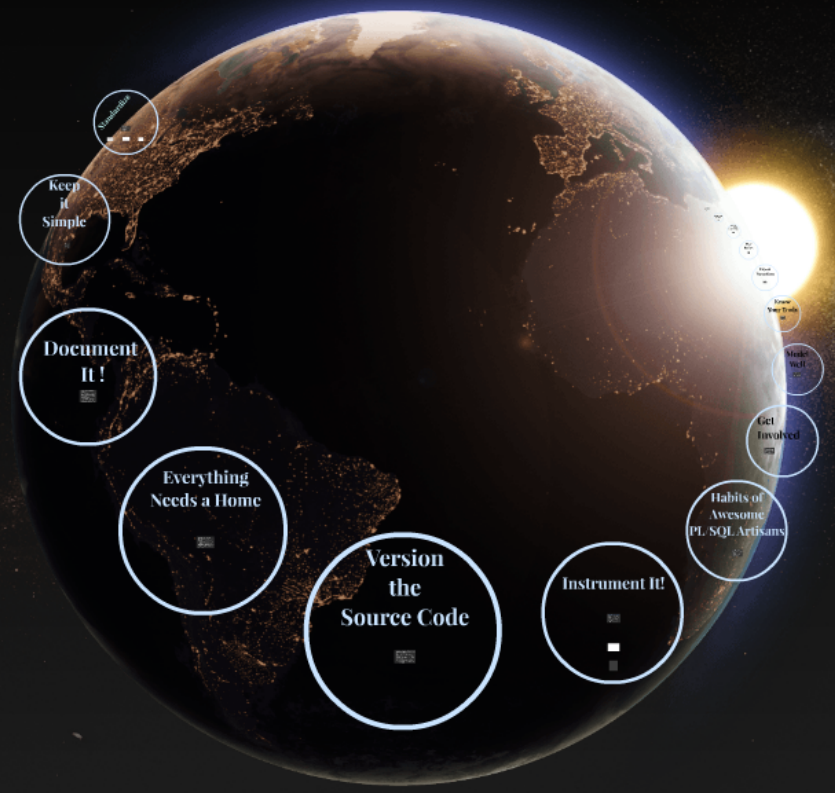
"Write as if your code will be maintained by a homicidal maniac who knows where you live."

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

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

How to Write Awesome PL/SQL !



Standardize

- Establish a programming style and standard
- Ensure the standard is followed
- Use templates
- Use automation
- Use a formatter

Standardize

- Establish a programming style and standard
- Ensure the standard is followed
- Use templates
- Use automation
- Use a formatter

- Establish a programming style and standard
- Ensure the standard is followed
- Use templates
- Use automation
- Use a formatter

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 th 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....y *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 boo "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

