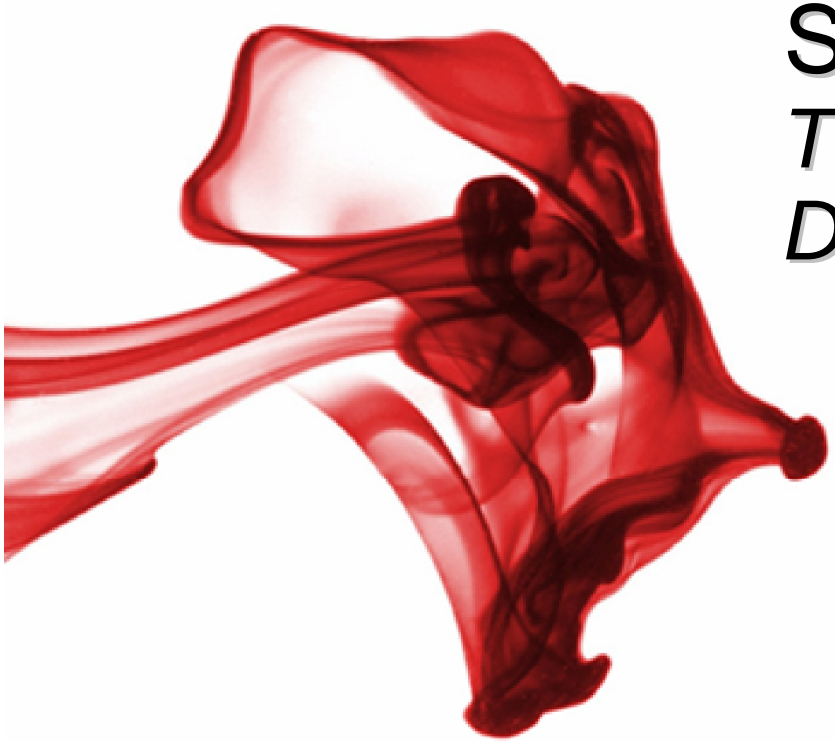


Data Delivery in a Service-Oriented World: *The BEA AquaLogic Data Services Platform*

Michael Carey

BEA Systems
www.bea.com

bea
Think liquid.

Agenda

Why data services?

Building declarative data services

Query processing in ALDSP

Updating data in ALDSP

Work in progress at BEA

Brief demo (optional)

Summary and Q&A

bea Think liquid.

Agenda

Why data services?

Building declarative data services

Query processing in ALDSP

Updating data in ALDSP

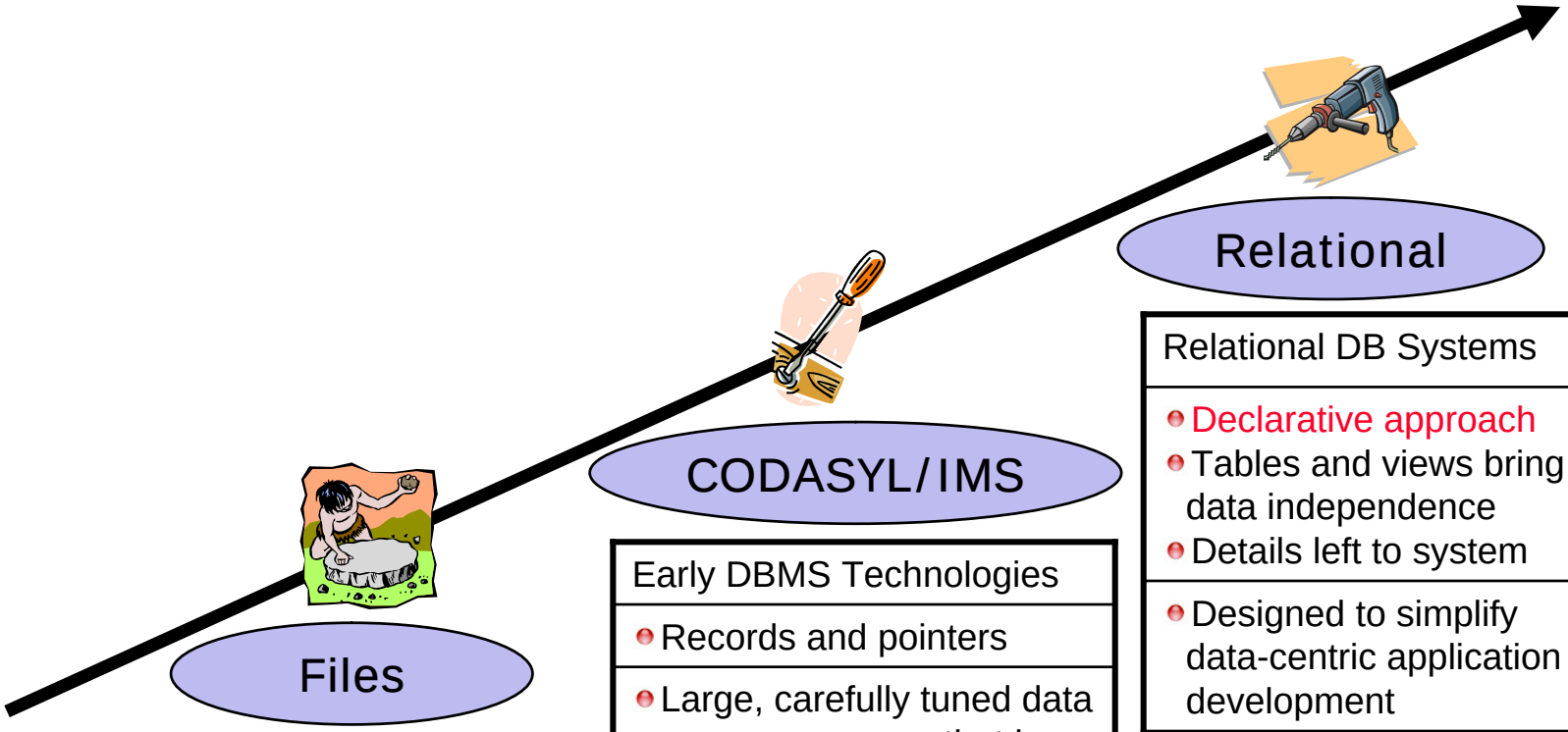
Work in progress at BEA

Brief demo (optional)

Summary and Q&A

bea Think liquid.

Evolution of Database Systems



Files

Manual Coding
• Byte or record streams
• Majority of application development effort goes toward building and maintaining data access logic

CODASYL/IMS

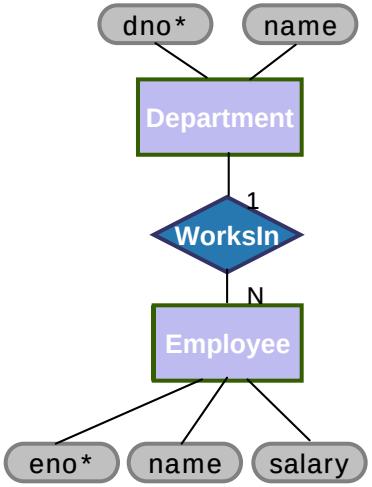
Early DBMS Technologies
• Records and pointers
• Large, carefully tuned data access programs that have dependencies on physical access paths, indexes, etc.

Relational

Relational DB Systems
• Declarative approach • Tables and views bring data independence • Details left to system
• Designed to simplify data-centric application development

bea Think liquid.

