The Path to Oracle Fusion Using a Thick Database Approach

Dr. Paul Dorsey

Dulcian, Inc.

www.dulcian.com

NYC Metro Area Oracle Users Group Meeting

October 2, 2007

What you will get out of this presentation

(This is NOT an Oracle approved message)

- All of the different things that “Fusion” means

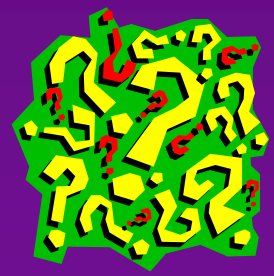
What parts of Fusion are worth learning about

  - DBMS, OAS, ADF BC, ADF Faces

- Which parts can be ignored
  - BPEL, BAM, Oracle Business Rules
- Explanation of the "thick database" approach and its benefits

Background

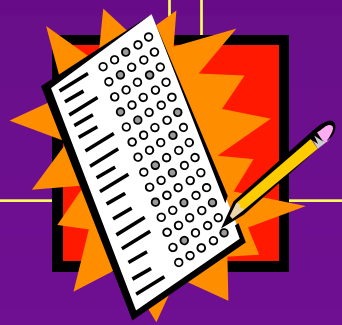
- Fusion technology stack is large and complex.
- Hard to make the transition into the J2EE environment.
- Host of different tools, programming languages, architectures, and technologies
- Projects often have the illusion of progress.
- Building functioning, scalable production software often becomes an impossible task.



Survey

- Non-Oracle DBMS
- Non-J2EE Application Server
- Apps user
  - PeopleSoft
  - JD Edwards
  - Siebel
  - eBusiness
  - None/other
- Web technology
  - J2EE
  - .Net
  - Other

- J2EE IDE
  - JDeveloper
  - Eclipse
  - Other
- J2EE persistence
  - ADF
  - EJB
  - EJB3
  - TopLink
  - Hibernate



Oracle Architecture

- First-rate Service Oriented Architecture (SOA)-centric environment.
- Built from an OO developer's perspective:
  - Lacks much of the vision that would make Designer users comfortable
  - "Not-so-subtle" encouragement to place business rules enforcement in the middle tier, coded as Java
  - Can be used to articulate data-centric complex business processes, using portions of the architecture
  - Business Process Execution Language (BPEL)
    - Can lead to applications with poor performance because of the number of round trips needed between the middle tier and the database.

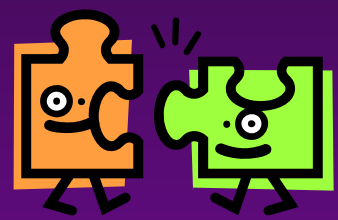
Fusion: What is it?

- In addition to e-Business, Oracle has purchased:
  - PeopleSoft (previously bought JD Edwards)
  - Siebel
  - Retek
  - AND…
    - Agile, Bharosa, Bridgestream, Netsure Telecom, TimesTen, Context Media, G-Log, Oblix, TripleHop, ProfitLogic, i-flex, Innobase, Thor Technologies, TempoSoft, OctetString, 360Commerce, Sleepycat, HotSip, Portal Software, Demantra, Net4Call, Telephony@Work, Signma Dynamics, Sunopsis, MetaSolv Software, Stellent, Hyperion, AppForge, SPL WorldGroup, Tangosol, LODESTAR
- Collectively > 200,000 database tables
- 500,000 million lines of code

Now what?

Oracle Fusion

- Will be based on the e-Business data model
- Features of other packages will be migrated into e-Business.
- Migration path from PeopleSoft, JD Edwards, Siebel
  - Impossible to automate
  - Very expensive
  - Ultimately essential



- Oracle cannot maintain all product stacks indefinitely.
- Fusion – V1 release scheduled for 2008
  - Will include the next major release of the e-Business suite using Fusion Middleware

Fusion Middleware Definition

Marketing term for "All products under development management"

Includes lots of stuff you should not care about at ALL!!

