Using Thick Database Principles to Leverage Oracle SQL and PL/SQL Part I:

Save Cloud Costs and Simplify User Interface Development

Peter Koletzke
Technical Director & Principal Instructor

ORACLE ACE Director

ORACLE Developer Community

NoCOUG

Me

- 34 yrs. database industry
- 30 yrs. consulting in Oracle arena
  - Since Oracle 5.1C, SQL*Forms 2.3
- 38 yrs. as trainer/presenter
- User groups
  - 350+ presentations, 12 awards
  - 7+ yrs. total on boards of directors
    - IOUG(-A), NYOUG, UTOUG
- Oracle ACE Director
  - Since program inception in Aug. 2005
- Oracle Certified Master
  - Since program inception in Dec. 2001
- 8 Oracle Press books coauthored
  - 6262 pages total

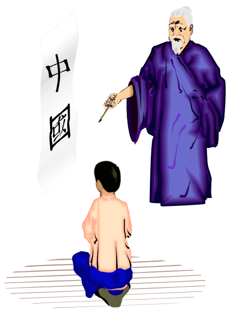
Oracle Designer/2000 Handbook
Oracle Designer Handbook, Second Edition
Oracle Developer Advanced Forms & Reports
Oracle JDeveloper 3 Handbook
ORACLE9i JDeveloper Handbook
ORACLE JDeveloper 10g Handbook
ORACLE JDEVELOPER for Forms & PL/SQL Developers: A Guide to J2EE Development
Oracle JDeveloper 11g Handbook

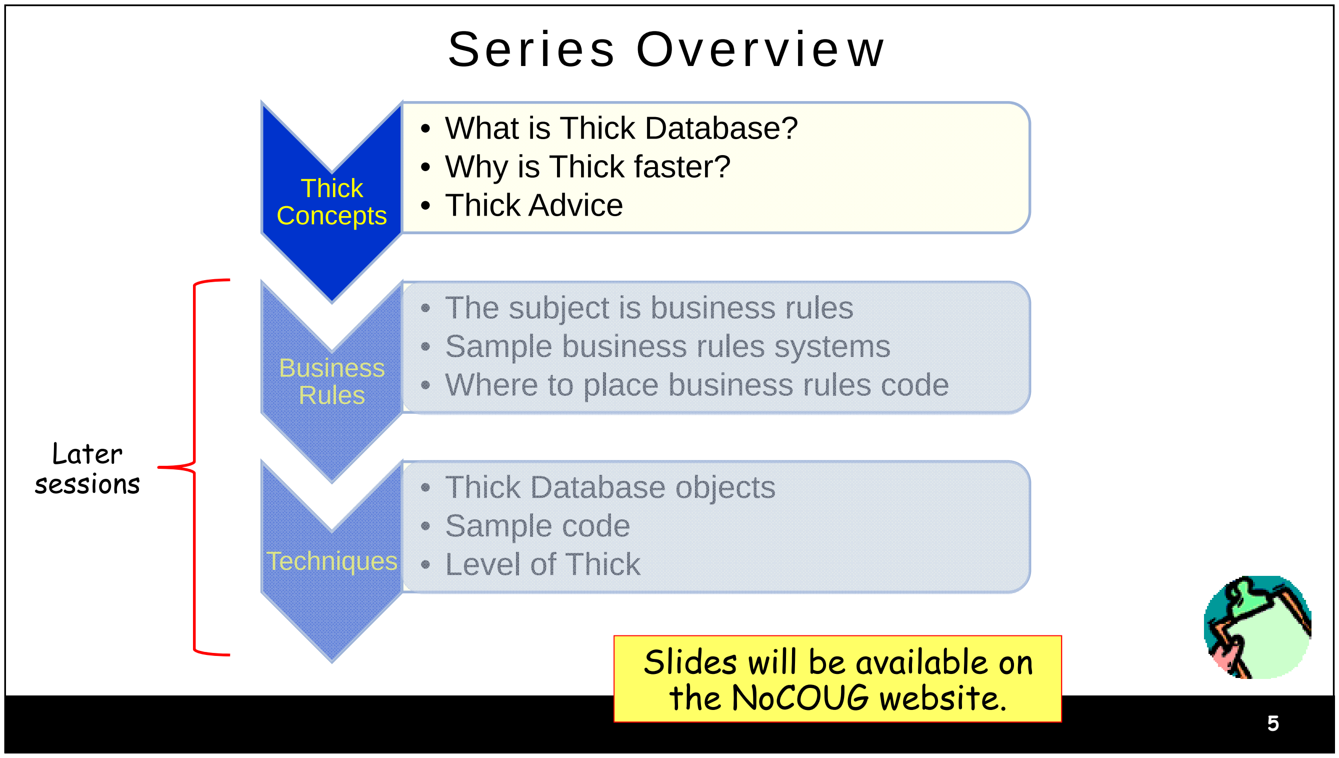
You

- Job responsibilities?
  - DBA, developer
- Development tools?
  - Oracle Developer Forms/Reports
  - APEX
  - ADF, MAF
  - MAX, VBCS
  - Other JavaScript tools
  - .NET
  - PL/SQL
  - Other