Relational Application Development



Department

dno	name
10	Toy
20	Shoe

Employee

eno	name	salary	dept
1	Lou	10000000	10
7	Laura	150000	20
22	Mike	80000	20

```
...
stmt = dbconn.prepareStatement (
   "select E.name, E.salary, D.no
    from Employee E, Department D
    where E.salary < 100000
       and D.name = ?
       and E.dept = D.dno"
);
...
```


Data Is *Everywhere* Now

- Perhaps relational databases made things too easy?
  - Departmental vs. inter-galactic centralized databases
- Databases come in many flavors
  - Relational: Oracle, DB2(s), SQL Server, MySQL, …
  - Hangers-on: IMS, IDMS, VSAM, …
- Not all data is SQL-accessible
  - Packaged apps: SAP, PeopleSoft, Siebel, Oracle, SalesForce, …
  - Custom “homegrown” apps
  - Files of various shapes and sizes
  - And the list goes on…

Painful to Develop Applications

- No one “single view of X” for any X
  - What data do I have about X?
  - How do I stitch together the info I need?
  - What else is X related to?
- No uniformity (model or language)
  - Data about X is stored in many *different formats*
  - Accessing or updating X involves many *different APIs*
  - *Manual coding* of “distributed query plans”
- No reuse of artifacts
  - Different access criteria and/or returned data → different access plans
  - And how would anyone even begin to find them? (No model)

The SOA Movement

- Service-Oriented Architecture (SOA)
  - Loosely-coupled interfaces (e.g., Web service contracts)
  - Each subsystem is a component with a service API
  - Create new assets by integrating & composing your existing assets!
- We're closer to dealing with heterogeneity
  - Services all have XML Web service foundations
  - Hide custom logic (e.g., data access and/or integration)
- Fine …. but what about my data…?
  - What are my business entities and how are they interrelated?
  - How can I find them, and what can I do to them?

→ SOA *what?*

bea Think liquid.

Agenda

Why data services?

Building declarative data services

Query processing in ALDSP

Updating data in ALDSP

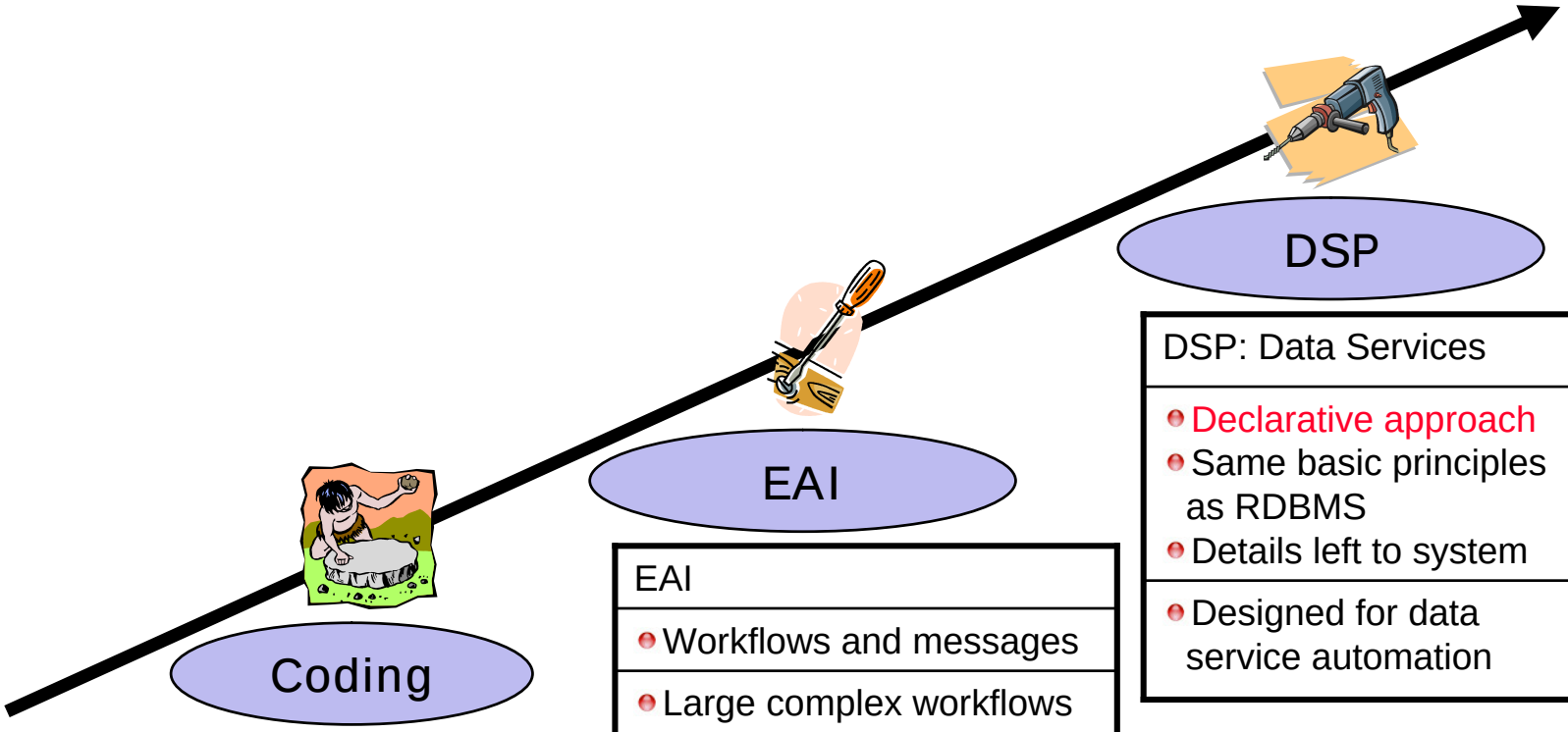
Work in progress at BEA

Brief demo (optional)

Summary and Q&A

bea Think liquid.

Evolution of SOA Data Access



Manual Coding
Java or C Programming
Majority of application development effort goes toward building and maintaining data access logic

EAI
Workflows and messages
Large complex workflows that are cumbersome to build and maintain

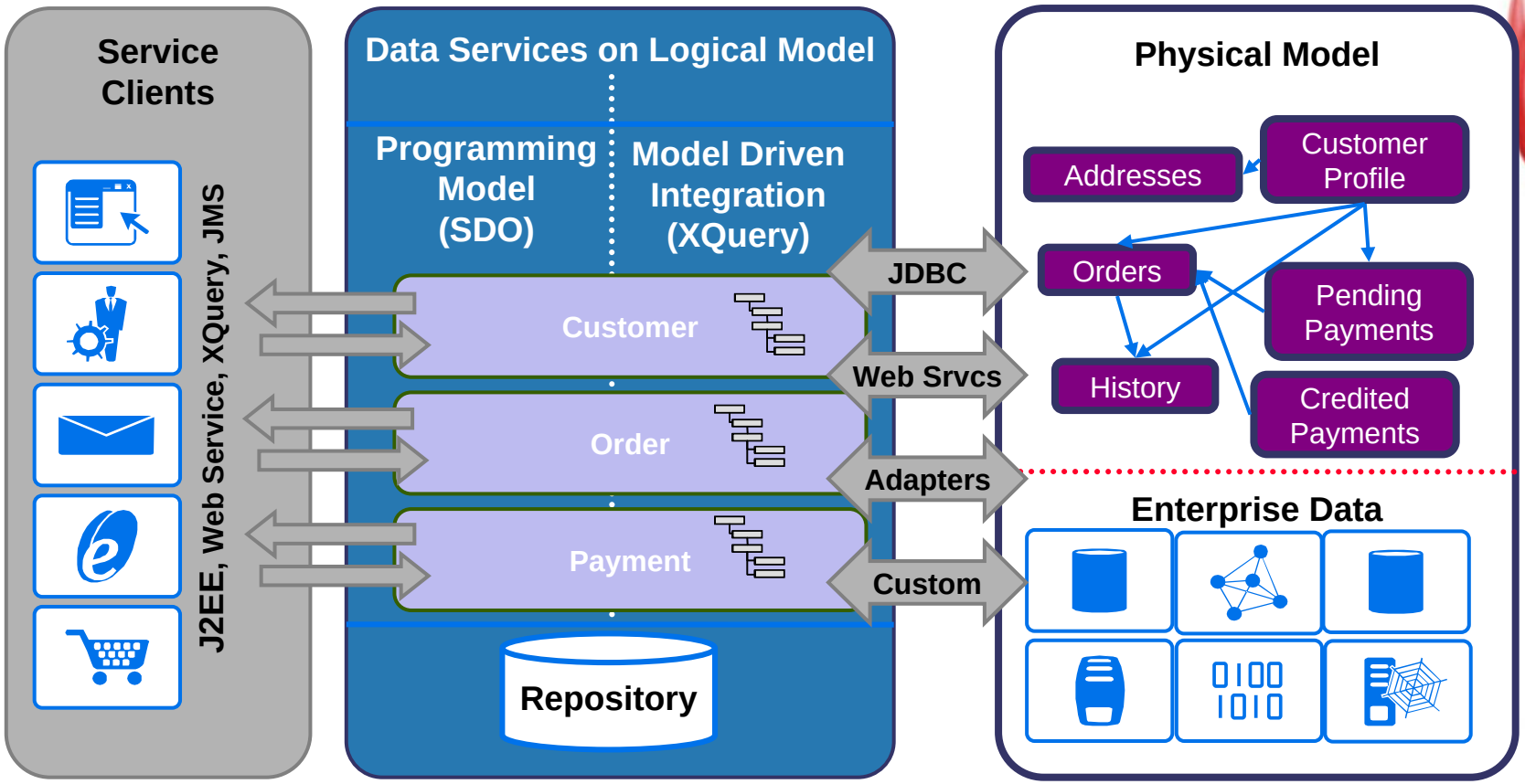
DSP: Data Services
Declarative approach Same basic principles as RDBMS Details left to system
Designed for data service automation

