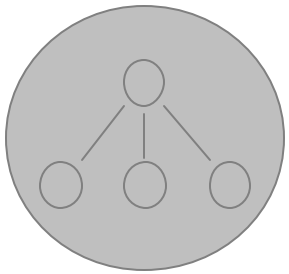


Service Data Objects (SDO)

Overview and Programming Model

Given by Greg Ackerman

(material shared from Martin Nally, Shane Claussen, and Brent Daniel)

Agenda

- IBM/BEA Public Announcement
- Introduction – what are the problems to be solved? What are SDO's goals?
- Classifying Data
- SDO Overview
- SDO Architecture
- SDO Topologies
- Some Use Cases – XML, JDBC, EJB, Web Service
- Applying SDO to a Sample Scenario
- SDO Release Status

IBM/BEA Public Announcement

Next-Generation Data Programming: Service Data Objects

A Joint Whitepaper with IBM and BEA

November 2003

Authors

John Beatty, BEA Systems
Stephen Brodsky, IBM Corp.
Martin Nally, IBM Corp.
Rahul Patel, BEA Systems

IBM home | Products & services | Support & downloads | My account

IBM developerWorks > Java technology

Specifications: Service Data Objects, WorkManager, and Ti

IBM and BEA Joint Specifications Overview

InfoWorld HOME NEWS TEST CENTI

requests from customers and joint Independent Software Vendor (ISV) partners, IBM and BEA
ls for Java™ 2 Enterprise Edition (J2EE) application servers that provide programmers with s
ons. Three specifications have been published under royalty-free terms and will be implement
he WebLogic Platform.

bjects: Simplifying the programming model for data access

SDO) is designed to simplify and unify the way in which applications handle data. Using SDO, application progra
eterogeneous data sources, including relational databases, XML data sources, Web services, and enterprise inf
oals and architecture of SDO, see the whitepaper "Next-Generation Data Programming: Service Data Objects."

NEWS

IBM, BEA propose Java standards

Companies aim to bring servers together

By Ed Scannell

IBM and BEA Systems on Tuesday disclosed they are working jointly on th
pressure by developers and users to bring their respective applications ser

SPONSOR

2003 InfoWorld 100 Awards - How much are the winning companies spending on IT initiatives? Which industries are the most innovative? Click here to download full report.

Sponsored by Qualys.

RELATED LINKS

- HP integration team leader resigns
- Application

The three new specifications, which include Service Data Objects, Work Manager for Application Servers, and Timer for Applications Servers, are all intended to increase much-needed applications portability across the applications servers of both companies.

"Some users, and certainly ISV partners, have been instrumental in showing us the light. They both have been innovating in a number of areas around Java APIs, and they have been looking for some commonality. [Users] encouraged us to get together and collaborate more closely to find some convergence," said Scott Dietzen, BEA's CTO.

siliconvalley.internet.com/news/article.php/3113361

Back to Article

Will IBM/BEA Collaboration Rile Rivals?

By Jim Wagner
November 25, 2003

IBM (Quote, Chart) and BEA Systems (Quote, Chart) are sending three J2EE specifications addressing server portability to the Java Community Process (JCP) for review.

The JCP, a standards-setting body created by Sun Microsystems (Quote, Chart) to bring consistency to the Java programming language, is made up of individual developers and businesses throughout the world.

It will decide next week whether to include the IBM/BEA specifications: Service Data Objects (SDO), Work Manager for Application Servers and Timer for Application Servers as Java Specification Requests (JSR).

CNET tech sites: | Prici

CNET NEWS.COM TECH NEWS FIRST

Enterprise software

IBM, BEA join on Java strategy

Last modified: November 25, 2003, 6:01 AM PST

By Martin LaMonica
Staff Writer, CNET News.com

IBM and BEA Systems, usually staunch rivals, are collaborating on ways to smooth out technical differences between their respective Java software lines.

