"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

Data

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.

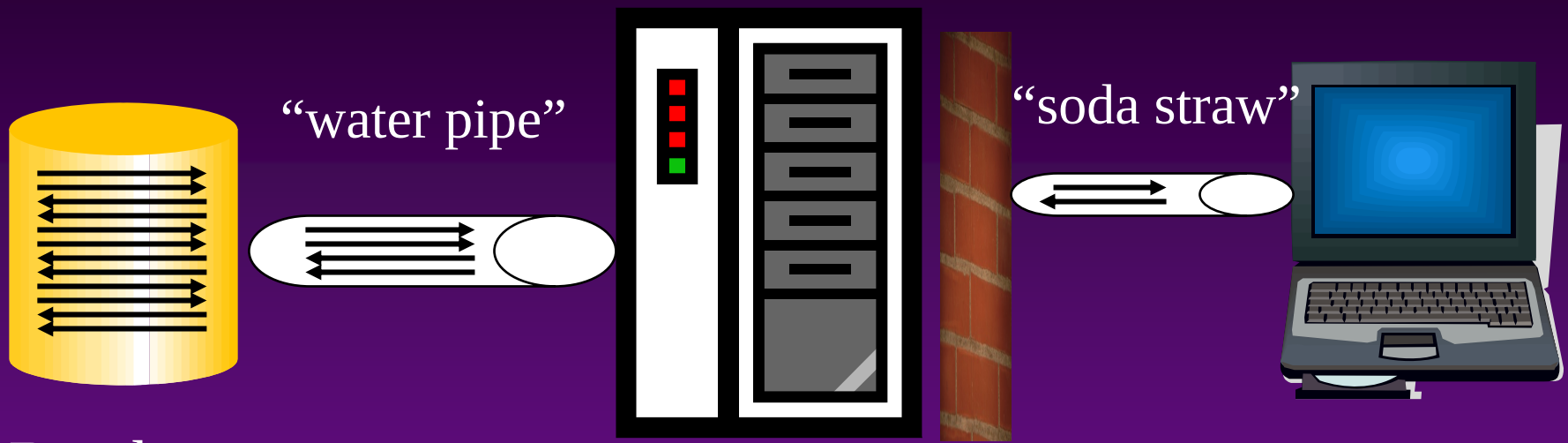
Thick Database Approach

- 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”

Application Server

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

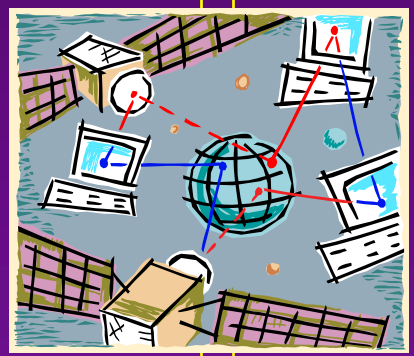
	Database code # of lines	Java code # of lines	Execution speed	Database development time	Java development Time
Conventional development	0	10000	20 mins, reduced to 20 seconds	1 week (SQL tuning)	6 weeks
Thick database development	500	3000	.2 seconds	1 week	1 week