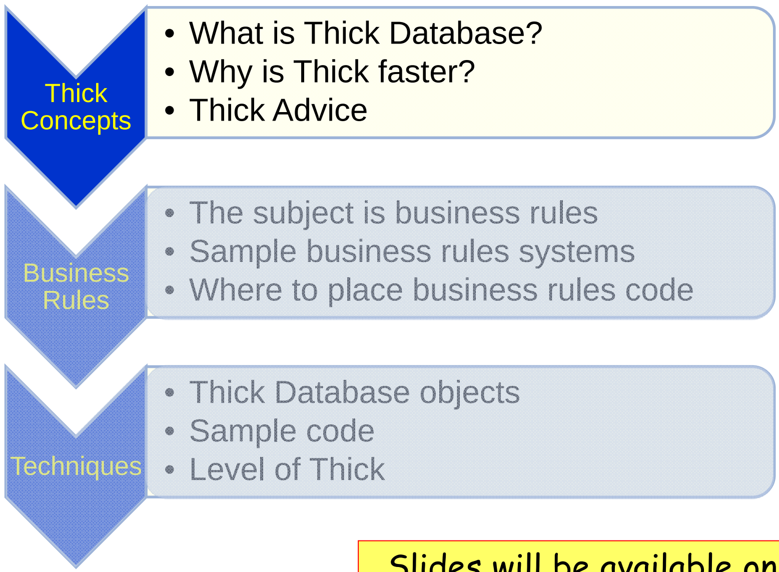
Thick Thoughts

- In the Days of Olde, you had to program in some non-DB language to issue SQL to the database
- The introduction of PL/SQL changed that
  - But in the early days, PL/SQL was more primitive so processing in the UI language was still indicated
- Now PL/SQL and the database have become more efficient, feature-full
  - But Days of Olde thought patterns are still in place
  - This leads to UI-heavy coding and less efficient processing





Series Overview



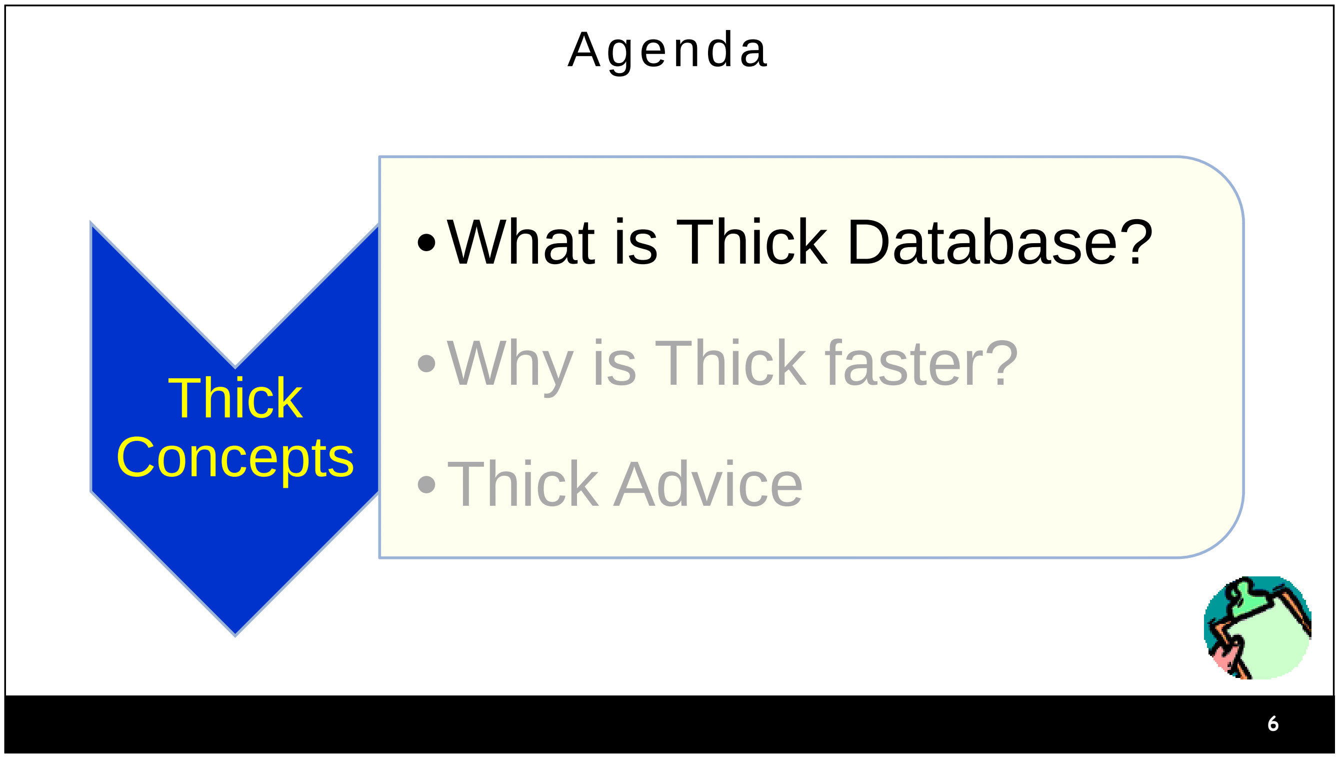
- What is Thick Database?
- Why is Thick faster?
- Thick Advice

Later sessions

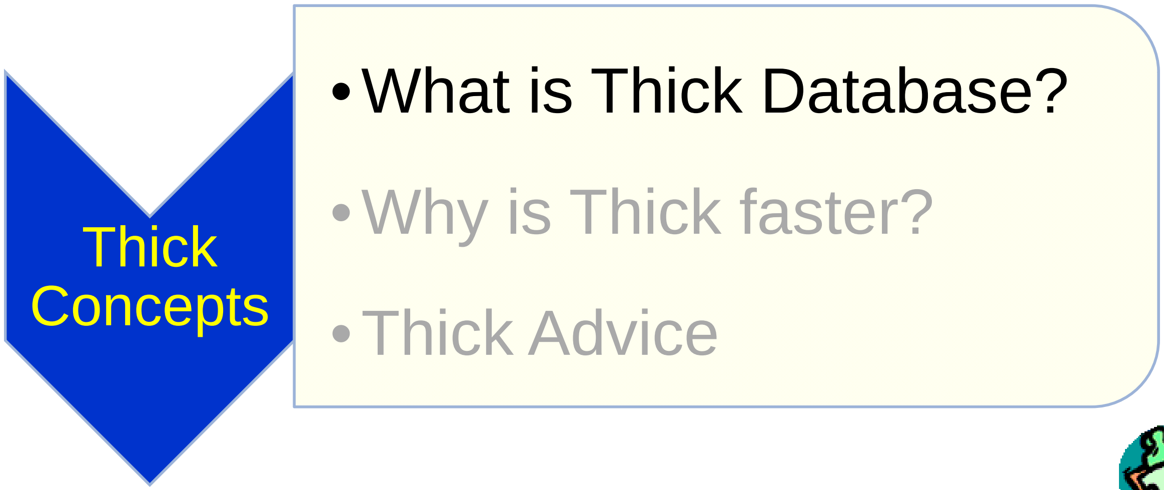
- The subject is business rules
- Sample business rules systems
- Where to place business rules code

- Thick Database objects
- Sample code
- Level of Thick

Slides will be available on the NoCOUG website.



