"Thick Database" Approach to Web Development

NYC Metro Area
Oracle Users Group Meeting

Dr. Paul Dorsey
Dulcian, Inc.
www.dulcian.com
Four Messages

- 1. Thick database approach is the #1 critical success factor for web architecture
- 2. Use trees for UI navigation
- 3. Views for each page
- 4. 100% is possible.
  - 100% of rules in the database
  - 100% generation
Conventional Wisdom

◆ No logic in the database (database-independent)
  ➢ Database may change.
  ➢ A database is just a persistent copy of the classes.

◆ Place all rules in the middle tier.
  ➢ Write in Java

◆ Class structure is independent of database structure.
  ➢ Integrate using Hibernate or TopLink
Conventional Wisdom = Project Failure

- Standard OO Architecture leads to…
  - Too many round trips to the database
  - Too many queries – no bind variables
  - Too much code
  - Hard to manage complexity
  - Redeploying to the middle tier is harder than redeploying to the database.

- Results
  - It looks like it will work – until stress testing.
  - TERRIBLE performance
    - It should be possible to get good performance.
“Thick Database” Defined

- “Move” code into the database.
- Use Oracle database views based on “Pages.”
- Use database INSTEAD OF triggers to control DML.
- Move logic into the database.
  - Validation
  - Page Navigation
Conventional Wisdom vs. Thick Database

Client -> App Server -> Database

Conventional Wisdom

Data

Thick database
Thick Database Advantages

- Leverage Oracle talent (little retraining)
- Use database for heavy lifting
- J2EE is an evolving environment
  - JSP to JSF
  - Fusion or open-source?
- Less network traffic
- Lower risk
- Easy to refactor
- Less total code
  - Code partitioning
The Main Ideas

◆ Let the database do what it is good at.
  ➢ Crunch data
◆ Let UI developers do what they are good at.
  ➢ Create sophisticated user interfaces
◆ Divide the project cleanly into data and user interface parts.
◆ The more database skills your shop has, the “thicker” the database side should be.
  ➢ A very thick database WILL NOT cause project failure.
  ➢ A very thin database WILL cause project failure.
◆ We need to work together.
Thick Database Approach

◆ Put everything you can into the database.
  ➢ Validation logic
  ➢ Page flow
  ➢ Tree logic
  ➢ Object process flow transitions
  ➢ Screen element display
Thick Database Approach

Advantages

- Scales well
- Easier development
- Requires database skill
- Not optimal – but close
- Will not kill the project
- Uses 50-80% less total code
  - Nicely partitioned DB and UI
J2EE Highway

Database
“super highway”

“water pipe”

Application Server

“soda straw”

Firewall

Browser
Thick Database Ideas (1)

- The application is never thrown an error by the database.
- Page flow logic resides in the database.
- View-only screens (screen portions) are built in the database as HTML.
- A single row view shows:
  - Who is logged in
  - Error message
  - Current menu selection
Thick Database Ideas (2)

- Create 1-2 views (INSTEAD OF triggers) for each screen.
  - Cast object collection to a view.
  - Separate Read-Only and Edit “Views”.
- Store error messages in a table.
- Use a tree for navigation.
  - Many fewer screens
  - Can be driven from the database
Case Study 1

- Batch routine
- Sales goaling

<table>
<thead>
<tr>
<th></th>
<th>Database code # of lines</th>
<th>Java code # of lines</th>
<th>Execution speed</th>
<th>Database development time</th>
<th>Java development Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>0</td>
<td>10000</td>
<td>20 mins, reduced to 20 seconds</td>
<td>1 week (SQL tuning)</td>
<td>6 weeks</td>
</tr>
<tr>
<td>development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thick</td>
<td>500</td>
<td>3000</td>
<td>.2 seconds</td>
<td>1 week</td>
<td>1 week</td>
</tr>
<tr>
<td>database development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©2006 Dulcian, Inc.
Case Study 2

- Two similar OLTP systems
- Both built by Dulcian
- Bug Tracker
  - Senior Java team (not Dulcian trained)
- Complex Ordering
  - Thick database concept
Case Study 2
Project Comparison

◆ Bug Tracker
  ➢ 11 screens
  ➢ Nightmare to maintain
  ➢ Locking, timeouts, etc.