- Fusion
  - OAS
  - JDeveloper
  - Developer
    - Forms
    - Reports
  - Designer
  - XML Publisher
  - BPEL
  - BAM
  - Business Rules Engine

- Non-Fusion
  - Application Express
  - PL/SQL
  - SQL
- Recently everything related to development is Fusion Middleware
  - TopLink/Swing Integration
  - EJB3

Two Goals of Fusion Middleware

1. Support Oracle Fusion
   - Clear development path
   - Tactical focus
   - Strategic support
   - HAS to work
   - Limited scope

2. Support all J2EE development
   - Market driven
   - Lots of pieces
   - Speculative
   - Ill-defined scope

Pieces in support of Fusion are safe.

Pieces in support of marketing are market-driven.

Getting there: Fusion

- Count on a significant conversion effort sometime within the next 5 years.
- New modules should be e-Business
- Move to Oracle DBMS
  - Server-side PL/SQL
  - Oracle Business Rules engine is in the DBMS.
- Move to Oracle Application Server
  - Probably make life much easier

ORACLE FUSION
Energy for Your Business Now

"Fusion Development Technology" (FDT)

- Not an Oracle term (but it should be)
  - Subset of Fusion Middleware
- The technology used in Oracle Fusion
- For the first time in Oracle's history, development is THE critical success factor.
- At Collaborate '06, Charles Phillips' keynote was "Fusion."
  - He never even mentioned the DBMS.
- Oracle is betting the farm on FDT.
- FDT is already good, and has all of the resources that it needs to become great.
- Will have a blank check for years to come
- This is what you really need to know.

Fusion Development Technology Parts

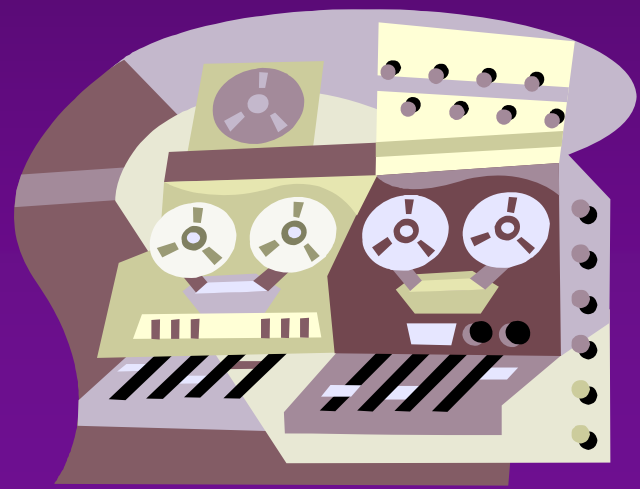
- OAS
  - J2EE application server
  - First-rate product
  - Mature
- Application Development Framework – Business Components (ADF BC)
  - Persistence layer
  - First-rate product
  - Recently revamped
- ADF Faces
  - Next generation UIX
  - Somewhat proprietary
  - Feels “new”
  - Hard to go beyond framework
- BPEL
  - Recent addition
  - Hot standard for inter-system process
  - Not sure where it fits
- Oracle Business Rules
  - No idea what to do with this
- DBMS, PL/SQL, SQL

Oracle Application Server (OAS)

- J2EE application server
- Does not play well with MS Application Server
  - No application server tech stacks interact well.
- Fusion will support other J2EE application servers.
- JDeveloper-to-OAS has single button deployment.
  - Deploying to other J2EE application servers is annoying.
- Your life will be MUCH easier with OAS.
  - Especially if doing custom deployment

Getting There: The Oracle Application Server

- Not such a big deal – can be avoided
  - Unless you are using MS Application Server
- Better integration than other application servers
- Lowest TCO
  - No finger pointing
  - Lower deployment costs



