Politics of Persistence

Donald Smith
Oracle Corporation
Goal

Have fun, and MAKE FUN, of the never ending persistence battles between Java Developers and DBAs, and within the Java community itself.
Managing persistence related issues is the most underestimated challenge in enterprise Java today – in terms of complexity, effort and maintenance.
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - Mapping, Queries, Transactions, Business Logic, Locking
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - Mapping, Queries, Transactions, Business Logic, Locking
About the Audience

- “Sympathies with” Java Developer or DBA?
- Fortune 500/Global 5000, SMB, or Startup?
- Does your organization predate:
  - Web Services?
  - The Web?
  - The Database?
  - The Computer?
  - WW2?
Java Developer Stereotype

- Young
- Pony tail
- Random piercing and tattoos
- T-Shirt, Jeans (at best)
- Everything should be free, open and transparent
- Java plus the random scripting language of the day can build any enterprise app
DBA Stereotype

- Uh... Let’s say “Mature”
- Big gray beard
- Dress code changes by ambition
- Everything should be expensive, closed and proprietary
- Java is YAPL, and I’ve seen a dozen come and go over the last several decades
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping, Queries, Transactions, Business Logic, Locking*
Two Diverse Perspectives

- **DAP Inc.**
  - Mature, Fortune 500
- **Bubble Inc.**
  - New, pre IPO
Case Study: DAP Inc.

- Data intense business services company founded in 1950
- Early adopter of database technology starting in 1978 (almost 30 years!)
- Rapid growth in 80’s and 90’s, IT became cornerstone of business
- CEO began career as entry level DBA
  ➢ Guess what his exec staff looks like?

Fairly typical of Fortune 500 / Global 5000
DAP Inc. Technical Evolution

- Business logic entrenched in the database from day 1
- Lots (and lots) of COBOL apps
- Plus fingers in a lot of technology pies
  - Basic
  - Smalltalk
  - C
  - PL-SQL
  - Many 4GL Languages
Along Comes the Web...

- Early adopters in web technology (mid 90’s)
- Strong believers in need for what we’d now call “Agile Development”
- BIG PROBLEM
  - Web + OOAD desires implies Java development
  - History implied DBA command and control
Case Study: Bubble Inc.

- Web service* for automating a niche sales function
- Founded in 2000 by 2 Stanford business grads, grown in 4 years to 49 staff
- Privately funded
- CEO Expects “idea to implementation” cycle to be < 2 weeks, loves to be on bleeding edge

* Not in the pejorative sense 😊
Bubble Inc. Technical Evolution

- Database “just tables and indexes”
- Business logic entrenched in mid tier
- Except for a few scripts here and there, exclusively Java shop
- Legacy apps == Java
Bubble Inc. Rapid Growth

- Bubble Inc. becomes major success
- Struggles with many DB related problems
  - Security
    - Customers don’t like sharing a single DB instance with competitors
  - Scaling
    - Close to a terabyte of data access weekly
  - Customization
    - Customers want highly customized data types and query capabilities
Challenges

- DAP and Bubble Inc. challenges and stereotypical behavior will be discussed through rest of session...
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping, Queries, Transactions, Business Logic, Locking*
Let’s Beat a Dead Horse…

- Let’s start by flogging the “Impedance Mismatch” story, ONE MORE TIME…
- It’s important because…

...*Impedance Mismatch* is MORE than a technical challenge, it’s also a business and political issue!
Impedance Mismatch

- The differences in relational and object technology is known as the "object-relational impedance mismatch"
- Challenging problem to address because it requires a combination of relational database and object expertise
## Impedance Mismatch

<table>
<thead>
<tr>
<th>Factor</th>
<th>J2EE</th>
<th>RDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Data Format</td>
<td>Objects, methods, inheritance</td>
<td>Tables, SQL, stored procedures</td>
</tr>
<tr>
<td>Scale</td>
<td>Hundreds of megs</td>
<td>Gigabytes, terabytes</td>
</tr>
<tr>
<td>Relationship</td>
<td>Memory references</td>
<td>Foreign keys</td>
</tr>
<tr>
<td>Uniqueness</td>
<td>Internal object identity</td>
<td>Primary keys</td>
</tr>
</tbody>
</table>
# Impedance Mismatch