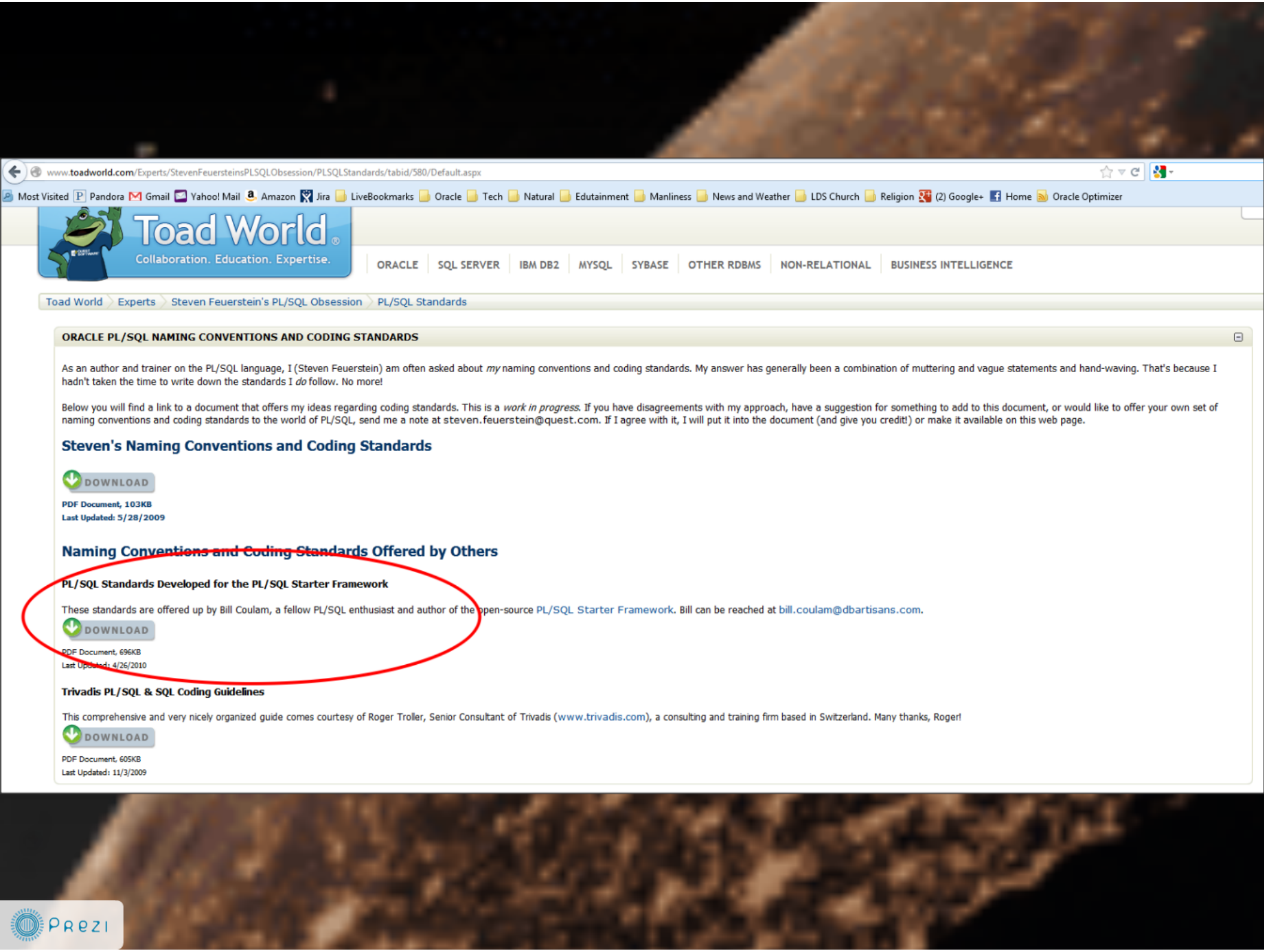
Steven-related Products & Utilities

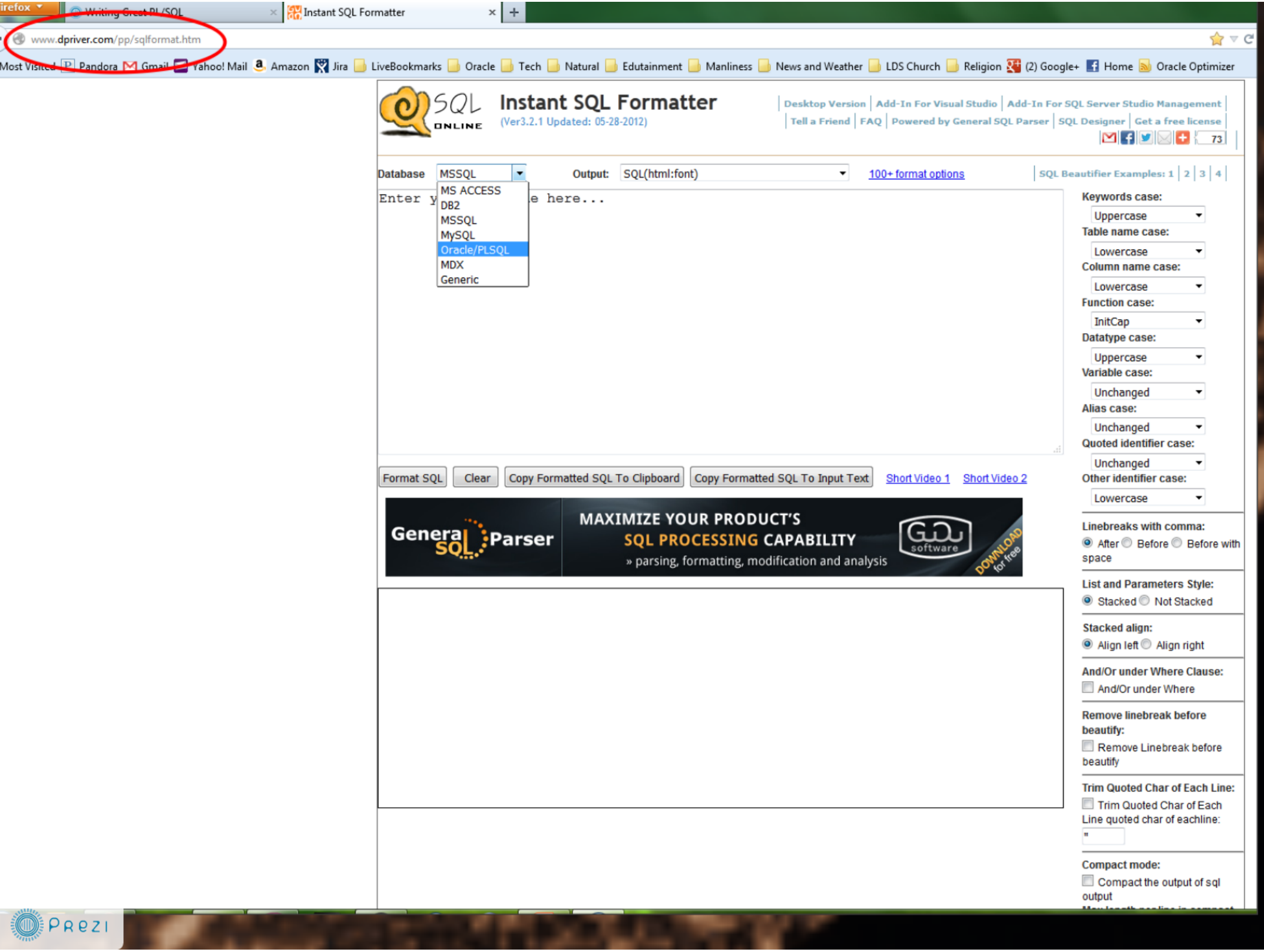
- Quest Code Tester for Oracle
- Quest Error Manager
- Quest CodeGen Utility
- Companion Files for Training (DEMO.ZIP)