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

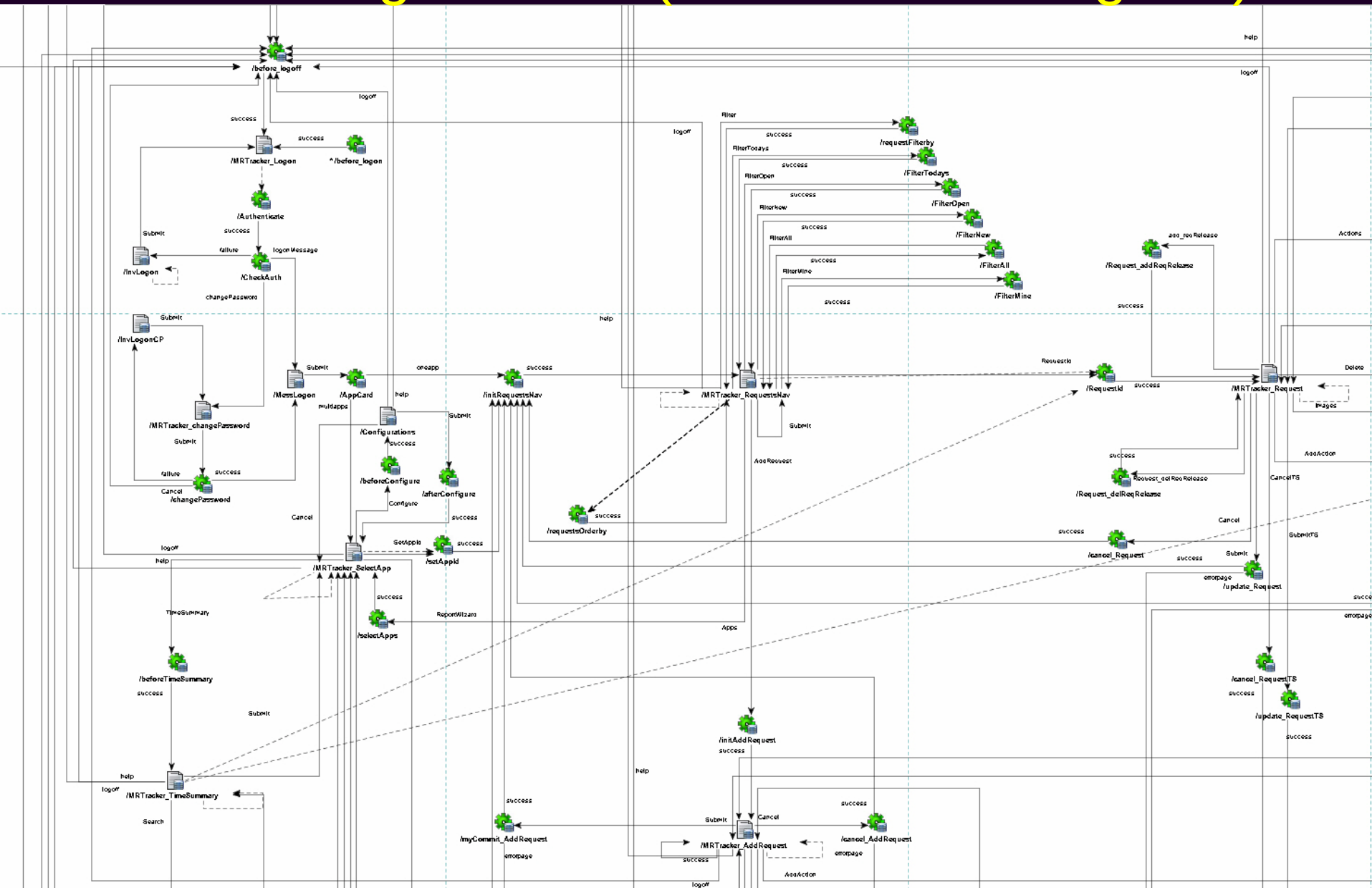
- Bug Tracker
  - 11 screens
  - Nightmare to maintain
  - Locking, timeouts, etc.
- Complex Ordering
  - 28-screen design
    - 10 screens for development
  - Trivial to maintain