bea Think liquid.

Declarative Integration via XQuery

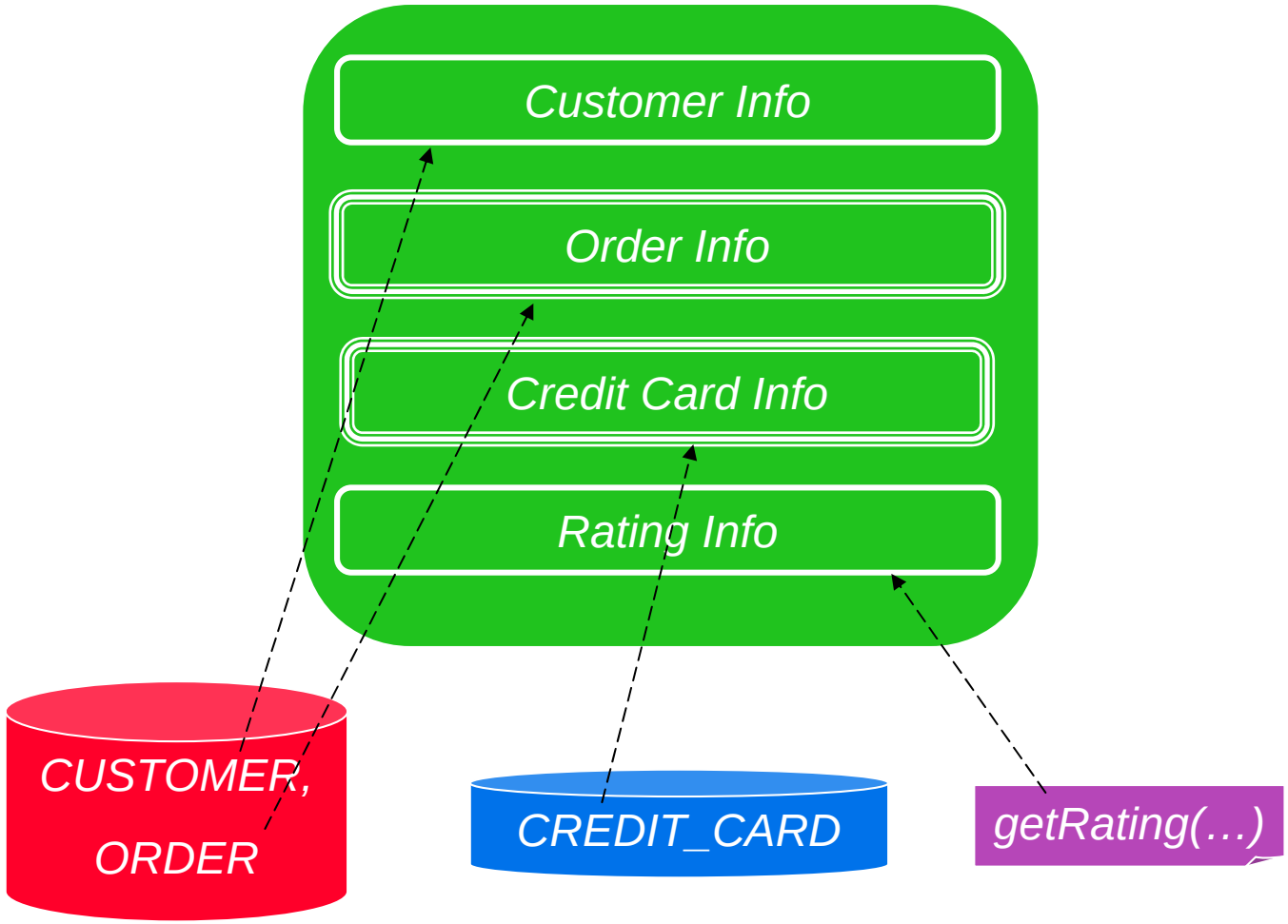
Requirements	Standards
A standard for data format and data interchange	XML
A standard for describing and modeling data	XML Schema
A standard for interfacing into applications	Web Services
A standard for querying both relational and non-relational data	XQuery
A standard Java programming model (read + write)	SDO (Service Data Objects)
A standard for publishing available services	Web Services