<table>
<thead>
<tr>
<th>Factor</th>
<th>J2EE</th>
<th>RDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Data Format</td>
<td>Objects, methods, inheritance</td>
<td>Tables, SQL, stored procedures</td>
</tr>
<tr>
<td>Scale</td>
<td>Hundreds of megs</td>
<td>Gigabytes, terabytes</td>
</tr>
<tr>
<td>Relationship</td>
<td>Memory references</td>
<td>Foreign keys</td>
</tr>
<tr>
<td>Uniqueness</td>
<td>Internal object identity</td>
<td>Primary keys</td>
</tr>
<tr>
<td>Key Skills</td>
<td>Java development, object modeling</td>
<td>SQL, Stored Procedures, data management, Data modeling</td>
</tr>
<tr>
<td>Tools</td>
<td>IDE, Source code management, Object Modeler</td>
<td>Schema designer, query manager, performance profilers, database config</td>
</tr>
<tr>
<td>Corporate Org. Structure</td>
<td>“Newer technology” often with weak organizational ties to database mgmt</td>
<td>Often mature infrastructure with significant legacy considerations</td>
</tr>
</tbody>
</table>
Business vs Technical IM

- Technical Impedance Mismatch can usually be overcome
  - Just need to find the tradeoff or reset expectations

- Business Impedance Mismatch not always possible to overcome without guts
  - May require re-think about how you’re organized and develop software
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping, Queries, Transactions, Business Logic, Locking*
Persistence in Java

- Three Major Strategies
  - OODBMS
  - “Extending the database” into Java
    - Straight JDBC
    - iBatis and other JDBC wrapper approaches
    - BC4J and other code gen frameworks
  - Persistence Layers
OODBMS

- Forays into niche markets
  - Derivatives, Genomics
- Good for persisting complex object models
- Not so good at:
  - Ad hoc queries
  - Reporting
  - Schema evolution
  - Befriending Relational DBAs
JDBC

- Hardcode SQL in Java
- Manage database semantics in Java
- May have wrappers and other utility layers
Persistence Layer

Object-level querying and creation results are objects

API uses SQL or database specific calls

Results are returned as raw data

Object creation and updates through object-level API

Objects

Persistence Layer

JDBC

SQL

Rows

Objects

J2EE & Web Services
Different Persistence Layers

- Home Grown
- Proprietary API
  - TopLink
  - Hibernate
- Standards based
  - EJB2
    - TopLink
    - App Servers
  - EJB3
    - TopLink
    - Hibernate
    - Solarmetric
    - etc.
A Quick Word on Spring

- Spring is not itself a persistence layer
- Spring specifies a common way that all persistence layers can be accessed and leveraged
- The Spring Team have become excellent advocates for the importance of persistence layer abstraction and standards
Let’s Generalize…

- Bubble Inc.
  - Likely to be more embracing of persistence layers
    - Unless you have “convert” DBA Java developers

- DAP Inc.
  - Likely to think of Java-Database development as “extension of DB”
  - Hiring “fresh talented” Java developers will lead to challenges with persistence layers
History of Java Persistence

- Disclaimers:
  - There are 100’s of “persistence layers” that have come (and gone), not realistic to include them all
    - JavaBlend, SQL2Java, Solarmetric, RougeWave, Apple WOF, OJB, JRF, ObjectBridge, JGrinder, Castor, JORM, JAWS, Cayenne, Tourque, Cocobase, Objectmatter, Basejdao, Versant, PLayer, Artyoum, SimpleORM, Jasmin, JDX, Powertier, JOP, ROF, Voyager, Poet, Xcalica, QBeans, Genie, Powermap, JCredo, BC4J, ObjectDB, ObjectFrontier, intelliBO, PE:J, RexIP, JRelay, Orient Technologies, JDOMax, JDOInstruments, Speedo, Cocoon, TriactiveJDO, XORM, JPOX, Druid, FirestormJDO, eclipseJDO, ...
  - Focus on specs and TopLink / Hibernate
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping, Queries, Transactions, Business Logic, Locking*
Business Behind Conflict

- **Java Developers**
  - **WE** know how to respect the ilities
    - Maintainability, reusability, scalability
  - **We’re new, we’re hot, we’re in demand!**

- **DBA**
  - **WE** have been running the shop since before you were born, and ain’t going to hand over the keys to the flavor of the day
“Problematic” Org Chart

Mary
CEO
San Jose

Bob
VP Sales
San Jose

Dan
Infrastructure Director
San Jose

Jeff
VP Engineering
San Jose

Alice
Product X Dev Manager
Ottawa

Sue
VP Product Management
San Jose

Rick
Product X PM
London UK

Ken
Data Services Manager
Kansas

Alice 1

Joe
Java Tech Manager
Dallas

Eng 1
Eng 2

GU 1
Flow 1

Alice 2

DBA 1
DBA 2

Eng

GUI 2
Flow 2
“More Optimistic” Org Chart

Mary
CEO
San Jose

Bob
VP Sales
San Jose

Jeff
VP Engineering
San Jose

Dan
Corporate Standards
San Jose

Sue
VP Product X
San Jose

Ken
Data Services Stds
Kansas

Joe
Java Tech Stds
Dallas

Alice
Product X Dev Manager
Ottawa

Rick
Product X PM
London UK

Net result? Alice has a fighting chance..
Org Chart Discussion

- Which chart are you more likely to find at DAP? Bubble Inc?