Agenda



- What is Thick Database?
- Why is Thick faster?
- Thick Advice

The Other Extreme

I'm as thick as a plank.

—Princess Diana (1961-1997)

About Thick Database

- A code development strategy
  - Maximize use of database code to simplify the user interface
  - The user's device (client) runs minimal code
- Name plays off the term "thin client"
  - A "Year of the Internet" term
  - Means most processing occurs on a server
  - Slightly outmoded now
- Thick database means "thin client"

A.k.a.
- Thick Database Approach
- Thick Database Paradigm
- Smart Database
- SmartDB
- Fat Database

Oracle prefers

Provenance

- Topic is rarely seen at conferences, but is not new
- Started trending many years ago
  - ODTUG Business Rules Symposium Day 2001-2004
    - Organized by Dr. Paul Dorsey of Dulcian, Inc.
  - Thoughts evolved into Thick Database
    - Conference sessions starting around 2006

Topic is Still Active

- dulcian.com
  - Look in Resources | Conference Presentations…Thick Database
- Mike Smithers' Blog
  - https://mikesmithers.wordpress.com/tag/thick-database-paradigm/
- Toon Koppelaars, Oracle Real World Performance Group
  - https://www.youtube.com/watch?v=8jiJDflpw4Y
  - http://www.prohuddle.com/webinars/ToonKoppelaars/ThickDB.php
- Bryn Llewellyn, Distinguished Product Manager (Oracle)
  - https://blogs.oracle.com/plsql-and-ebr/entry/why_use_pl_sql
  - https://blogs.oracle.com/plsql-and-ebr/noplsql-versus-thickdb

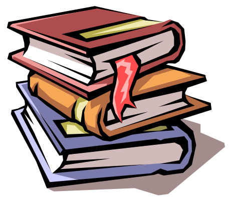
Relatively Recent Dulcian Presentations

- **A New View of Database Views**
  - https://www.slideshare.net/MishaRosenblum/2015-458-rosenblumpptfinal?next_slideshow=1
- **Why is the application running so slowly?**
  - https://www.slideshare.net/MishaRosenblum/why-is-the-application-running-so-slowly

DULCIAN INC.
SOFTWARE SOLUTIONS THAT WORK

More Resources

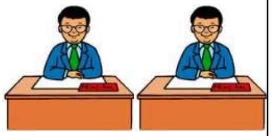
- Anton Nielsen, concept2completion.com, APEX and the Thick Database Paradigm
- Toon's ODTUG 2017 slides
  - http://thehelsinkideclaration.blogspot.com/2017/06/my-noplsql-versus-smartdb-deep-dive.html
- Database debunking
  - http://www.dbdebunk.com/



Two Related Strategies

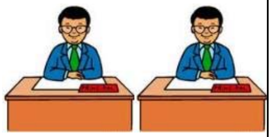
1. System for tracking business rules
   - Store definitions in database tables
   - Code implementations linked to rules
   - Linked to, or serves as, requirements documentation
2. Database features implement the business rules
   - Constraints
   - Database API: PL/SQL code, updatable views

♪ Oracle is considering support for SQL Assertions:
https://community.oracle.com/ideas/13028



Guiding Principles for Code

- Database code that implements business rules in PL/SQL
  - Using database features to enforce data integrity defined by business rules seems obvious
- Database views to represent complex business objects (SQL)
  - Each view has an accompanying application programming interface (API)
    - Written in PL/SQL
  - Interaction is with view and API



Drawbacks

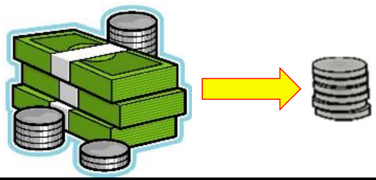
- Time and effort required
  - Design and set up
  - Documenting standards
  - Instructing staff
- Requirements on the IT shop side
  - Architect/database designer
  - Expert coder
    - Develop generic code “engines” to run and/or generate business rules code
- Need buy-in from management
  - For all of the above

Some Benefits

- Application accuracy
  - Business rules match application code
  - Test plans can be generated from business rules system
- Productivity
  - Can greatly simplify user interface code
- Code reusability
  - Ease of application maintenance
- Faster performance
  - Code is close to data storage – fewer messages, easy access
  - Views also reduce the number of round trips needed
- Proper use of staff
  - User interface developers can concentrate on UI code
  - Database code developers can concentrate on database code to support the UI

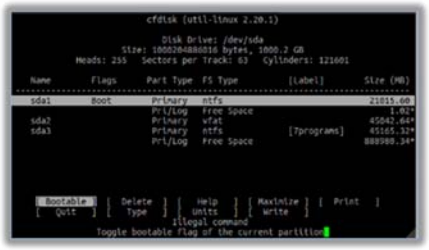
Big Benefit: Lower Cloud Costs

- Save cloud database processing time
  - Application's use of database is more efficient
  - Proof in next section
- Less database processing time == lower cloud costs
- If application front-end is in cloud
  - Thick DB saves application runtime processing
  - Examples
    - Java Cloud Service
    - Container Cloud Service