Data Services a la AquaLogic DSP

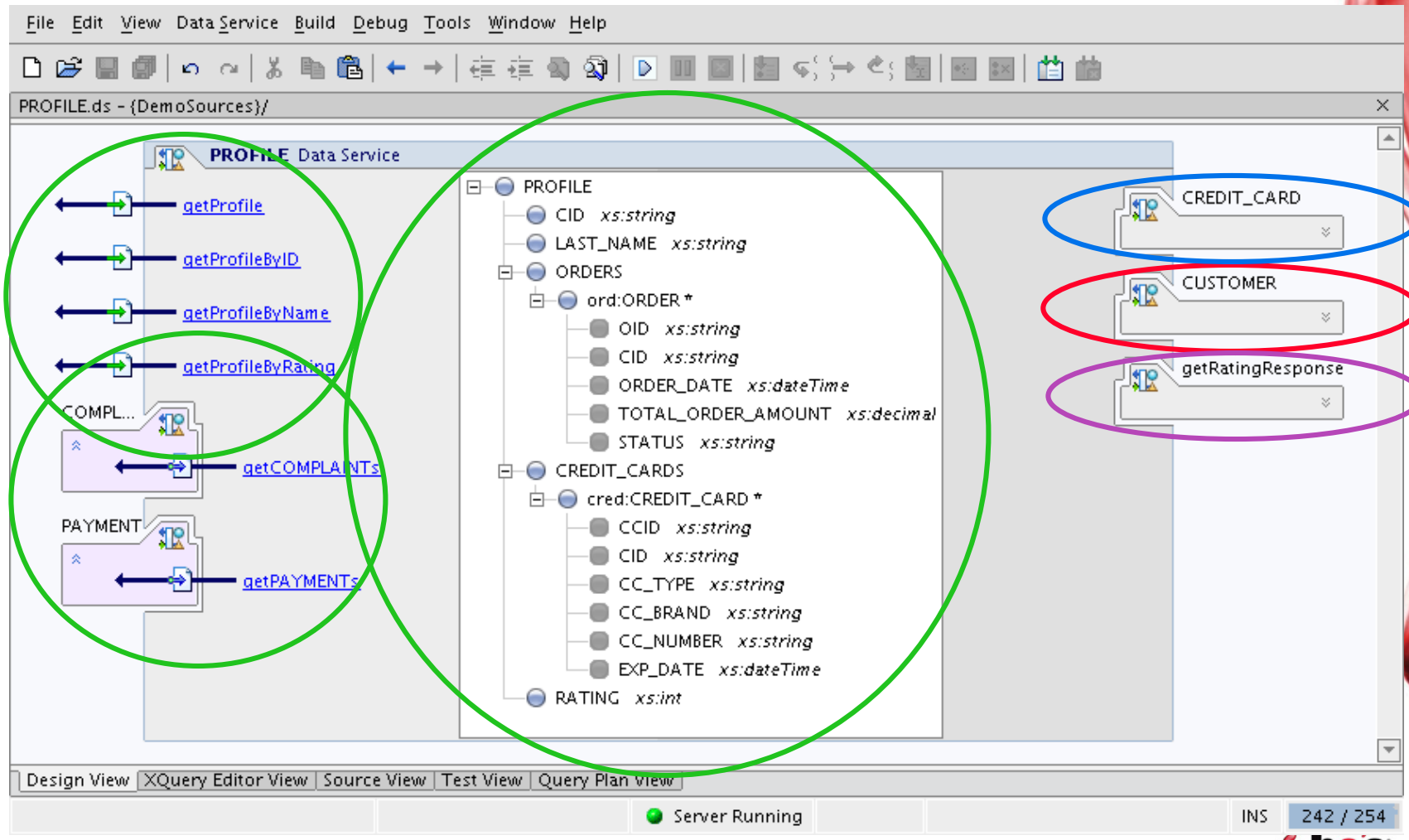


- Logical models capture data access and integration complexity **once**
- **Same** data model, programming model, and API for all enterprise data