The two companies on Tuesday announced that they have published specifications for three additions to their products that would make it easier for customers to run programs on their respective Java application servers. The changes are expected to be fully

BEA and IBM Collaborate to Extend Common Functionality for Customers and Business Partners on BEA WebLogic Platform and IBM WebSphere

OSE, Calif. & SOMERS, N.Y., Nov.25, 2003-- BEA and IBM today announced that
re working on three new specifications for the Java™ platform that will increase
ation portability across both companies' application server software.

specifications will benefit both customers and partner ISVs by providing more
tency across the companies' application servers – BEA WebLogic Server™ and IBM
phere. Additionally, these specifications will make it easier for developers to enable
ss applications to work across both companies' application server software.

ompanies are publishing the resulting specifications on a royalty-free basis and are
g industry feedback to ensure maximum benefits to developers and customers. The
ed specifications are also being submitted for Java Community Process (JCP)
eration.

goal of this collaborative workbetween IBM and BEA to offer our customers a simpler
ore consistent platform for J2EE development" said Rod Smith, Vice President,
ing Technologies, IBM Software Group. .

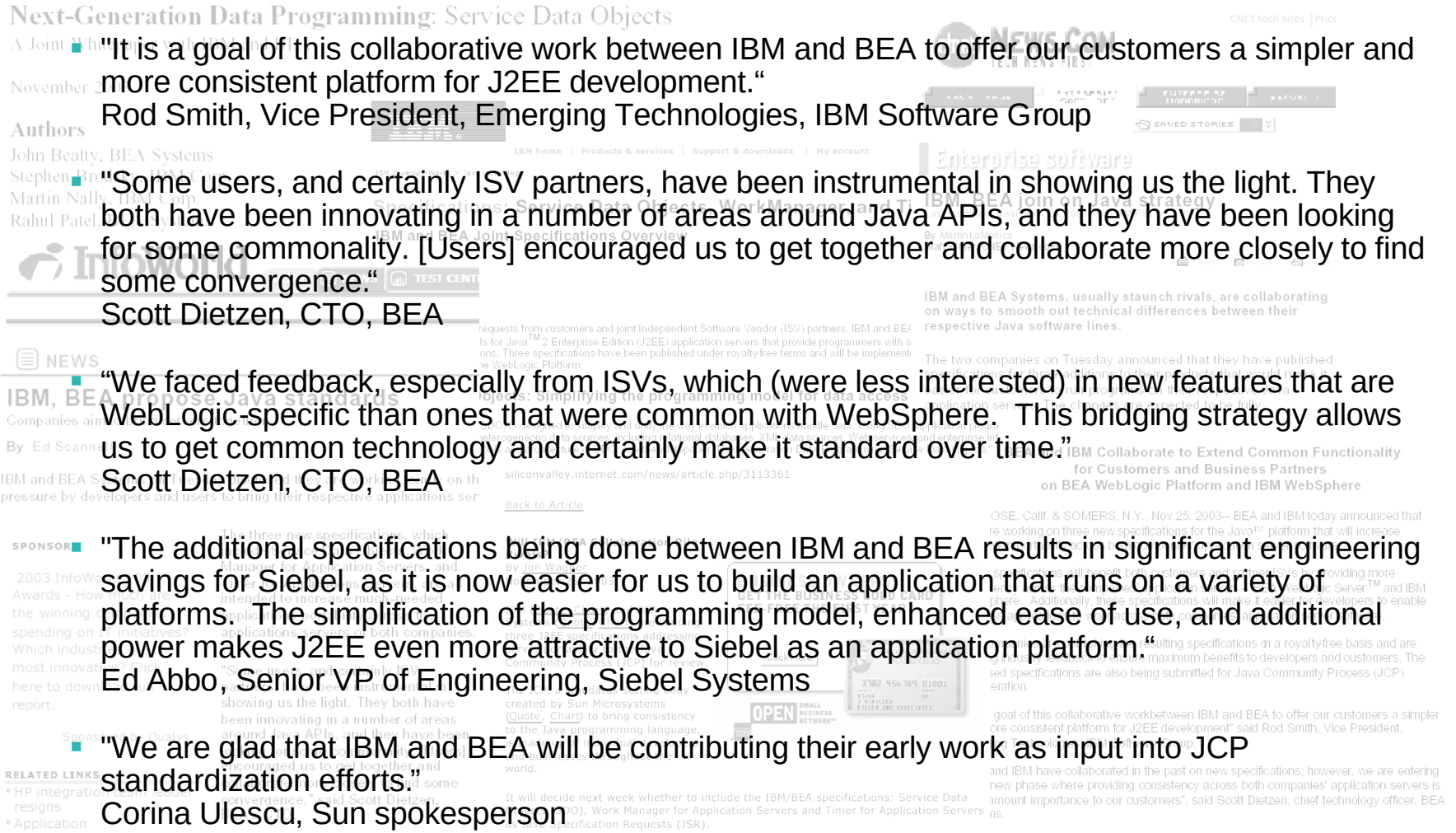
and IBM have collaborated in the past on new specifications; however, we are entering
new phase where providing consistency across both companies' application servers is
amount importance to our customers", said Scott Dietzen, chief technology officer, BEA
ns.