Big Benefit: Simplifies User Interface Work

- Database views can represent multiple tables
  - Arbitrarily complex logic
  - Aggregate functions: MAX(), COUNT()
  - Set operators: UNION, MINUS
  - Calculation functions: first_salary()
  - Even: a PL/SQL function cast as a table
- One view per application UI page
  - The page submit commits the entire page
  - Reminds one of mainframe "block submit"
  - Back end code deals the data into
    the proper tables

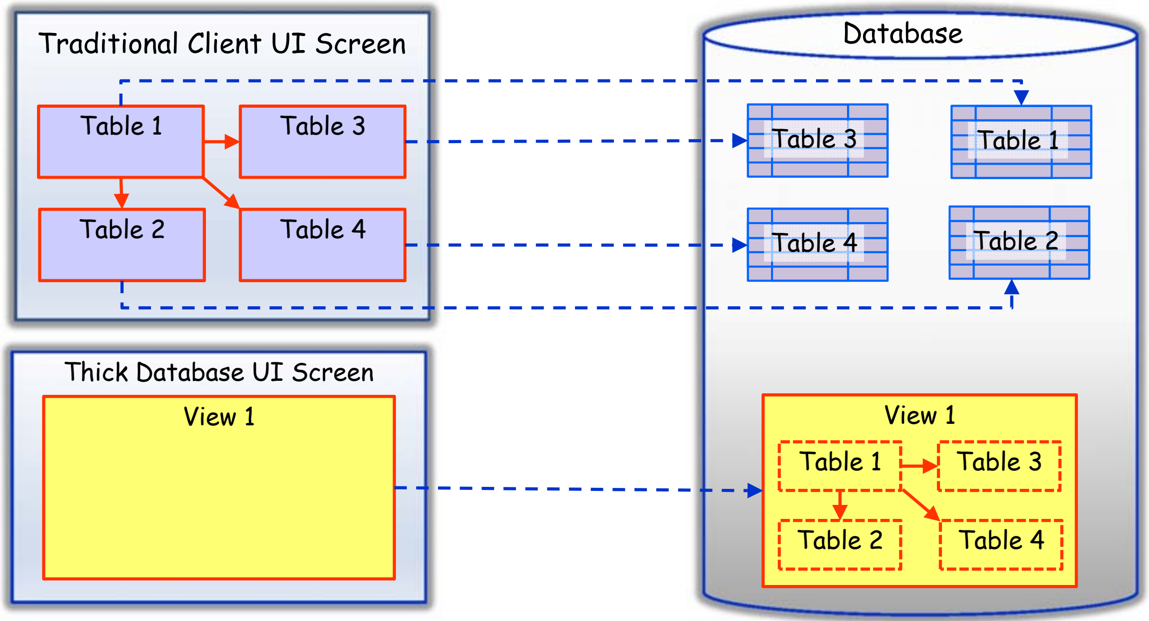


```
                      cfdisk (util-linux 2.20.1)

                        Disk Drive: /dev/sda
                 Size: 1000204886016 bytes, 1000.2 GB
        Heads: 255   Sectors per Track: 63   Cylinders: 121601

  Name        Flags       Part Type  FS Type          [Label]        Size (MB)
 -----------------------------------------------------------------------------
  sda1        Boot         Primary   ntfs                             21815.60
                           Pri/Log   Free Space                           1.02*
  sda2                     Primary   vfat                             45042.64*
  sda3                     Primary   ntfs             [7programs]     45165.32*
                           Pri/Log   Free Space                      888980.34*


   [ Bootable ]  [  Delete  ]  [   Help   ]  [ Maximize ]  [  Print   ]
   [   Quit   ]  [   Type   ]  [  Units   ]  [  Write   ]
                          Illegal command
             Toggle bootable flag of the current partition
```

Some Changes Require Less UI Rewriting

- UI technology changes
  - If code is in database, only UI needs rewriting
  - Application logic in database can carry forward
- Table refactoring
  - For example, if a set of tables used in UI views is normalized into more tables
    - Joins and query of view can be updated
    - UI may not need to change

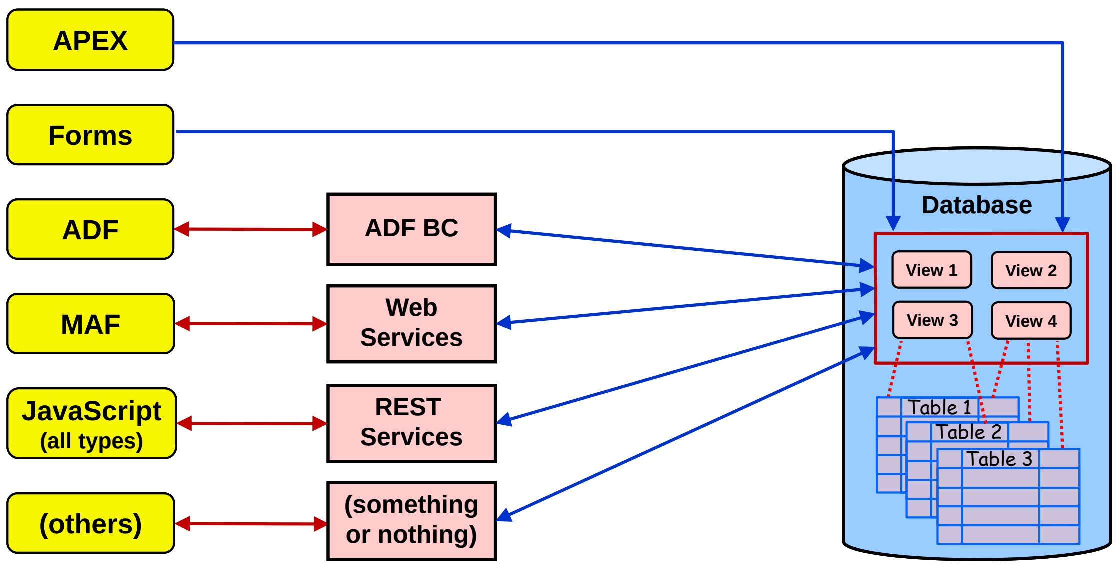
Traditional vs. Thick Database UIs



Front-end Tool "Agnostic"