Ex: Customer Profile Data Service

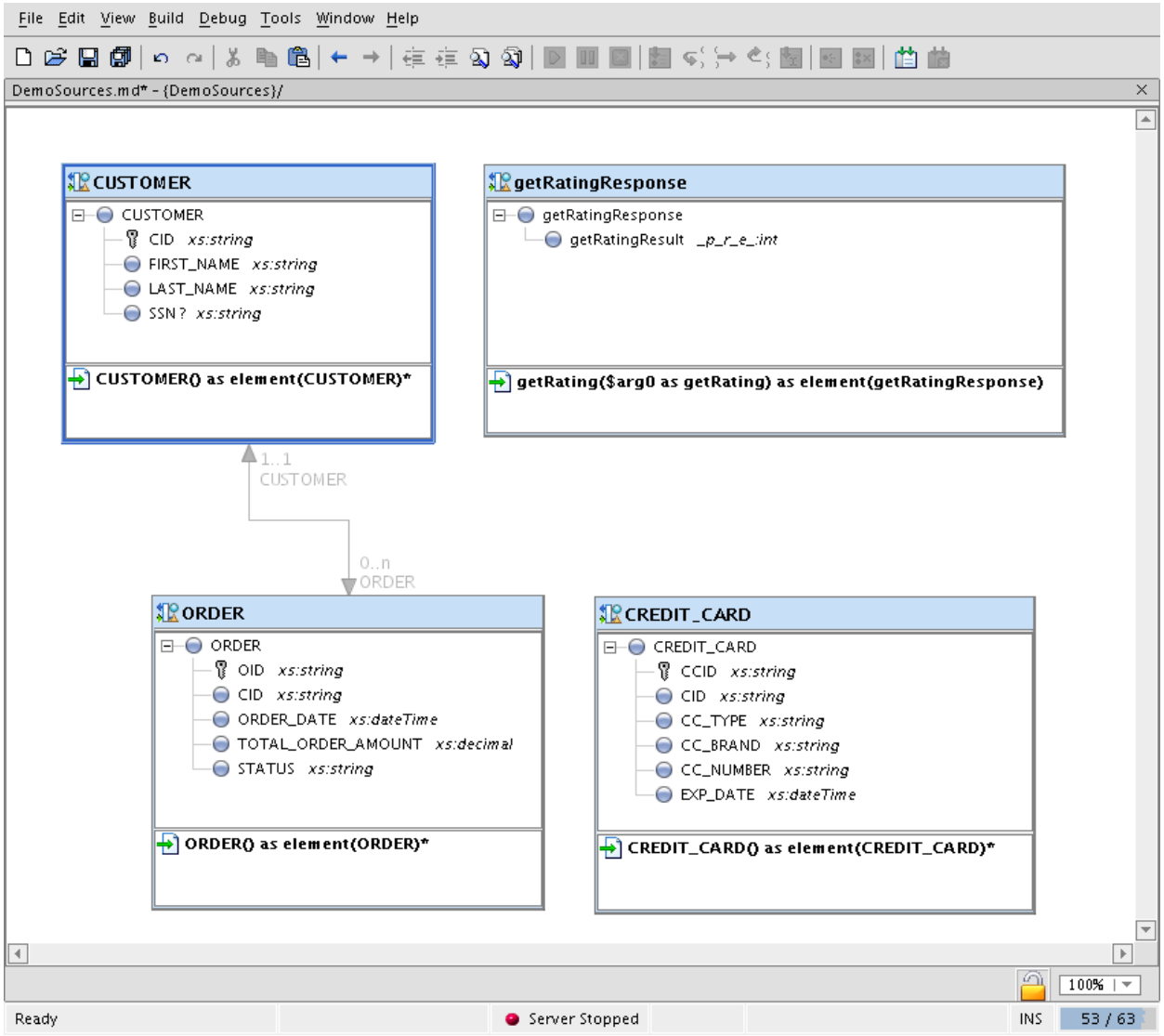


Data Service – Design View



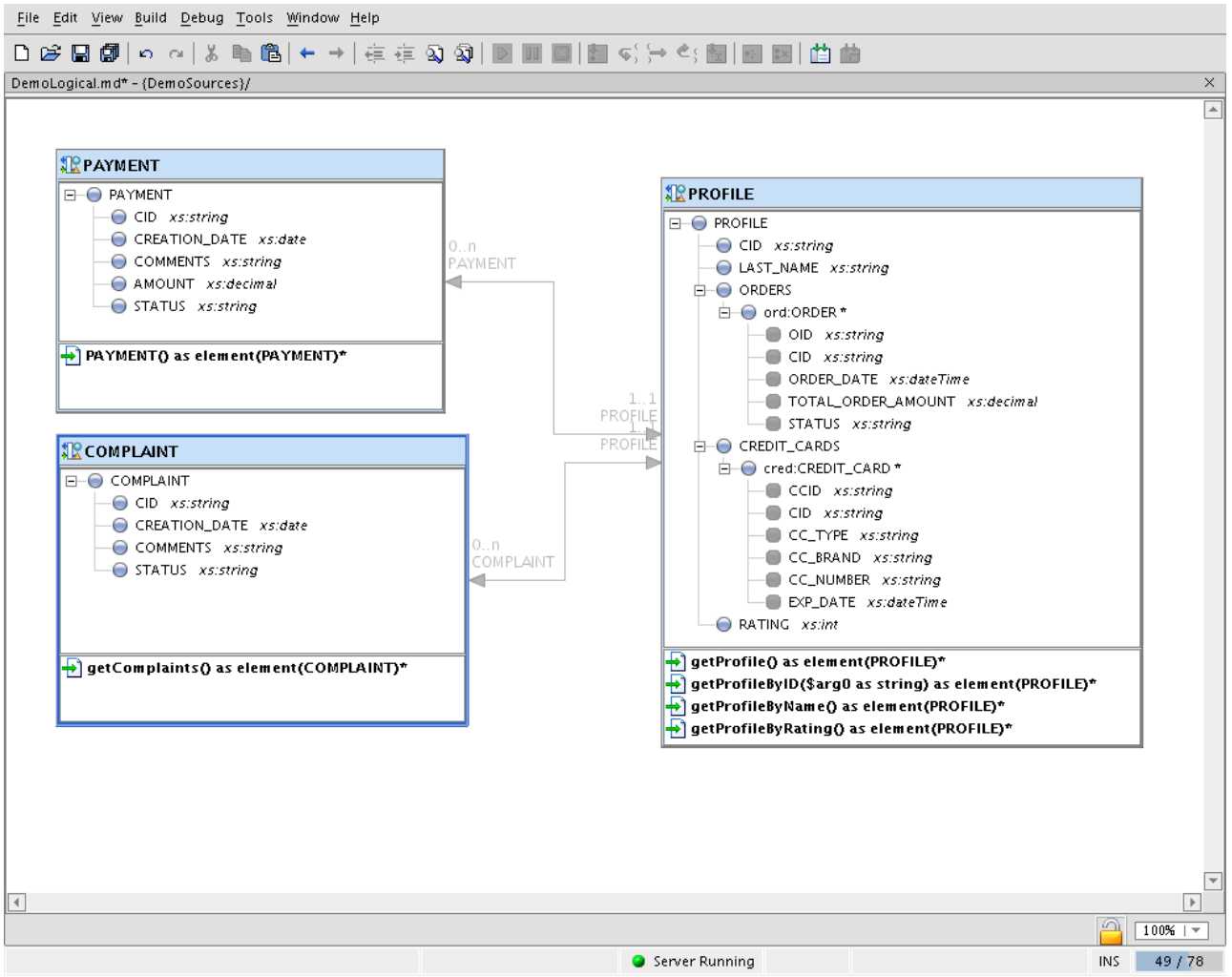
bea Think liquid.

Service Model View (Physical Services)



bea Think liquid.

Service Model View (Logical Services)



 |

Data Service – “Get All” Read Method

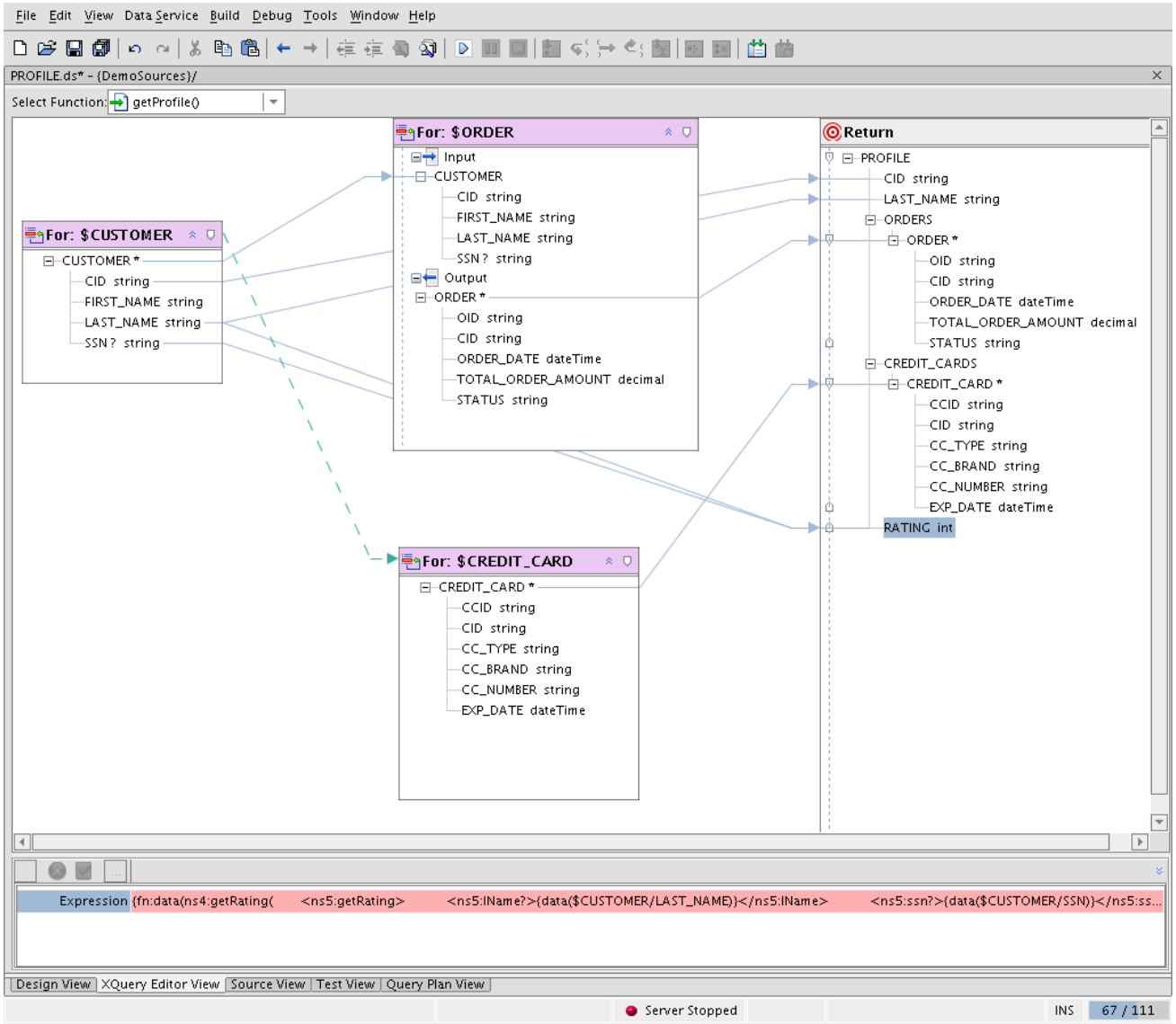
```
(::pragma function ... kind="read" ...::)

declare function tns:getProfile() as element(ns0:PROFILE)*
{
  for $CUSTOMER in db1:CUSTOMER()
  return
   <tns:PROFILE>
      <CID>{ fn:data($CUSTOMER/CID) }</CID>
      <LAST_NAME>{ fn:data($CUSTOMER/LAST_NAME) }</LAST_NAME>
      <ORDERS>{ db1:getORDER($CUSTOMER) }</ORDERS>
      <CREDIT_CARDS>{
         db2:CREDIT_CARD()[CID eq $CUSTOMER/CID]
      }</CREDIT_CARDS>
      <RATING>{
         fn:data(ws1:getRating(
            <ns5:getRating>
               <ns5:lName>{ data($CUSTOMER/LAST_NAME) }</ns5:lName>
               <ns5:ssn>{ data($CUSTOMER/SSN) }</ns5:ssn>
            </ns5:getRating>
         )
      }</RATING>
   </tns:PROFILE>
};
```


Data Service – Read & Navigate Methods

```
(::pragma function ... kind="read" ...::)

declare function tns:getProfileByID($id as xs:string)
   as element(ns0:PROFILE)*
{
   tns:getProfile()[CID eq $id]
};

...
```