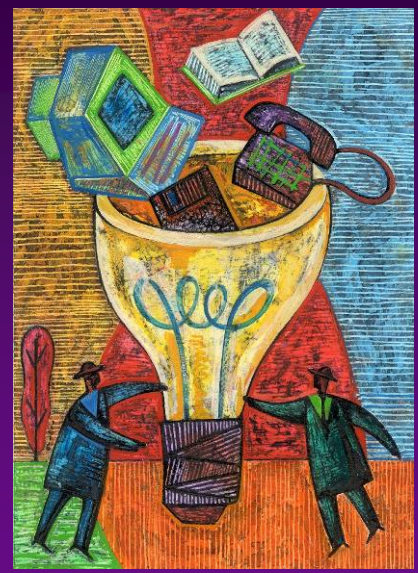
ADF BC

- Persistence interface for Fusion
- Oracle alternatives
  - TopLink
- Non-Oracle alternatives
  - Hibernate – open source
  - EJB
  - EJB3 – supported in JDeveloper
- Very high-quality
- Proprietary framework
- Very little penetration outside of Oracle

DULCIAN INC.

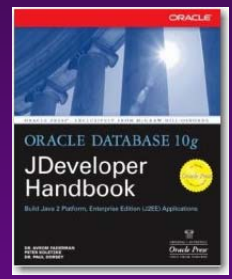
Why use ADF BC?

- Lower probability of project failure
- Very rich product
- Very mature
  - BC4J – V1 released in 2001
  - Rewritten several times
- Fusion will use it.
- Leading causes of J2EE project failure are hand-written persistence interfaces

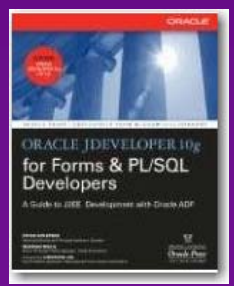


Getting There: ADF BC

- Start now
- High learning curve
- Easy to misuse
- *Oracle JDeveloper 10g Handbook*
  - (Roy-Faderman, Koletzke, Dorsey)



- *Oracle JDeveloper 10g for Forms & PL/SQL Developers*
  - (Koletzke, Mills)



BPEL
(Business Process Execution Language)

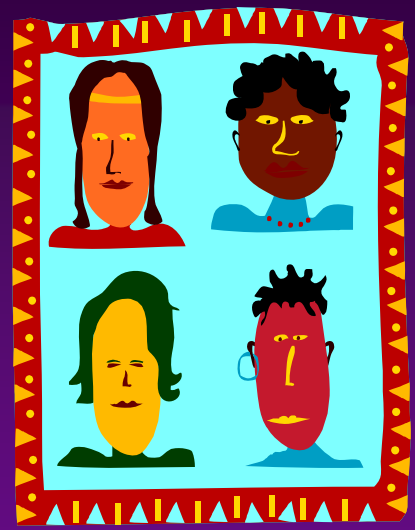
- Emerging standard
- Oracle implementation is very nice.
- Middle tier process flow language
- SOA inspired
  - Makes great sense for inter-system flows
  - Makes no sense for complex, local process flow

Getting There: BPEL

- Wait and see.
- How will Oracle really use this?
  - SOA: You may not even need it unless you are big.
  - Process Flow: You have some time.

ADF Faces

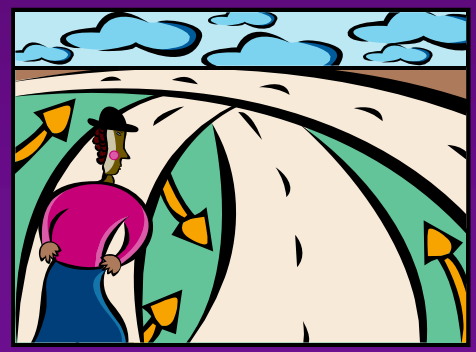
- Rich (sort of) user interface
- Standards-based
  - JavaServer Faces
  - Proprietary extension of Faces
- Next generation UIX
- Not really mature
  - Some quirks
  - Evolving fast
- Just another tag library
- Hard to extend
  - WYGIWYG (“What you get is what you get”)
- Still evolving