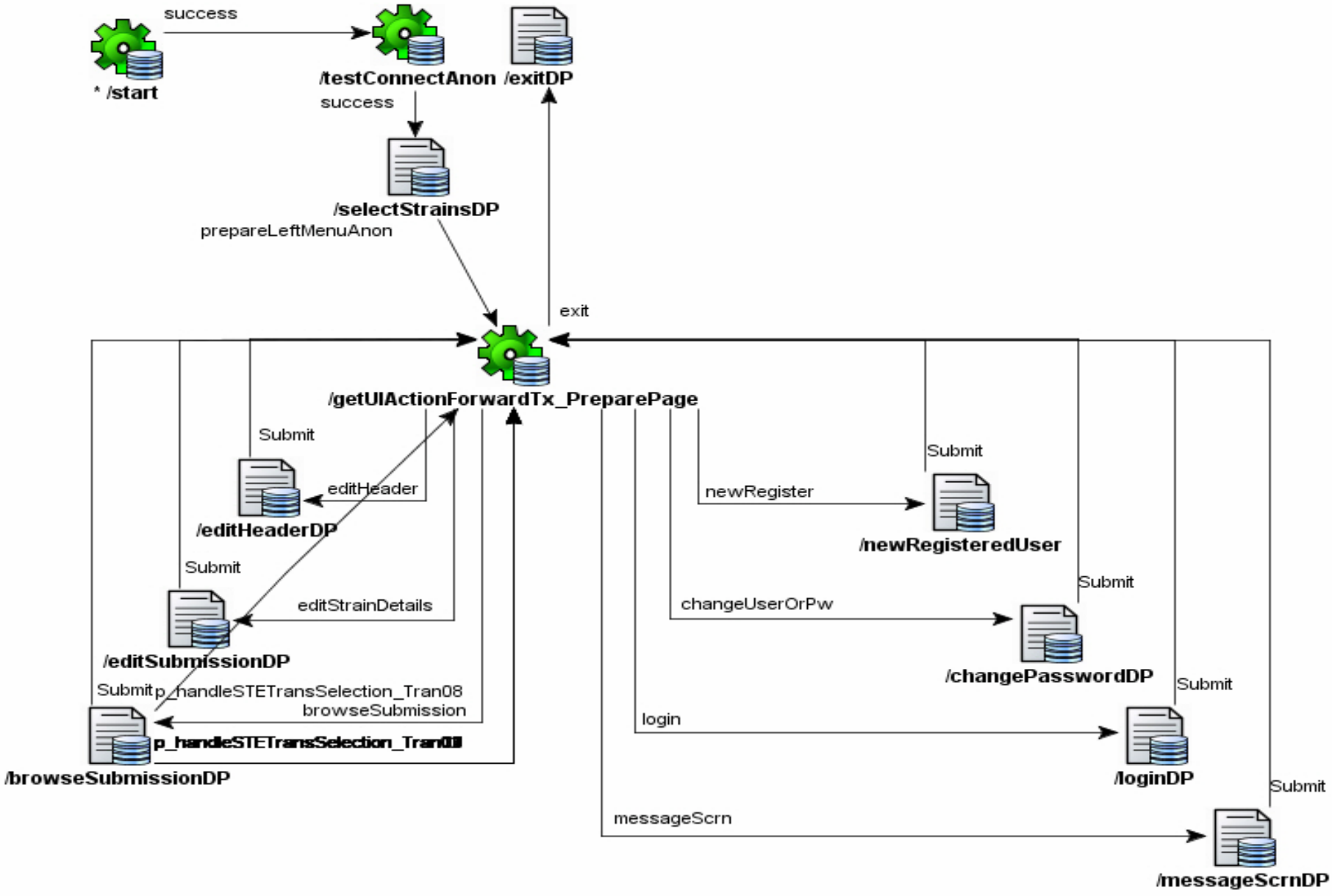
Case Study 2
Page Navigation in the Database

- 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)