- Sometimes it’s not possible, or desirable, to transition away from “problematic” charts
  - “Problematic” may be essential to business
    - Scale, overlap, security
Degree of DBA Control  2x2
Degree of DBA Control  2x2

<table>
<thead>
<tr>
<th>Age &amp; Entrenchment of IT Department</th>
<th>Segregation of Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong DBA control</td>
<td>Cooperative opportunity</td>
</tr>
<tr>
<td>Intense DBA control</td>
<td>Depends on Executive Background</td>
</tr>
</tbody>
</table>
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping, Queries, Transactions, Business Logic, Locking*
J2EE Developer Desires

- Data model should not constrain object model
- Don’t want database code in object/component code
- Accessing data should be fast
- Minimize calls to the database – they are expensive
- Object-base queries – not SQL
- Isolate J2EE app from schema changes
- Would like to be notified of changes to data occurring at database
DBA Desires

- Adhere to rules of database (referential integrity, stored procedures, sequence numbers, etc.)
- Build the J2EE application but do NOT expect to change schema
- Build the J2EE application but the schema might change
DBA Desires

- Let DBA influence/change database calls/SQL generated to optimize
- Be able to log all SQL calls to database
- Leverage database features where appropriate (outer joins, sub queries, specialized database functions)
Differences

- Desires are contradictory
  - “Insulate application from details of database but let me leverage the full power of it”
  - Different skill sets
  - Different methodologies
  - Different tools

- Technical differences must also be considered!
Discussion

- As an experienced DBA, would you rather work at DAP or Bubble Inc? Why?
- As a known Java GURU, would you rather work at DAP or Bubble Inc? Why?
- What do you think this means for the kinds of people attracted to DAP and Bubble Inc?
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping*, Queries, Transactions, Business Logic, Locking
Mapping

- Object model and Schema must be mapped
  - True for most persistence approaches
- First contentious issue facing teams
  - Which classes map to which table(s)?
  - How are relationships mapped?
  - What data transformations are required?
  - Who “bends” in the name of performance and the “ilities”
Data and Object Models

- Flexible mapping capabilities provide data and object models a degree of independence.
- Otherwise, business object model will force changes to the data schema or vice-versa.
- Often, J2EE component models are nothing more than mirror images of data model or vice versa – NOT desirable.
- Bubble Inc. just changes something, DAP Inc. has a challenge on their hands!
Simple Object Model

Customer

id: int
name: String
creditRating: int

Address

id: int
city: String
zip: String

1:1 Relationship
Other Possible Schemas...
Even More Schemas...
Mapping Summary

- Just showed **nine** valid ways a 1-1 relationship could be represented in a database
  - Most persistence layers and application servers will only support **one**
- Without good support, designs will be forced
- Imagine the flexibility needed for other mappings like 1-M and M-M
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch \textit{with a twist}
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - \textit{Mapping, Queries, Transactions, Business Logic, Locking}
Queries

- Java developers are not usually SQL experts
  - Maintenance and portability become a concern when schema details hard-coded in application
- Allow Java based queries that are translated to SQL and leverage database options

```java
employee.manager.address = someAddress
```

Generates...

```sql
SELECT * FROM EMP t1, EMP t2, ADDR t3
WHERE t1.MGR_ID = t2.EMP_ID AND t2.ADDR_ID = t3.ADDR_ID AND t3.ADDR_ID = <someAddress.id>
```
Queries

- **Bubble Inc.**
  - May never care to see or touch SQL (if it’s good enough not to be problematic)
  - May be open minded to various object query languages and formats

- **DAP Inc.**
  - You better have hooks for all things SQL related
  - Trace, tune, debug, log, change, optimize
  - SQL and Stored Procs interchangable
  - Leverage DB – Outer joins, hierarchal queries
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping, Queries, Transactions, Business Logic, Locking*
Transaction Management

- J2EE apps typically support many clients sharing small number of db connections
- Ideally would like to minimize length of transaction on database
Transactions

- Bubble Inc.
  - More likely to run long UOW
  - Use of persistence layers can help
- DAP Inc.
  - Will closely monitor txns
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics

  - Mapping, Queries, Transactions, *Business Logic*, Locking
Some Definitions

- **Database Integrity**
  - Ensuring consistence in the database
  - FK’s match up, calculations up to date

- **Database Triggers**
  - Used to ensure integrity
  - Calculations (“total purchases” on new order)