- Application Express (APEX)
- Application Development Framework (ADF)
- Mobile Application Framework (MAF)
- Oracle Forms
- JavaScript
  - JavaScript Extension Toolkit (JET)
  - Mobile Application Accelerator (MAX)
  - Visual Builder Cloud Service (VBCS, formerly ABCS)
- PL/SQL Toolkit
- PHP: Hypertext Processor (PHP)
- Rails
- ColdFusion
- (whatever)

Tools' Use of Thick Database



Agenda

Thick Concepts

- What is Thick Database?
- Why is Thick faster?
- Thick advice

Toon Koppelaars' Research

- Described in YouTube video
  - "NoPLSql and Thick Database Approaches with Toon Koppelaars"
  - https://www.youtube.com/watch?v=8jiJDflpw4Y
- Related conference presentations
- Now uses the Oracle-preferred term "SmartDB"

(Slides with this icon contain content used with Toon's permission.)

What SmartDB is NOT

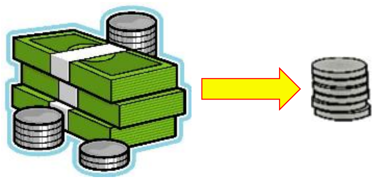
- NoPlSql
  - Starts with OO model in the middle tier
  - DB is just a bunch of tables
  - “PL/SQL is proprietary - don’t use it.”
  - “Database is for data, not logic because logic won’t scale.”
  - Only SQL is issued from the user interface code
  - API is primitive SQL: insert, update, delete, select
  - Frameworks like EJB in Java hide the SQL but it is there: single row SQL usually
- Problems
  - Maintenance – code is in the UI code, not centralized

What SmartDB Is

- Starts with relational model
- API is PL/SQL
  - All user interface code calls PL/SQL
  - PL/SQL does SQL: insert, update, delete, select, MORE
  - Logic takes place within the database
  - Logic can issue single-row or array-based or set-based statements

Again

- If you move logic out of the DB, you put more stress on DB
- Save DB cycles by using SmartDB



Toon’s Team’s Testing

- Two row-by-row processing programs
  – One in plain Java (no framework)
  – One in PL/SQL (extreme approach – would never do this)
- Implement a number of business rules
  – Various types of SQL statements
  – 5 million rows
- Observe CPU use
  – No app server process for PL/SQL
  – NoPlSql – JVM on database server, no network
  – How much database CPU?

Results #1

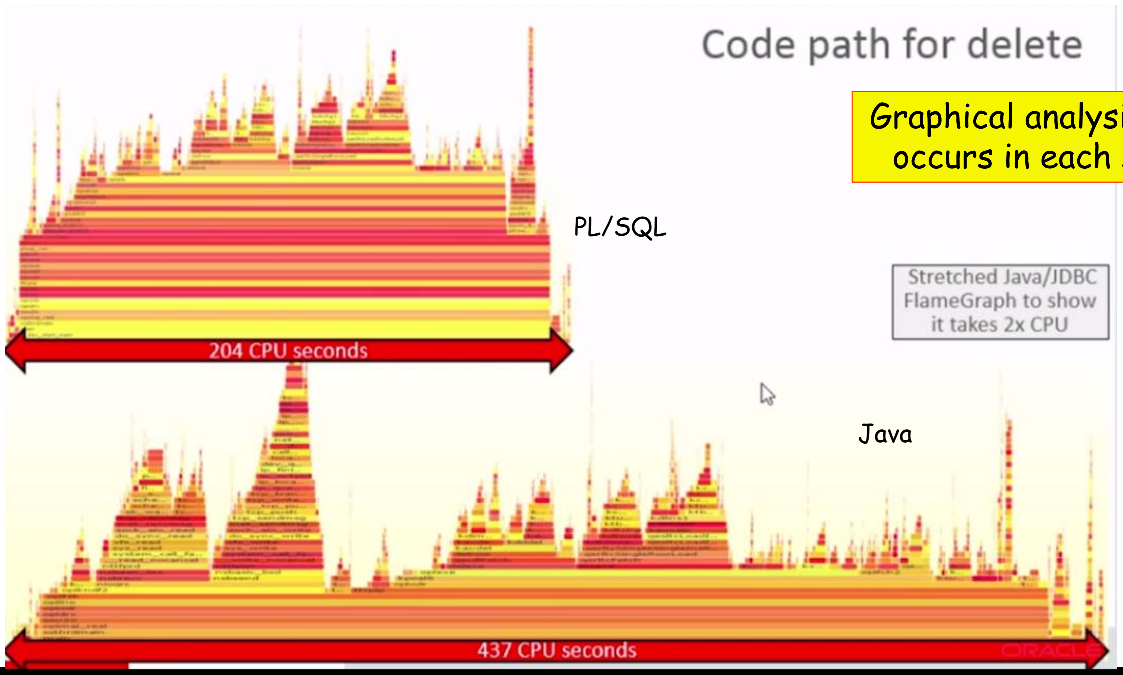
- Java on JDBC
  - 45% DB server
  - 25% CPU JVM – 703 DB-CPU seconds
  - 26 minutes
  - Didn’t add up to 100% Some wait time
  - Log file sync waits – commits row-by-row
- PL/SQL
  - 97% CPU – 371 DB-CPU seconds
  - 6:23 minutes