DULCIAN INC.™

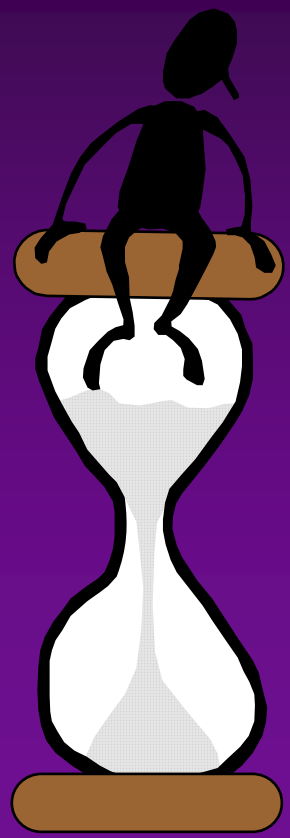
Getting There: Faces

- Start now
- Long learning curve
- Use with ADF
- Build a small project (or 3)



Getting There: Oracle Business Rules

- Wait and see.
- Not sure where this fits
- Focus resources elsewhere for now.



Non-Apps – Why should I care?

- Since I'm not an Oracle Applications customer, why should I care?
- This will be the best development platform on the planet.
  - Great Oracle integration
  - ADF BC is too good to ignore.
  - Fusion will be a force in the industry and dominant within the Oracle development community.

Fusion Middleware - Conclusions

- A great (or will be soon) development environment
- Still evolving - all parts are not totally civilized.
- Seems weak for architects (but I am biased)
- Too good and big to ignore
- Will be the standard for all Oracle Applications (eBusiness, PeopleSoft, JD Edwards)
- Still evolving, so use “thick database” approach

“Thick Database” Defined (1)

- Micro-Service-Oriented-Architecture (M-SOA) approach
- Division between the database and user interface (UI) portions.
- Two key features involved in "thick database thinking":
  - Nothing in the UI ever directly interacts with a database table. All interaction is accomplished through database views or APIs.
  - Nearly all application behavior (including screen navigation) is handled in the database.
- Thick database does not simply mean stuffing everything into the database and hoping for the best.

“Thick Database” defined (2)

- Creating a thick database makes your application UI technology-independent.
  - Creates reusable, UI technology-independent views and APIs.
  - Reduces the complexity of UI development.
  - Database provides needed objects.
  - Reduces the burden on the UI developer

Database vs. UI Technology Stack-Independence

Database

- Oracle will add features.
- DBMS will not internally refactor.
- Existing stack “works.”
- Huge DBA learning curve
- Huge cost of switching

UI Technology Stack

- Java EE or .Net?
  - AppEx
  - FLEX
- All environments change
  - Redesign assured
- Every year BRIM® has been rebuilt.

Benefit 1. Better Performance

- Improved overall throughput
- Caused by combined effect of:
  - Fewer roundtrips
  - Less network traffic
  - Better database access
- Test: Average improvement in performance?
  - a) 10%
  - b) 100%
  - c) 10x
  - d) 100x
  - e) 500x

Answer: c) 10x

Benefit 2. Fewer Round Trips

- Requires many fewer round trips from the application server to the database.
- Each screen should be 1-3 round trips
- Test: OO developers can write screens that require this many database round trips:
  - a) dozens
  - b) hundreds
  - c) thousands
  - d) millions

Answer: I have seen a, b and c.
The record was 6000 roundtrips.

Benefit 2. Less Code Required

- Less PL/SQL code is needed to perform data centric operations than Java.
- PL/SQL has more data tricks.
- Database-intensive code will always be more efficiently written in the database.
- Test: Average reduction in the amount of code needed is:
  - a) 10%
  - b) 25%
  - c) 50%
  - d) 90%

Answer: c) 50%

Benefit 3. Less Development Time Needed

- Less code means less coding time.
- Simpler architecture
  - Separate user interface and logic
  - Building two smaller applications is easier than building one large one.
- UI is trivial.
  - Can be shown to users right away.
  - Faster feedback to the development team
  - Helps to identify design errors much earlier in the process
- Test: Using a thick database approach can reduce development time by
  - a) 10%
  - b) 33%
  - c) 50%
  - d) 66%

Answer: d) 66%

Benefit 4. Easier to Maintain

- Application being built is divided into two parts
  - Each has less code to maintain.
- Application is clearly partitioned.
  - When a business rule changes, only need to look through half of the code to find it.
- As the number of lines of code in an application doubles, the complexity increases by a factor of four.