- **Cascade Deletes**
  - Special kind of database trigger that deletes “primary owned relationships” (delete address when deleting customer)
Cascaded Deletes

Cascaded deletes done in the database have a real effect on what happens at J2EE layer

Middle tier app must:

- Be aware a cascaded delete is occurring
- Determine what the “root” object is
- Configure persistence settings or application logic to avoid deleting related objects already covered by cascaded delete
Cascade Deletes

- Bubble Inc.
  - Java developers likely will manage in JVM and run deletes explicitly on the DB

- DAP Inc.
  - DBA will likely want control and management
  - How to synch?
Database Triggers

- Without intervention, database triggers will be completely transparent to the J2EE application
- However, their effects must be clearly communicated and considered
- Example: Data validation → audit table
  - Objects mapped to an audit table that is only updated through triggers, must be read-only on J2EE
Database Triggers

- More challenging when trigger updates data in the same row and the data is also mapped into an object
- Example: Annual salary change automatically triggers update of life insurance premium payroll deduction
  - J2EE app would need to re-read payroll data after salary update OR
  - Duplicate business logic to update field to avoid re-read
  - Saves a DB call but now business logic in 2 places
Database Triggers

- **Bubble Inc.**
  - Java developers likely will manage in JVM and update database explicitly

- **DAP Inc.**
  - Will likely have lots of legacy business logic in DB
  - How to interact with Java Developers?
Referential Integrity

- Java developers manipulate object model in a manner logical to the business domain
- May result in ordering of INSERT, UPDATE and DELETE statements that violate database constraints
- Persistence layer should automatically manage this and allow options for Java developer to influence order of statements
Referential Integrity

- Bubble Inc.
  - Will just turn off constraint checking in DB

- DAP Inc.
  - Will insist Java developers figure it out...
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - Mapping, Queries, Transactions, Business Logic, Historization, Locking
Insurance Historization

Example

- Mapping is static, but what objects to recover is based on dynamic information
- Can be done, but not very transparently, especially for relationships
**Difficult Case – “Historization”**

- Composite primary key, consisting of “real” pkey and date range

<table>
<thead>
<tr>
<th>Policy_ID</th>
<th>Start_Date</th>
<th>End_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1035</td>
<td>1/12/1999</td>
<td>2/5/2002</td>
</tr>
<tr>
<td>1035</td>
<td>2/6/2002</td>
<td>12/10/2005</td>
</tr>
</tbody>
</table>
Historization

- **Bubble Inc.**
  - Likely to tackle this as an object modeling, business logic problem

- **DAP Inc.**
  - Likely to attack it as a Schema/DBA feature challenge
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - Mapping, Queries, Transactions, Business Logic, Caching, Locking
Caching

- Any application that caches data, now has to deal with stale data
- When and how to refresh?
- Will constant refreshing overload the database?
- Problem is compounded in a clustered environment
- App server may want be notified of database changes
Caching

** OO Query  **

NO – Build bean/object from results

Does PK for row exist in cache?

YES – Get from Cache

Return object results

** SQL Query  **

Results(s)
Caching

- Bubble Inc.
  - Will see caching as critical to performance and scaling

- DAP Inc.
  - May feel threatened by cache, in memory queries and may be concerned with refreshing/integrity issues
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping, Queries, Transactions, Business Logic, Locking*
Locking

- J2EE Developers want to think of locking at the object level
- Databases may need to manage locking across many applications
- Persistence layer or application server must be able to respect and participate in locks at database level
Optimistic Locking

- DBA may wish to use version, timestamp and/or last update field to represent optimistic lock
  - Java developer may not want this in their business model
  - Persistence layer must be able to abstract this
- Must be able to support using any fields including business domain
Pessimistic Locking

- Requires careful attention as a JDBC connection is required for duration of pessimistic lock
- Should support SELECT FOR UPDATE [NOWAIT] semantics
Locking

- Bubble Inc.
  - May treat locking as an application issue

- DAP Inc.
  - Will want control over and encourage optimistic locking semantics
Other Impacts

- Use of special types
  - BLOB, Object Relational
- Open Cursors
- Batch Writing
- Sequence number allocations
Agenda

- About the Audience and Stereotypes
- Two diverse cases for discussion
- Impedance Mismatch *with a twist*
- Major Persistence Strategies and Evolution
- Business Politics
- Technical Politics
  - *Mapping, Queries, Transactions, Business Logic, Locking*
- Summary
Deep thoughts...

- Causality question:
  - Are “problematic” org charts likely to choose certain technologies, and therefore, can we blame the generalized “failures” of certain tech on the tech itself?
Summary

- More to Impedance Mismatch than technical challenges