Updated December 12, 2011

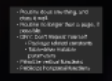
Websites

- 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


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

Keep it Simple



- Routine does one thing, and does it well.
- Routine no longer than a page, if possible.
- DRY: Don't Repeat Yourself
  - Package related constants
  - Table-drive mutable parameters
- Privatize vertical functions
- Publicize horizontal functions

Document It !

- Placeholders for explanation in templates
- Each stored object should have a comment 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

- Placeholders for explanation in templates
- Each stored object should have a comment 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

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
- Group literals in the package spec to which they are best related.
- Trigger and job code should also go in a package

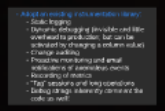
- Everything goes in a PL/SQL package
- If package becomes a dumping ground, the name was too generic.
    - Break up package into functional groups
- Group literals in the package spec to which they are best related.
- Trigger and job code should also go in a package

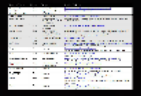
Version the Source Code

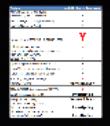
- Install and configure a reliable source code control system (Subversion, Git, RCS, CVS, PVCS, VSS, etc.)
- Ensure it is used consistently (100% of the time)
- Development and maintenance activities always begin with the source file, not the database object
- Use default read-only mode of DB Tools
  - Should be hard for developer to edit a DB object in-situ

- Install and configure a reliable source code control system (Subversion, Git, RCS, CVS, PVCS, VSS, etc.)
- Ensure it is used consistently (100% of the time)
- Development and maintenance activities always begin with the source file, not the database object
- Use default read-only mode of DB Tools
  - Should be hard for developer to edit a DB object in-situ.

Instrument It!





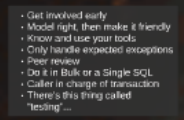


- Adopt an existing instrumentation library:
  - Static logging
  - Dynamic debugging (invisible and little overhead to production; but can be activated by changing a column value)
  - Change auditing
  - Proactive monitoring and email notifications of anomalous events
  - Recording of metrics
  - "Tag"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

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.

Feature	Useful in Starter Framework
More PL/SQL-Only Data Types Cross PL/SQL-to-SQL Interface	N
ACCESSIBLE_BY Clause (Whitelisting)	N
Implicit Statement Results	N
Invoker Views (BEQUEATH CURRENT_USER)	Y
Roles for Program Units	N
New Conditional Compilation Directives ($$PLSQL_UNIT_OWNER, $$PLSQL_UNIT_TYPE)	N
Optimizing Function Execution in SQL	N
Using %ROWTYPE with Invisible Columns	N
FETCH FIRST Clause and BULK COLLECT	N
The UTL_CALL_STACK Package	Y
Identity columns	Very cool, but N
DEFAULT [ON NULL] sequence.NEXTVAL	Y
VARCHAR2 expansion to 32K	Investigating
Row pattern matching.	Very cool, but N
Combine invoker rights and function result cache	Investigating
Global application contexts w/o RAC limitations (added in 11g?)	Investigating

Habits of Awesome PL/SQL Artisans



- 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

- 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 Well

- The data model is the foundation of the system.
- Ensure the model is done properly the first time.
- 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.

- The data model is the foundation of the system.
- Ensure the model is done properly the first time
- 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.

- 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.

Expert 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

- 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

Peer Review

- Pair programming is fantastic
- Formal or informal second set of eyes for:
  - simple DML scripts
  - models
  - source code
  - design approach and assumptions
  - DB build manifest
  - whenever you're stuck for more than 20-30 minutes

- Pair programming is fantastic
- Formal or informal second set of eyes 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 Big
Do it in Bulk
Bulk is da Bomb

- Quickest way to do something is not do it at all. If performance is bad, ask...
  - "Do I really need this?"
  - "Can it be done another way?"
- 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)

- Quickest way to do something is not do it at all. If performance is bad, ask...
  - “Do I really need this?”
  - "Can it be done another way?"
- 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)

Transaction Control

- 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.

- 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

- 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

bcoulam@yahoo.com
www.dbartisans.com
www.sourceforge.net/projects/plsqlframestart/

How to Write Awesome PL/SQL !