Benefit 5. Easier to Refactor

- UI technology stack changes are common.
- .Net Java EE battle rages on.
- Web architecture is more volatile than the database platform.

- Defense against the chaos of a rapidly evolving standard
- Test: What is the probability that your web UI standards will be the same in 18 months?

Answer 0%

Benefit 6. Better Use of Different Talent Levels

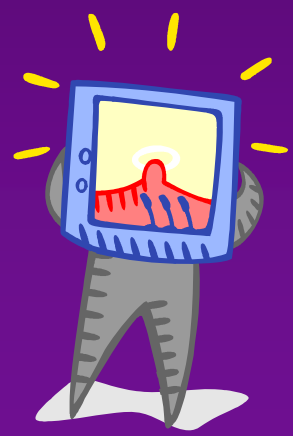
- With minimal additional training, skilled SQL and PL/SQL developers can help build web applications with no web skills whatsoever.
- If sophisticated UI developers are available, they can focus on delivering very high quality user interfaces.

Thick Database Development Process

- Two portions of an application can be coded independently
  - Teams can work in isolation until substantive portions are working.
- First version of the UI is built within a few days
  - Use as testing environment for the database team
  - Feedback can be received from users.
- Use Agile process
  - Minimal design work done to produce a partially working system.
  - Additional functionality created in an iterative design process.

User Interface Design

- Design the application.
  - Screens are designed on paper.
  - White boards are used for page flows.
  - Real screen mock-ups are usually a waste of time.
    - A careful diagram on a piece of paper suffices for the initial UI design.
    - MS Access is also good.

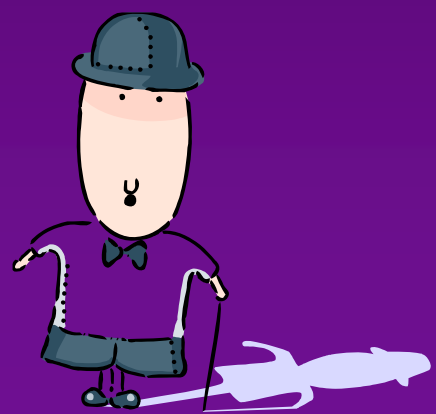


Interface Design

- Once the UI design is complete, determine:
  - What views are required
  - APIs that will be called

Interface Stubbing

- Stub out the code for the views and APIs.
  - `select <values> from dual`
  - APIs = functions that return a correct value (usually hard-coded).
- Interfaces will change as the application matures.



UI and Database Development

- UI and database development take place at the same time.
  - UI team takes the APIs and incorporates them into the application.
  - Database team makes them work.

Persistence in “Stateless Land”

- Server-side
  - Create a table and persist all global info
  - Persistent lock rows (lock_id column)
  - Pass session ID on each call
  - Worry about abandoned sessions
  - Best approach – but requires more work
- Middle tier
  - Can’t be done unless you are only using 1 application server
  - Usually persists to the database
- Client
  - Cookies
  - Pass context to database each time

Function-Based Views

- Functions can return object collections.
- An object collection can be cast to a table.
- Object collections types are supported in SQL.
- The idea:
  - Build a view over the function to hide complex procedural logic.

Underlying Types and Functions

```
type lov_oty is object (id, display_tx);
type lov_nt is table of lov_oty;

function f_getLov_nt
  (i_table_tx,i_id_tx,i_display_tx,i_order_tx)
return lov_nt is
  v_out_nt lov_nt := lov_nt();
begin
  execute immediate
     'select lov_oty('
        |i_id_tx||','||i_display_tx||')'||
    ' from '||i_table_tx||
    ' order by '||i_order_tx
   bulk collect into v_out_nt;
  return v_out_nt;
end;
```

Query the Function as a Table

- Generic value list query for any UI:
  - Uses bind variables – no significant performance impact
  - Completely dynamic – any new fields/tables/etc.

```
select id_nr, display_tx
from table(
            cast(f_getLov_nt
                    ('emp',
                     'empno',
                     'ename||''-''||job',
                     'ename')
            as lov_nt)
            )
```