Results #2

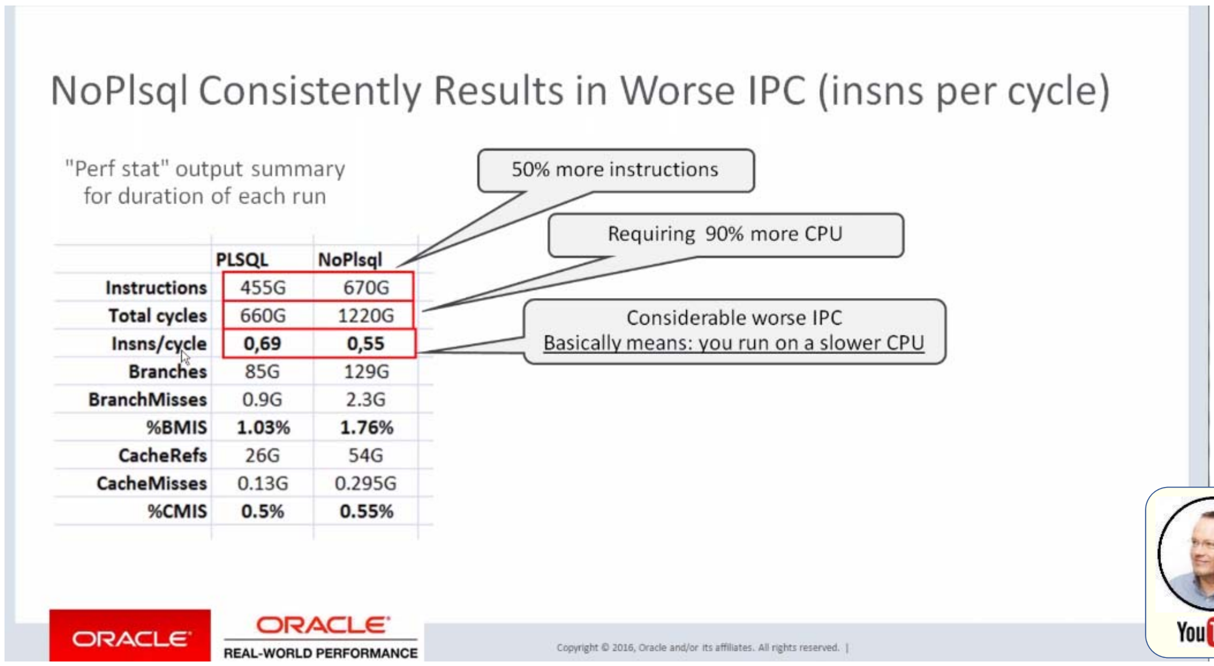
- Commit every 128 rows
- Java on JDBC
  - 437 DB-CPU seconds
  - 11 minutes
  - Closer to 100%
- PL/SQL (does not have sync waits, committing is still expensive)
  - 97% CPU – 204 DB-CPU seconds
  - 3:30 minutes

Results Summarized

- Elapsed drops by 3X, DB-CPU drops by 2X
- Tried in C on OCI
  - 6:54 Minutes
  - 332 DB-CPU seconds
- Why is PL/SQL so much faster?
  - PL/SQL is in the same place as SQL – database, messaging simplified; Java/C incur network hits
- Why is C OCI outperforming Java



Graphical analysis of what occurs in each scenario



NoPlsql Consistently Results in Worse IPC (insns per cycle)

"Perf stat" output summary
for duration of each run

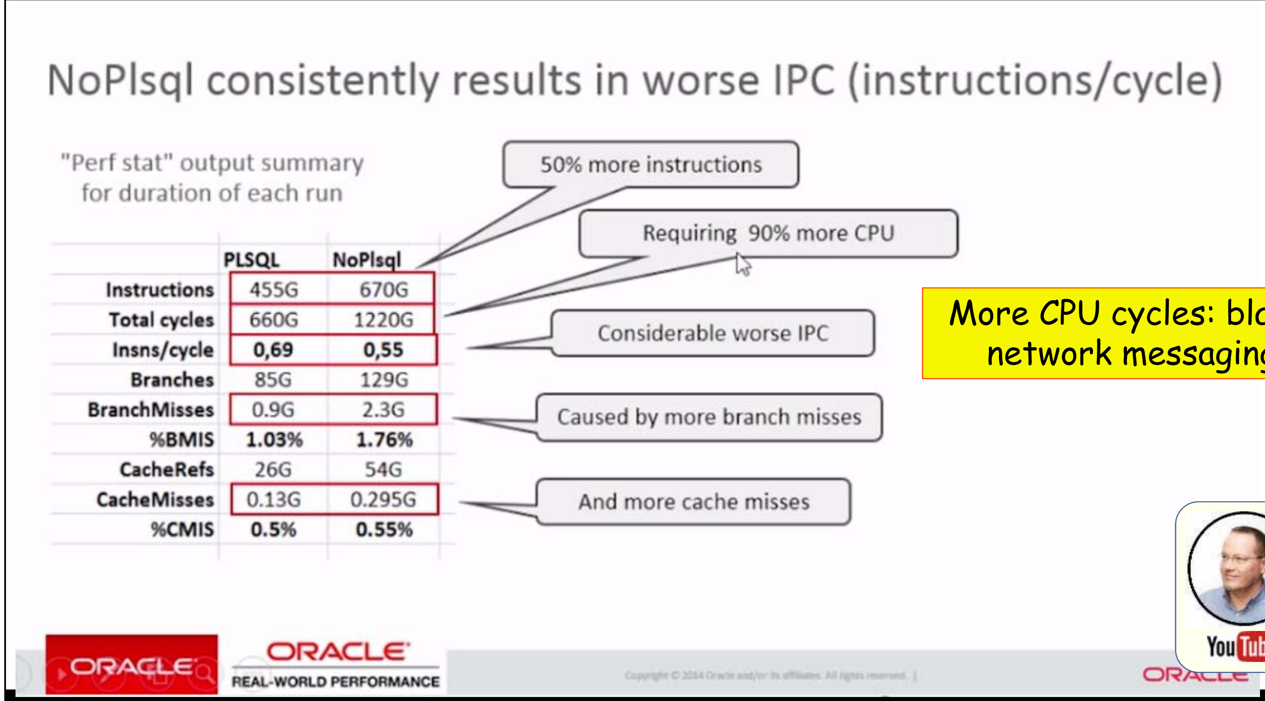
	PLSQL	NoPlsql
Instructions	455G	670G
Total cycles	660G	1220G
Insns/cycle	0,69	0,55
Branches	85G	129G
BranchMisses	0.9G	2.3G
%BMIS	1.03%	1.76%
CacheRefs	26G	54G
CacheMisses	0.13G	0.295G
%CMIS	0.5%	0.55%