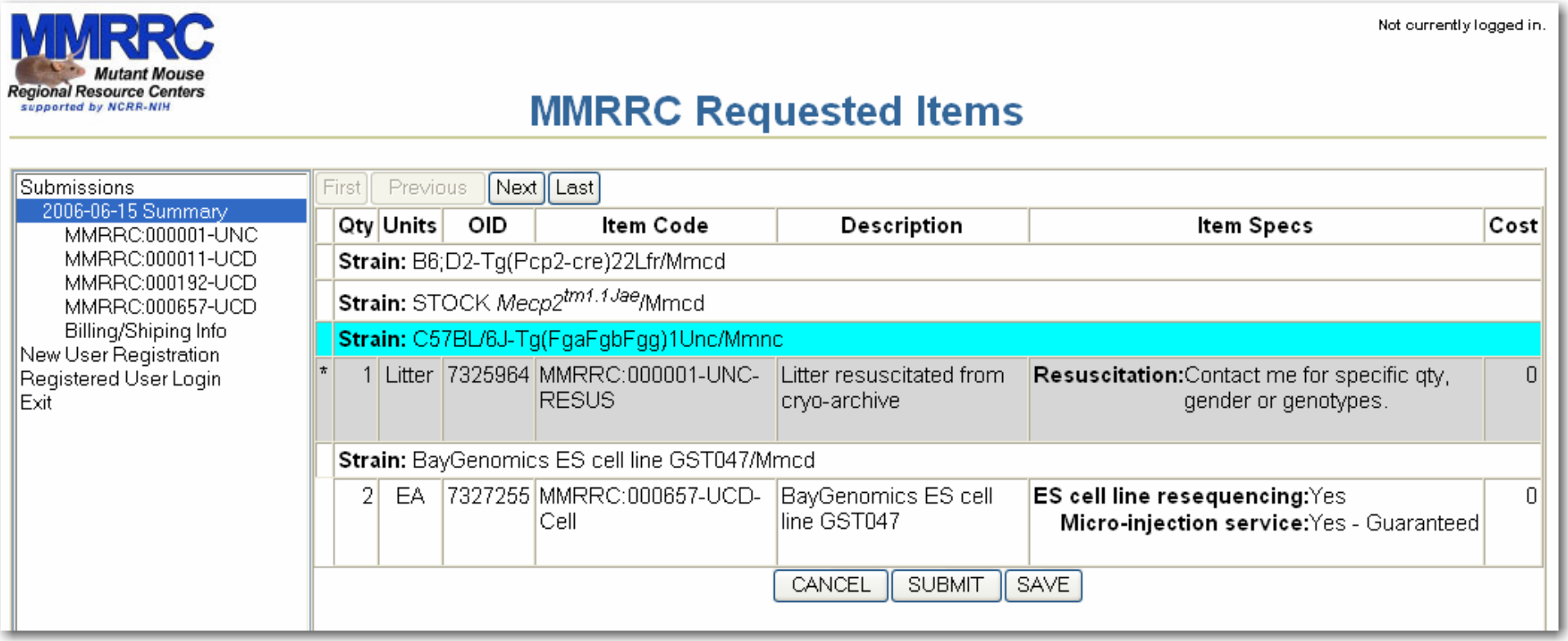
Case Study 2
Complex Ordering



Case Study 2
Denormalize Page Views

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

Case Study 2
Complex Order Screen



MMRRC Mutant Mouse Regional Resource Centers supported by NCRR-NIH

Not currently logged in.

MMRRC Requested Items

Submissions
- 2006-06-15 Summary
  - MMRRC:000001-UNC
  - MMRRC:000011-UCD
  - MMRRC:000192-UCD
  - MMRRC:000657-UCD
  - Billing/Shiping Info

New User Registration
Registered User Login
Exit

First | Previous | Next | Last

	Qty	Units	OID	Item Code	Description	Item Specs	Cost
	Strain: B6;D2-Tg(Pcp2-cre)22Lfr/Mmcd						
	Strain: STOCK *Mecp2*$^{tm1.1Jae}$/Mmcd						
	Strain: C57BL/6J-Tg(FgaFgbFgg)1Unc/Mmnc						
*	1	Litter	7325964	MMRRC:000001-UNC-RESUS	Litter resuscitated from cryo-archive	**Resuscitation:**Contact me for specific qty, gender or genotypes.	0
	Strain: BayGenomics ES cell line GST047/Mmcd						
	2	EA	7327255	MMRRC:000657-UCD-Cell	BayGenomics ES cell line GST047	**ES cell line resequencing:**Yes **Micro-injection service:**Yes - Guaranteed	0