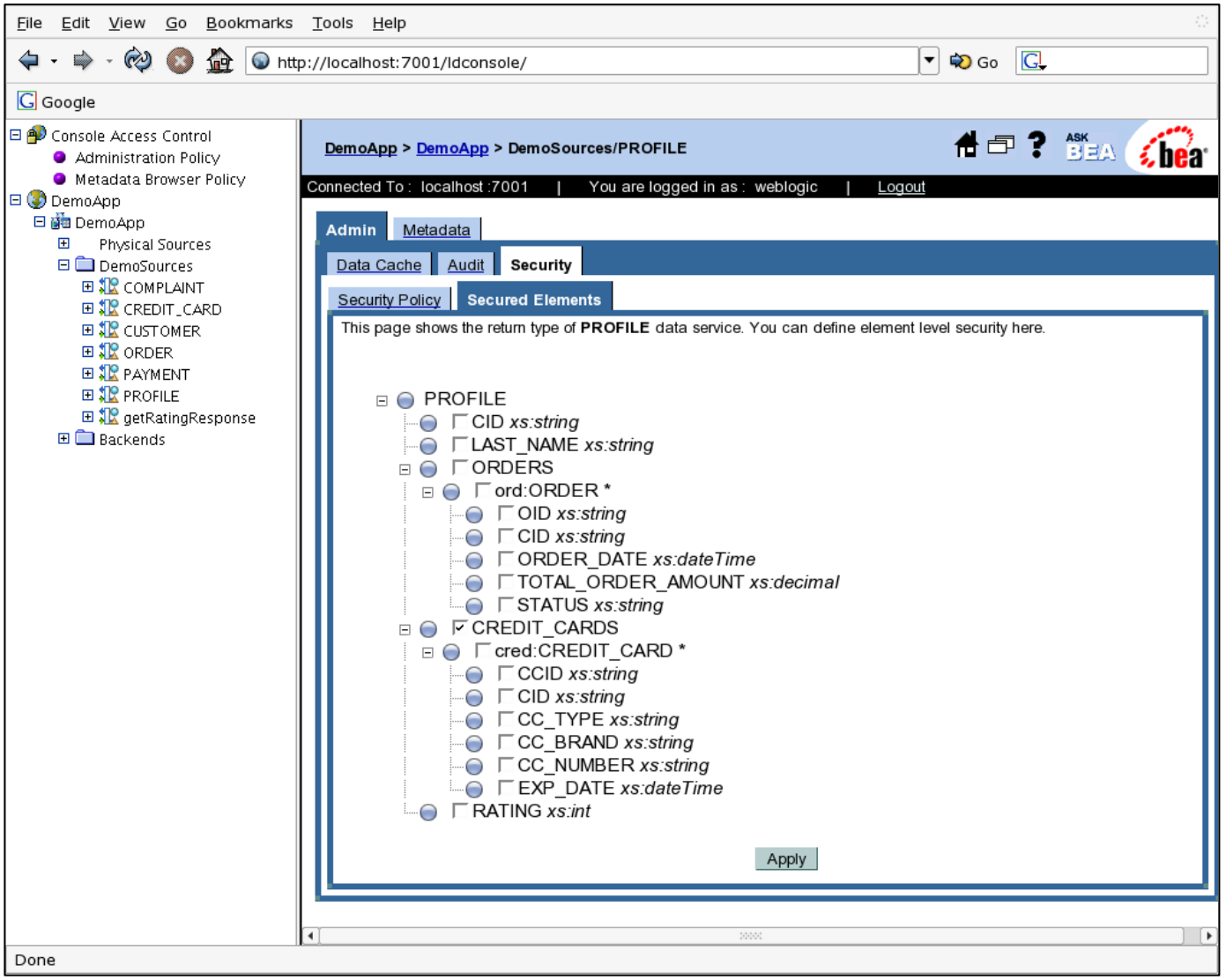
```
(::pragma function ... kind="navigate" ...::)

declare function tns:getCOMPLAINTs($arg as element(ns0:PROFILE))
   as element(ns8:COMPLAINT)*
{
   db3:COMPLAINT()[CID eq $arg/CID]
};

...
```


Graphical Query Editor



bea Think liquid.

Fine-Grained Security in ALDSP



bea Think liquid.

Agenda

Why data services?

Building declarative data services

Query processing in ALDSP

Updating data in ALDSP

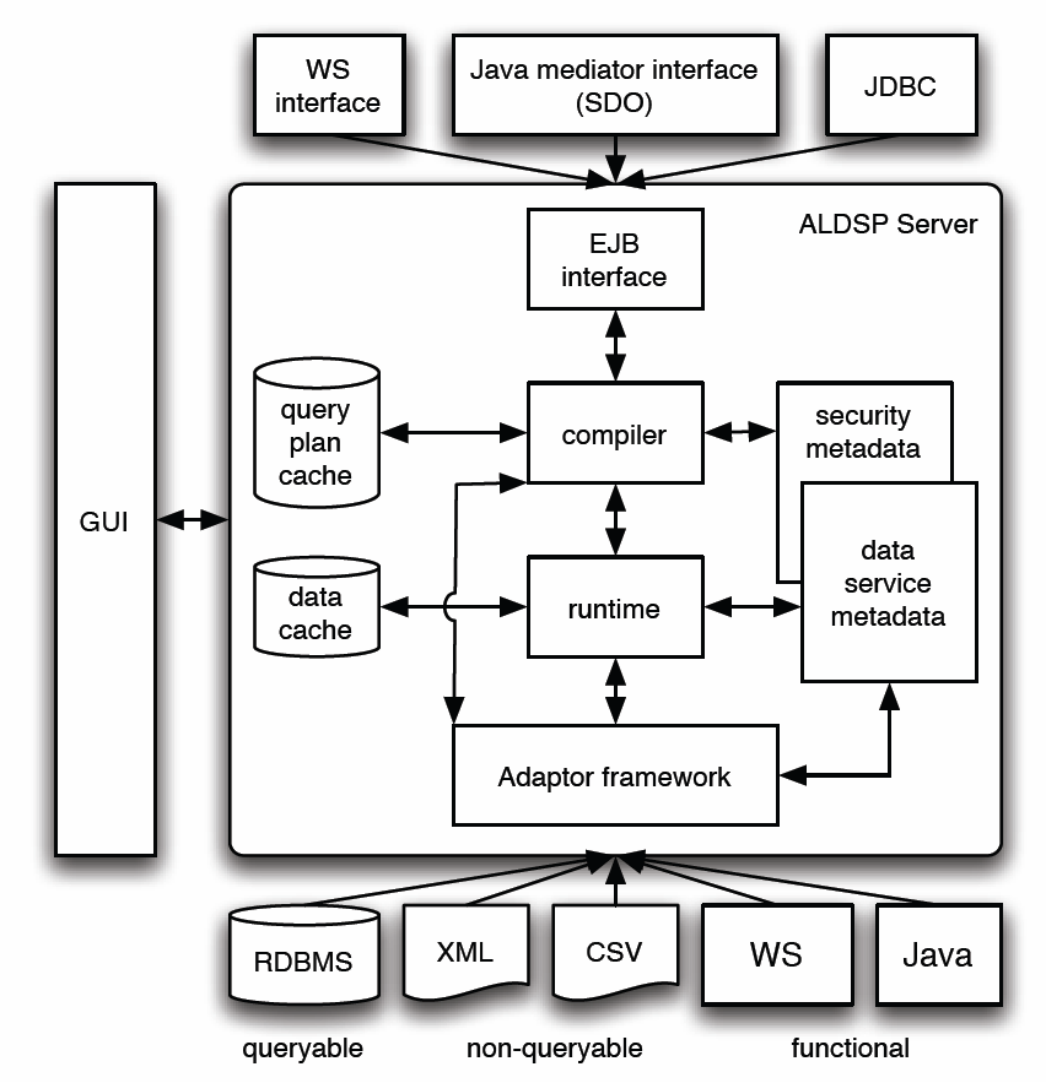
Work in progress at BEA

Brief demo (optional)

Summary and Q&A

bea Think liquid.