◆ Complex Ordering
  ➢ 28-screen design
    ▪ 10 screens for development
  ➢ Trivial to maintain
Case Study 2
Page Navigation in the Database

- Conventional Wisdom
  - Rats nest page flow diagram

- Thick DB
  - Each pages routes to “dispatcher”
  - Page flow is a star diagram
Case Study 2
Bug Tracker (25% of total diagram)
Case Study 2
Complex Ordering
Case Study 2
Denormalize Page Views

- Each page is one record
  - Flatten master-detail-detail to single record
- 100 column table
  - Strain1Specification1Value
  - Strain1Specification2Value
  - Strain2Specification1Value
  - ...

©2006 Dulcian, Inc.
### MMRRC Requested Items

<table>
<thead>
<tr>
<th>Qty</th>
<th>Units</th>
<th>OID</th>
<th>Item Code</th>
<th>Description</th>
<th>Item Specs</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Litter</td>
<td>7325964</td>
<td>MMRRC:000001-UNC-RESUS</td>
<td>Litter resuscitated from cryo-archive</td>
<td><strong>Resuscitation:</strong> Contact me for specific qty, gender or genotypes.</td>
<td>0</td>
</tr>
</tbody>
</table>
| 2   | EA    | 7327255      | MMRRC:000667-UCD-Cell            | BayGenomics ES cell line GST047  | **ES cell line resequencing:** Yes  
**Micro-injection service:** Yes - Guaranteed | 0    |

**Strain:** B6;D2-Tg(Pcp2-cre)22Lfr/Mmcd

**Strain:** STOCK: Mecp2<sup>tm1.1,jae</sup>/Mmcd

**Strain:** C57BL/6J-Tg(FgaFgbFgg)1Unc/Mmrc
## Case Study 2: Results

<table>
<thead>
<tr>
<th></th>
<th>Java code # of lines</th>
<th>PL/SQL code # of lines</th>
<th>Development time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>13,000</td>
<td>2,800</td>
<td>6 months V1, 6 months V2, 6 months V3</td>
</tr>
<tr>
<td>development</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thick database</td>
<td>2300</td>
<td>3,000</td>
<td>6 weeks V1, 2 weeks V2</td>
</tr>
<tr>
<td>development</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Case Study 3: Super Tree

- API-driven
- All logic in the database
- Keeps copy of tree in the database
- APIs
  - NewTree (type)
  - Expand (node)
  - GetMenu (node)
  - NodeSelect (node)
  - MenuSelect (menuItem)
- Return result actions as XML
Case Study 3
Right-click Leads Working

◆ Fired by application when right-clicking Leads Working node

api$tree.f_main('14994901', 'GetMenu', -80, 'RecrtrMain')

◆ Result returned from database

  <actionSet>
  <Menu ID="-80" >
   <tran display="Create New Lead" ID="-15240" action="330"
    RC_Desk_oid="737003" />
  <tranLine ID="-15250" />
  <tran display="Power Calling" ID="-15260" action="20" />
  <tran display="Mail Merging" ID="-15270" action="20" />
  <tran display="E-Mail Merging" ID="-15280" action="20" />
  </Menu>
  </actionSet>
## Case Study 3: Results

- **Tree UI Control (like SQL Navigator)**

<table>
<thead>
<tr>
<th></th>
<th>Database code # of lines</th>
<th>Java code # of lines</th>
<th>StrutsConfig.xml # of lines</th>
<th>Database development time</th>
<th>Java development Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional development</td>
<td>2300</td>
<td>13000</td>
<td>657</td>
<td>2 weeks</td>
<td>6 months</td>
</tr>
<tr>
<td>Thick database development</td>
<td>3900</td>
<td>2800</td>
<td>98</td>
<td>2 weeks</td>
<td>1 week</td>
</tr>
</tbody>
</table>
Case Study 4: Thick DB is NOT a Silver Bullet

Version 1 – Java:
- Large batch routine
- 14,000,000 records
  - 100 columns in each record
  - Each read = 100 SELECTs (getters)
  - Each write = 1 insert and 99 UPDATEs (setters)
  - 1 minute per record = 26.5 years (month-end batch) on a 64 CPU Ultra-Spark

Version 2 – PL/SQL:
- Refactor code to the database.
- Use the same wrong algorithm
- Still poor performance

Mindless refactoring to PL/SQL does not guarantee success!
100% is Possible (well at least 99%)

◆ 100% of all code in the database
  - At worst, 1-3 round trips per user interface action
  - Rules in the application server cause MANY more round trips.

◆ 100% application generation
  - All of the rules are specified.
  - All rules are generated or accessed at runtime.
Contact Info

- Dr. Paul Dorsey – paul_dorsey@dulcian.com
- Dulcian website - www.dulcian.com

Available now!

Oracle PL/SQL for Dummies