50% more instructions

Requiring 90% more CPU

Considerable worse IPC
Basically means: you run on a slower CPU

ORACLE REAL-WORLD PERFORMANCE



NoPlsql consistently results in worse IPC (instructions/cycle)

"Perf stat" output summary
for duration of each run

	PLSQL	NoPlsql
Instructions	455G	670G
Total cycles	660G	1220G
Insns/cycle	0,69	0,55
Branches	85G	129G
BranchMisses	0.9G	2.3G
%BMIS	1.03%	1.76%
CacheRefs	26G	54G
CacheMisses	0.13G	0.295G
%CMIS	0.5%	0.55%

50% more instructions

Requiring 90% more CPU

Considerable worse IPC

Caused by more branch misses

And more cache misses

More CPU cycles: blame network messaging

ORACLE REAL-WORLD PERFORMANCE

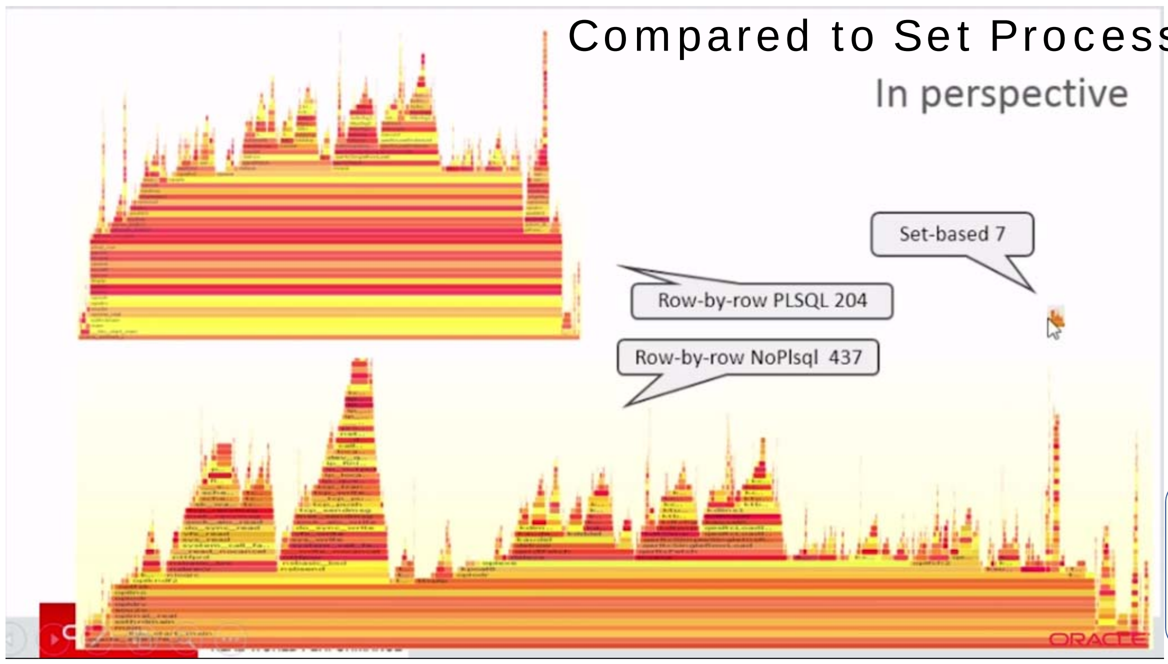
Results

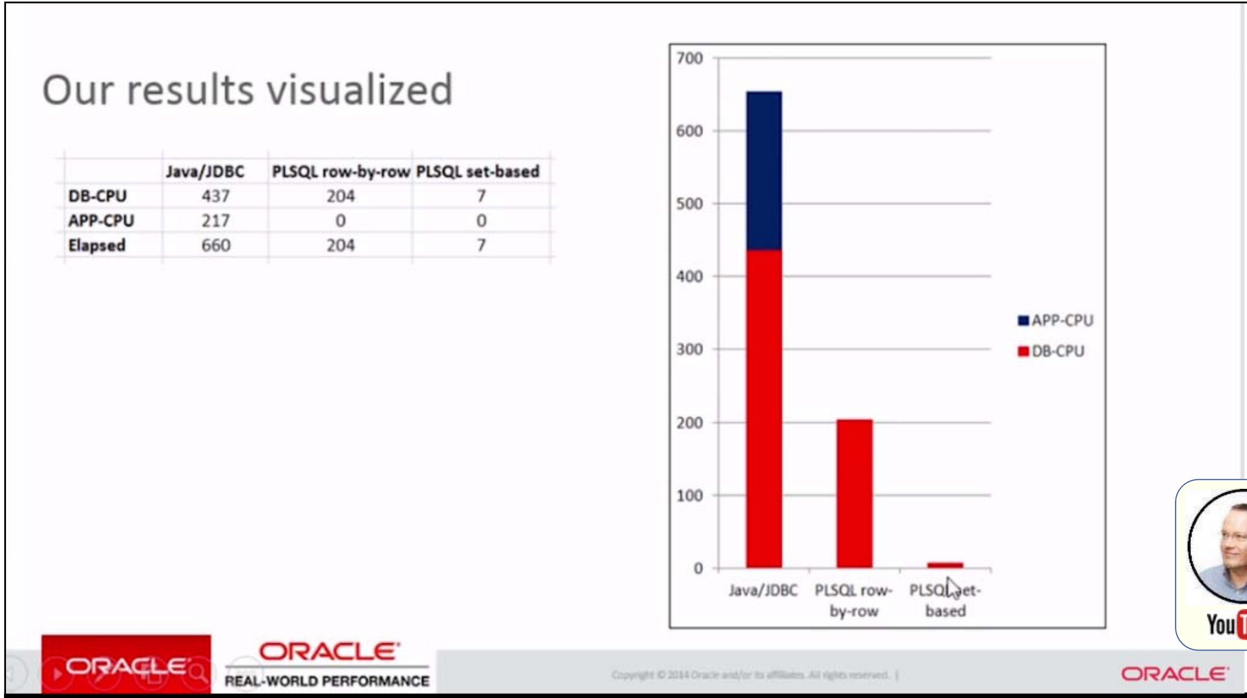
- Single-row SQL, PL/SQL results in 2x speedup
- Business logic in PL/SQL results in 10x speedup
  - NoPlsql has overhead for multiple business logic SQL statements
  - Shipping data in and out of app and db server is expensive
- Another factor not analyzed
  - Network latency and waits