Create a View

- Views placed on top of dynamic functions:
  - Completely hide the logic from the UI
  - INSTEAD-OF triggers make logic bi-directional.
  - Minor problem – There is still no way of passing parameters into the view other than by using some type of global.

```
Create or replace view v_generic_lov as
select id_nr, display_tx
from table( cast(f_getLov_nt
     (GV_pkg.f_getCurTable,
      GV_pkg.f_getPK(GV_pkg.f_getCurTable),
      GV_pkg.f_getDSP(GV_pkg.f_getCurTable),
      GV_pkg.f_getSORT(GV_pkg.f_getCurTable))
           as lov_nt)
          )
```

De-Normalized Views

- The idea:
  - Convert relational data into something that will make user interface development easier
- The solution:
  - Use a view with a set of INSTEAD-OF triggers

De-Normalized view

```
create or replace view v_customer
as
select c.cust_id,
       c.name_tx,
       a.addr_id,
       a.street_tx,
       a.state_cd,
       a.postal_cd
from customer c
left outer join address a
   on c.cust_id = a.cust_id
```

INSTEAD-OF Insert

```
create or replace trigger v_customer_ii
instead of insert on v_customer
declare
  v_cust_id customer.cust_id%rowtype;
begin
  if :new.name_tx is not null then
   insert into customer (cust_id,name_tx)
    values(object_seq.nextval,:new.name_tx)
   returning cust_id into v_cust_id;
  if :new.street_tx is not null then
   insert into address (addr_id,street_tx,
         state_cd, postal_cd, cust_id)
   values (object_seq.nextval,:new.street_tx,
    :new.state_cd,:new.postal_cd, v_cust_id);
  end if;
end;
```

A Tale of Two Systems

- 1. Internal Modification Request Tracker:
  - Built using conventional approach by an experienced Java team.
  - Earlier version built by offshore, inadequately skilled development team.
  - To create a working version of the system took about 6 months – flawed architecture
- 2. Complex order entry system
  - Built using the “thick database” approach by a team with equivalent experience.
  - Thick database approach was used from the start
    - All navigation supported using a tree on the left hand side of the screen.
    - Tree itself is built into the database.
    - All navigation logic is handled in the database.

Case Study: 2 Similar Systems - Results

	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

DULCIAN INC.

Summary

- Oracle Fusion will be based on e-Business.
  - Everyone else migrates.
- Must use:
  - Oracle DBMS, J2EE stack
- Should use:
  - OAS, ADF
- Avoid for now:
  - BPEL, Oracle Business Rules

DULCIAN INC.

Conclusions

- Moving code from the middle tier to the database had the following benefits:
  - Reduced the total amount of code
  - Reduced development time
  - Improved performance
  - Reduced network traffic
  - Reduced the complexity of the application
- Thick database approach is a viable alternative to the conventional wisdom of reducing reliance on the database.
  - Leverages existing database talent
  - Can result in dramatic improvements in performance

Share your Knowledge: Call for Articles/Presentations

- Submit articles, questions, … to

IOUG – The SELECT Journal

select@ioug.org

ODTUG – Technical Journal

pubs@odtug.com

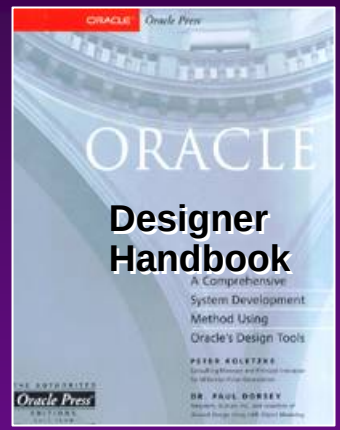
Reviewers needed

Dulcian's BRIM® Environment

- Full business rules-based development environment
- For Demo
  - Write "BRIM" on business card
- Includes:
  - Working Use Case system
  - "Application" and "Validation Rules" Engines

Contact Information

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



Available now!
Oracle PL/SQL for Dummies