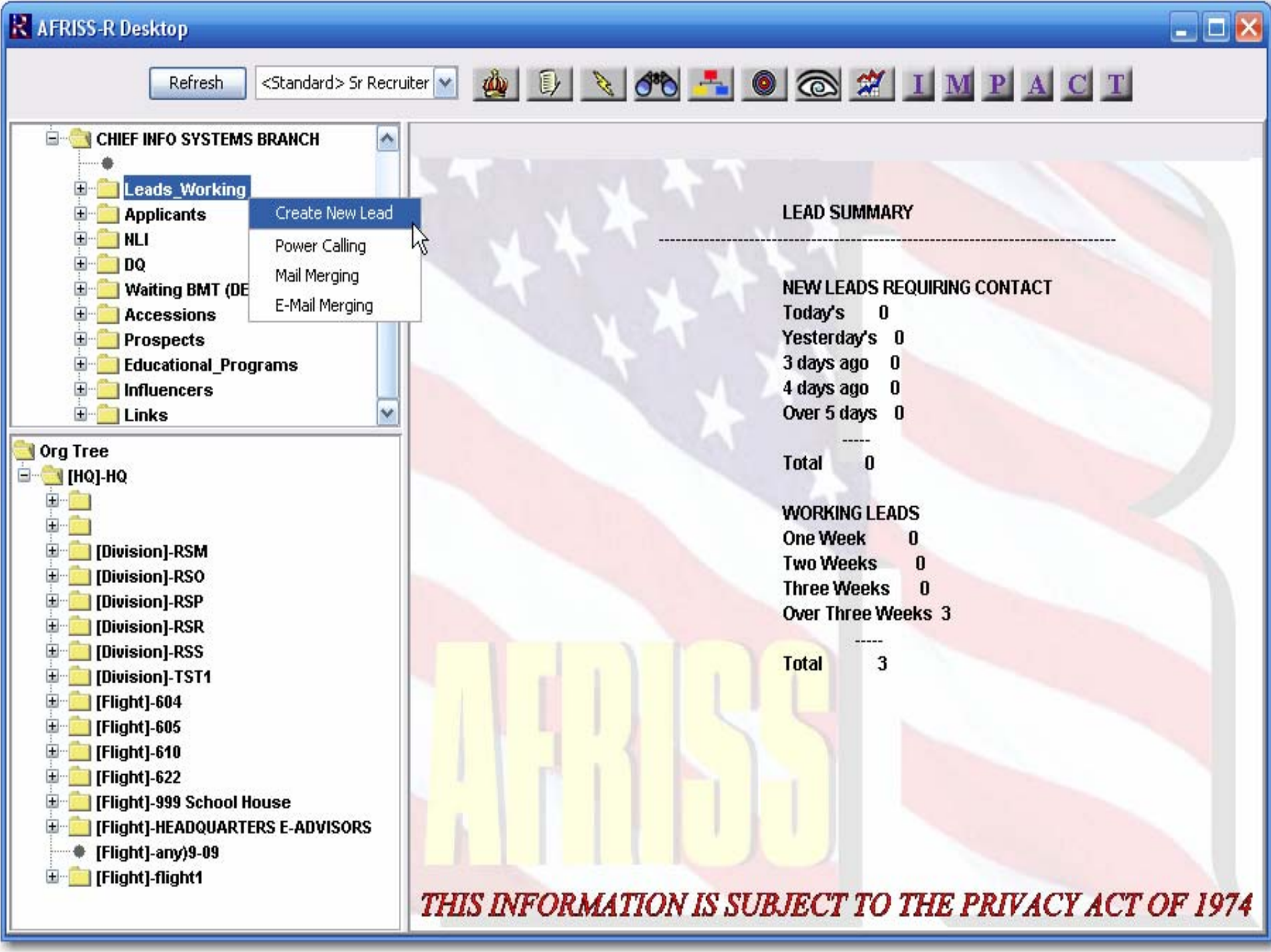
CANCEL | SUBMIT | SAVE

Case Study 2: Results

	Java code # of lines	PL/SQL code # of lines	Development time
Conventional development	13,000	2,800	6 months V1 6 months V2 6 months V3
Thick database development	2300	3,000	6 weeks V1 2 weeks V2

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


AFRISS-R Desktop
Refresh
<Standard> Sr Recruiter
I M P A C T
CHIEF INFO SYSTEMS BRANCH
Leads_Working
Applicants
NLI
DQ
Waiting BMT (DE
Accessions
Prospects
Educational_Programs
Influencers
Links
Create New Lead
Power Calling
Mail Merging
E-Mail Merging
Org Tree
[HQ]-HQ
[Division]-RSM
[Division]-RSO
[Division]-RSP
[Division]-RSR
[Division]-RSS
[Division]-TST1
[Flight]-604
[Flight]-605
[Flight]-610
[Flight]-622
[Flight]-999 School House
[Flight]-HEADQUARTERS E-ADVISORS
[Flight]-any)9-09
[Flight]-flight1
LEAD SUMMARY
NEW LEADS REQUIRING CONTACT
Today's 0
Yesterday's 0
3 days ago 0
4 days ago 0
Over 5 days 0
Total 0
WORKING LEADS
One Week 0
Two Weeks 0
Three Weeks 0
Over Three Weeks 3
Total 3
AFRISS
THIS INFORMATION IS SUBJECT TO THE PRIVACY ACT OF 1974

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>
```

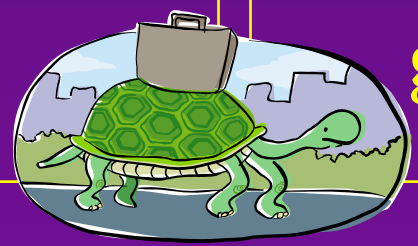
Case Study 3: Results

- Tree UI Control (like SQL Navigator)

	Database code # of lines	Java code # of lines	StrutsConfig .xml # of lines	Database development time	Java develop-ment Time
Conventional development	2300	13000	657	2 weeks	6 months
Thick database development	3900	2800	98	2 weeks	1 week

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