Architectural Overview



Query Processing in ALDSP

- Compile-time function composition
  - Similar to RDBMS view rewriting & unnesting optimizations
  - Facilitates efficient pushdown, eliminates irrelevant data sources, …
    - → It's what makes data services *reusable!!*
- Joins and related operations
  - *Goal:* Let each RDBMS do what it does best → *maximize SQL pushdown!*
  - Outerjoins, presorted grouping, sorting pushdown, function calls, …
  - PP-k joins for pipelined/distributed query processing
- Runtime system
  - Pipelined ("streaming") via XML TokenIterator model
- Other related goodies
  - Including *async(exp), failover(exp1,exp2), timeout(exp1,t,exp2)*

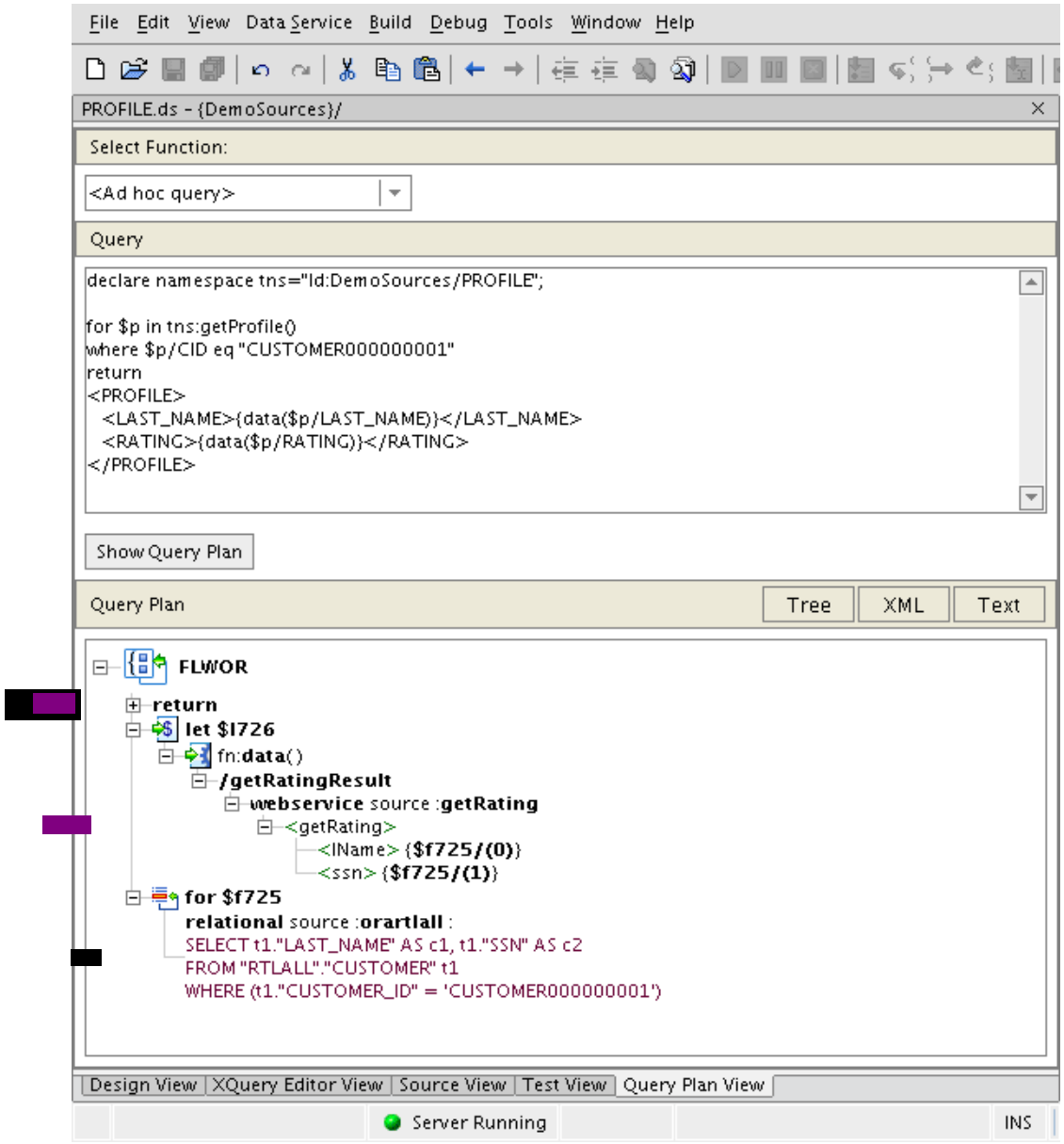
Example: “Get All” Read Method Revisited

```
(::pragma function ... kind="read" ...::)

declare function tns:getProfile() as element(ns0:PROFILE)*
{
  for $CUSTOMER in db1:CUSTOMER()
  return
   <tns:PROFILE>
      <CID>{ fn:data($CUSTOMER/CID) }</CID>
      <LAST_NAME>{ fn:data($CUSTOMER/LAST_NAME) }</LAST_NAME>
      <ORDERS>{ db1:getORDER($CUSTOMER) }</ORDERS>
      <CREDIT_CARDS>{
         db2:CREDIT_CARD()[CID eq $CUSTOMER/CID]
      }</CREDIT_CARDS>
      <RATING>{
         fn:data(ws1:getRating(
            <ns5:getRating>
               <ns5:lName>{ data($CUSTOMER/LAST_NAME) }</ns5:lName>
               <ns5:ssn>{ data($CUSTOMER/SSN) }</ns5:ssn>
            </ns5:getRating>
         )
      }</RATING>
   </tns:PROFILE>
};
```

Query Processing, Example 1 *(getProfile)*

bea Think liquid.

Query Processing, Example 2 *(query getProfile)*



bea Think liquid.

Caching in ALDSP

- Query plan cache
  - Cache recently compiled query plans, as in RDBMSs
  - Cache partially-compiled plans for views to speed query compilation
- Data service function cache
  - Favorite RDBMS can be configured as a cluster-wide data cache
  - Cache is functional, i.e., a map: *function(params)* → *results*
  - Autonomous data sources →TTL-based “consistency”
  - Turns expensive (high-latency) operations into single-record fetches, so a typical use case might be *getCreditRating(ssno)*

Agenda

Why data services?

Building declarative data services

Query processing in ALDSP

Updating data in ALDSP

Work in progress at BEA

Brief demo (optional)

Summary and Q&A

bea Think liquid.

Data Service Updates

- So far we have covered read services
  - Declaratively specified using XQuery
  - System selects efficient implementation
- Obviously need write services as well
  - Automation through lineage analysis of read services
  - Full automation possible for SQL-based data services
  - Update overrides required for Web services (non-SQL sources)
- What programming model for writes?
  - Disconnected model is highly desirable
  - Want flexible optimistic concurrency options
  - *Answer:* SDO from IBM, BEA, Oracle, SAP, and XCalia

SDO API & Change Tracking

```
//Get SDO
CustomerDoc custSDO =
    CustomerDS.getCustomerById("007");

// Make changes to SDO
custSDO.setCustName("Mike");
custSDO.setEmail("mcarey@bea.com")

//Submit SDO
CustomerDS.submit(custSDO);
```