Unique to PL/SQL

- Multi-row handling for the destination of relational tables
  - Set-based: INSERT .. SELECT
  - Array processing: FOR ALL IN … INSERT
  - OO doesn't have this
  - Of course, PL/SQL needs to be written to take advantage of this
- Set-based is 10x to 30x faster than row-by-row in PL/SQL

Compared to Set Processing





Our results visualized

	Java/JDBC	PLSQL row-by-row	PLSQL set-based
DB-CPU	437	204	7
APP-CPU	217	0	0
Elapsed	660	204	7

Findings

- Using DB as processing engine saves CPU licenses
  - Fewer app servers needed to scale for more users
- Moved from NoPlsql to SmartDB
  - Elapsed drops by 3X → #SmartDB is faster
  - DB-CPU drops by 2X → #SmartDB is more scalable

Gets work done faster while at same time using less CPU

Summary of Testing Results

- PL/SQL has none of these
  - No O/S startup for network layers
  - No JVM involvement (for Java option)
    - JVM and JDBC process overhead
    - Shipping data in and out of processes
    - JVM cleanup
  - For C OCI, still have
    - OCI statement preparation startup and cleanup
    - Shipping data in and out

Agenda

Thick Concepts

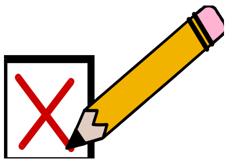
- What is Thick Database?
- Why is Thick faster?
- Thick advice

When Not to Use Thick Database

- If your organization is dedicated to “database independence”
  - Changing from Oracle to SQL Server, for example
    - This is not a simple endeavor
  - Forces applications to use ANSI SQL only
    - Applications are “thicker” than the database
  - Product app companies may need to be DB independent
- If your applications have few or simple business rules
  - Overhead of Thick Database may not be worthwhile

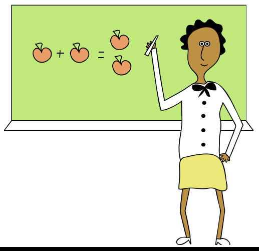
Thick Database Team Success

- If your team is expert in a development discipline other than PL/SQL, selling ThickDB is difficult
  - Experts in database design, SQL, and PL/SQL do not have deep UI skills
  - Experts in UI do not have deep DB skills
  - Or, at least, this is very rare
- Success lies in "divide and conquer"
  - Small team of DB developers
  - Any size team of UI developers
  - Exact numbers depend on the workload
  - Bonus advice: consider Agile



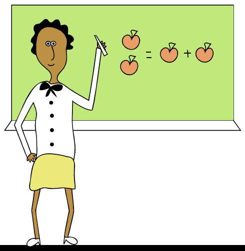
Advice for UI Developers

- Rely on the frameworks to do what they are designed for
  - For example, ADF
    - Highly evolved, efficient, effective use of JDBC; Oracle Support-ed
    - Similarly, EJB
- Writing custom overrides to framework code can lead to disaster
    - Or at least inefficiencies
    - AND: there is no one to blame except yourself



Advice for DB Developers

- Learn the API points into the UI
  - Helps communication with UI developers
- ADF example:
  - doDML() method can call record-level business rules code
    - Constants tell you whether the DML is insert, update, or delete



Do You Need an Oracle Database?

- No, but…
  - A central location for business rules code is necessary
    - Best in a database
  - Views are needed to hide details of the data storage
    - INSTEAD OF triggers may not be available
    - So application may be responsible for calling the central code
  - Table API concept may be possible
    - DB2 supports PL/SQL
    - You can always just allow access to views not tables

How to Transition to Thick Database

- Like applying any other standard while "in flight"
- Apply it 100% to new applications
- Can apply it to existing application enhancements
- Start small
  - Incorporate user interface interaction with database views
- Refine as you go
- "Completely Thick" can be a longer-term goal

Thick Database: A Gastronomic Delight

If the theory turns out to be right, that will be tremendously thick and tasty icing on the cake.

—Brian Greene (1963-), physicist

Summary

- Thick Database is driven by business rules
- Thick Database can improve UI simplicity, productivity, system performance, application accuracy, security
- ThickDB can save cloud computing costs
- ThickDB can use development team talent more efficiently

ORACLE Designer Handbook
Developer Advanced Forms & Reports
ORACLE JDeveloper 3 Handbook
ORACLE9i JDeveloper Handbook
ORACLE JDeveloper 10g Handbook
ORACLE JDEVELOPER for Forms & PL/SQL Developers: A Guide to J2EE Development
ORACLE
Oracle JDeveloper 11g Handbook
A Guide to Fusion Web Development
Duncan Mills
Peter Koletzke
Avrom Roy-Faderman

- **Please fill out the evaluations**
- **7 of 8 books co-authored with Dr. Paul Dorsey, Avrom Roy-Faderman, & Duncan Mills**
- **Slides will be on the NoCOUG website**