Original SDO

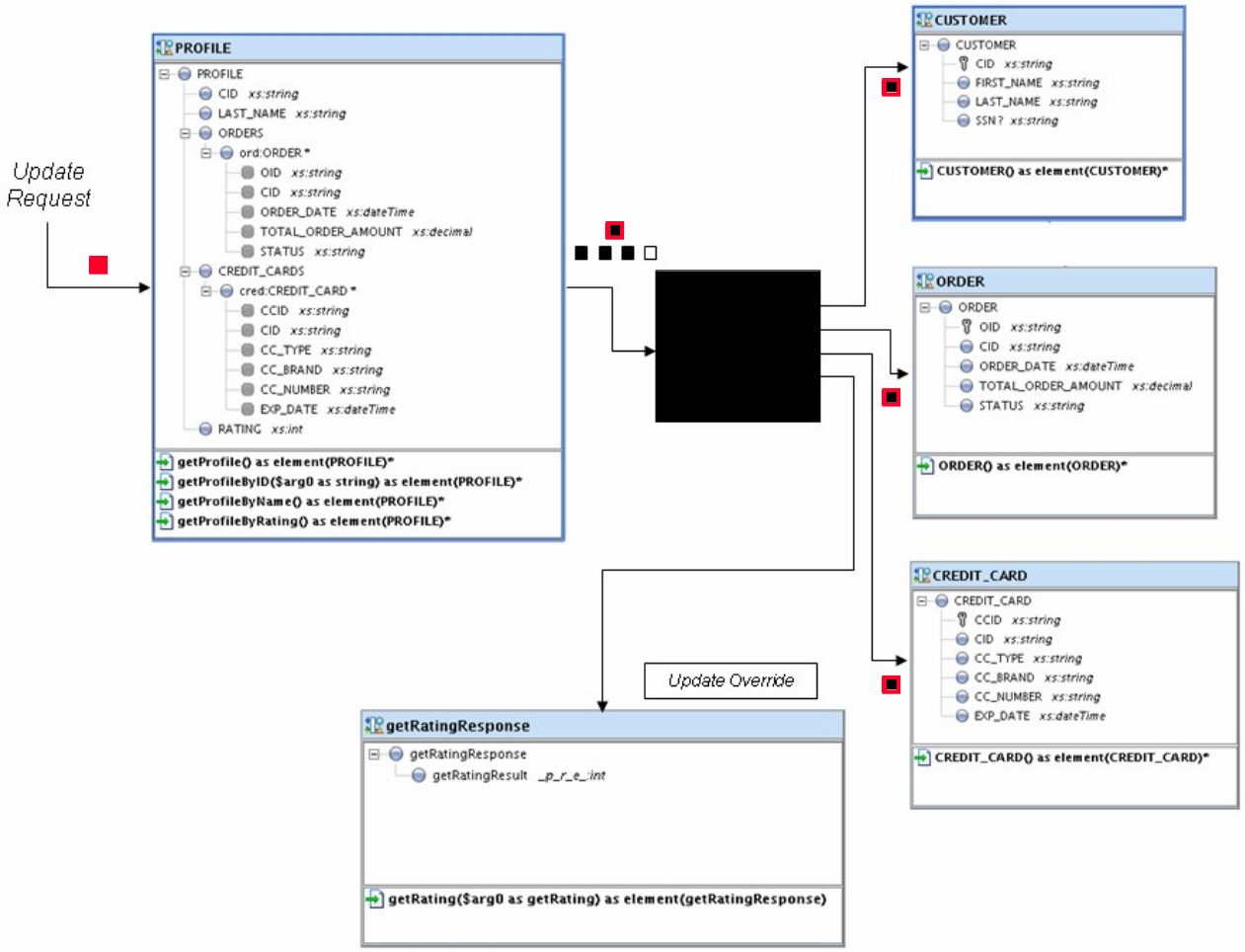
```
<CustDataGraph>
<cus:CUSTOMER xmlns:cus="ld:LiquidDataApp/CUSTOMER">
<CUSTOMER_ID>007</CUSTOMER_ID>
<CUST_NAME>Michael</CUST_NAME>
<EMAIL_ADDRESS>mikejcarey@aol.com</EMAIL_ADDRESS>
<TELEPHONE_NUMBER>408-570-8599</TELEPHONE_NUMBER>
</cus:CUSTOMER>
</CustDataGraph>
```

SDO w/ Changes

```
<CustDataGraph>
<cus:CUSTOMER xmlns:cus="ld:LiquidDataApp/CUSTOMER">
<CUSTOMER_ID>007</CUSTOMER_ID>
<CUST_NAME>Mike</CUST_NAME>
<EMAIL_ADDRESS>mcarey@bea.com</EMAIL_ADDRESS>
<TELEPHONE_NUMBER>408-570-8599</TELEPHONE_NUMBER>
</cus:CUSTOMER>
<ChangeSummary>
<CUSTOMER com:ref="/CUSTOMER">
<CUST_NAME>Michael</CUST_NAME>
<EMAIL_ADDRESS>mikejcarey@aol.com</EMAIL_ADDRESS>
</CUSTOMER>
</ChangeSummary>
</CustDataGraph>
```


bea Think liquid.

Update Decomposition



Update Framework

- XA and non-XA sources
- Automated change decomposition
- Automatic SQL generation for RDBMS
- Update “hooks” for business validations, replacement logic, or compensation logic (e.g., via a workflow)

bea Think liquid.

Update Automation (RDBMS Sources)

- Primary key handling
  - Automated key generation using Identity or Sequence
- Foreign keys can be filled in based on context
  - Need not be projected in the child elements
  - Inferred from predicates in the designated read query
- Updates sequenced to avoid RI issues
  - Deletion of children before deletion of parent
  - Insertion of parent before inserting children

Concurrency Model (RDBMS Sources)

- Based on optimistic concurrency control
  - Before values are compared to current database values
  - *Ex:* **update** CUSTOMER **set** FIRST_NAME=?
    where CUSTOMER_ID=? **and** FIRST_NAME=?
- Comparison (consistency) options include
  - All updated fields
  - All read or updated fields
  - Designated field or fields (e.g., timestamp or version id)
- Benefits of this approach
  - Stateless and therefore scalable
  - Natural fit for Web apps and services

Agenda

Why data services?

Building declarative data services

Query processing in ALDSP

Updating data in ALDSP

Work in progress at BEA

Brief demo (optional)

Summary and Q&A

bea Think liquid.

Work in Progress (or Recently Completed)

- Native JDBC/SQL92 support – ALDSP 2.5
  - Bilingual server for efficient reporting/BI tool access
  - Limited to flat views and procedures (of course)
- Update automation – goal is for no Java coding to be needed in most cases
  - Declarative editor for modifying system's default update behavior
  - XQuery update & procedure language (XUP – related to XQueryP)
- Compensating transactions – goal is for no BPEL (or JPD) coding to be needed in most cases either
  - Like current SDO updates, but with non-XA sources (via Sagas)
  - DS architect will provide undo/did-I-do operations (and CRUD)

Agenda

Why data services?

Building declarative data services

Query processing in ALDSP

Updating data in ALDSP

Work in progress at BEA

Brief demo (optional)

Summary and Q&A

bea Think liquid.

Demo (*Time Permitting*)

- BEA AquaLogic Data Services Platform 2.5:

A declarative basis for data service creation & management…

bea Think liquid.

Agenda

Why data services?

Building declarative data services

Query processing in ALDSP

Updating data in ALDSP

Work in progress at BEA

Brief demo (optional)

Summary and Q&A

bea Think liquid.

Summary

- Challenges in the Brave New World
  - From databases (then) to *data services* (now!)
- Simplify data service development
  - Data-oriented modeling and design still critical
  - XQuery and XML Schema → *declarative* data services
  - Java / WS APIs + SDO → update as well as read automation
- BEA AquaLogic Data Services Platform 2.5
  - A declarative basis for designing and building data services
  - (Now bilingual for SQL-based reporting applications)
- Next steps
  - Richer, more declarative update facility (including Sagas)

For More Info

- Technical papers and online information
  - V. Borkar *et al*, “XML Data Services”, *Int’l. J. of Web Services Research*, 3(1), January–March 2006
  - M. Carey *et al*, “Data Delivery in a Service-Oriented World: The BEA AquaLogic Data Services Platform”, *Proc. ACM SIGMOD Conf.*, Chicago, Illinois, June 2006
  - V. Borkar *et al*, “Query Processing in the AquaLogic Data Services Platform”, *Proc. VLDB Conf.*, *Seoul, Korea, September 2006*
  - Product information: http://www.bea.com/dataservices
  - Product documentation: http://edocs.bea.com/aldsp/docs25/
- Feel free to contact me: mcarey@bea.com
- *Questions…?*