IBM/BEA Public Announcement Technologies

- IBM/BEA announce collaboration on three technologies (11/25/03)

1. **Service Data Objects (SDO)**
   "Perhaps the most important of three standards being proposed, according to IBM and BEA officials, is the one for Service Data Objects. It provides a unifying programming model for data from heterogeneous data sources, including relational databases, XML data sources, Web services, and a range of different enterprise information systems. It offers a simpler programming model that also supports best-practice application design patterns, according to company executives." –InfoWorld
2. **Work Manager for Application Servers**
   Provides a simple API for application-server supported concurrent execution of work items. This enables J2EE- based applications (including Servlets and EJBs) to schedule work items for concurrent execution, which will provide greater throughput and increased response time for applications.
3. **Timer for Application Servers**
   Provides a simple API for setting timers in an application server-supported fashion. This enables J2EE-based applications (including Servlets, EJBs, and JCA Resource Adapters) to schedule future timer notifications and receive timer notifications.

IBM/BEA Public Announcement Press Quotes



- "It is a goal of this collaborative work between IBM and BEA to offer our customers a simpler and more consistent platform for J2EE development."
  Rod Smith, Vice President, Emerging Technologies, IBM Software Group

- "Some users, and certainly ISV partners, have been instrumental in showing us the light. They both have been innovating in a number of areas around Java APIs, and they have been looking for some commonality. [Users] encouraged us to get together and collaborate more closely to find some convergence."
  Scott Dietzen, CTO, BEA

- "We faced feedback, especially from ISVs, which (were less interested) in new features that are WebLogic-specific than ones that were common with WebSphere. This bridging strategy allows us to get common technology and certainly make it standard over time."
  Scott Dietzen, CTO, BEA

- "The additional specifications being done between IBM and BEA results in significant engineering savings for Siebel, as it is now easier for us to build an application that runs on a variety of platforms. The simplification of the programming model, enhanced ease of use, and additional power makes J2EE even more attractive to Siebel as an application platform."
  Ed Abbo, Senior VP of Engineering, Siebel Systems

- "We are glad that IBM and BEA will be contributing their early work as input into JCP standardization efforts."
  Corina Ulescu, Sun spokesperson

Introduction

- Service Data Objects (SDO) is a new data programming technology
- Aims to unify data programming across data source types
- Provides support for common application patterns that are underserved today
- Enable tools and frameworks to more easily query, view, update, and introspect data.

Motivation

- Today, application developers go to great lengths to implement common application patterns
  - Data Transfer
  - Data Binding
  - Disconnected Operations
  - Paging
  - Metadata description
- Have to learn and use different APIs to access different data sources
  - XML, relational databases, EJBs, etc, all have different APIs, but often their needs are the same
- SDO provides a higher level of abstraction
  - Simplifies data programming
  - Allows development tools to rely on a single programming model
  - Does not replace existing technologies – It simplifies them

The Problems

- J2EE and the set of JSRs that extend it (over 200, not all closed) define a very complex programming model. This continues to get worse
- The programming model is strongly technology-oriented – EJBs, connectors, web services, messaging, and so on. Current tools reflect the programming model directly – they do not start with “what do you want to do”, they start with “what technology do you want to use”
- Many common application patterns require extensive, low-level coding
  - Asynchronous, reliable communication between components
  - Stateful services
  - Optimistic concurrency collision detection
- Current tools help with programming the details of this programming model, but not much with the big picture

SDO Goals

- Rapid, simple application development
  - Current J2EE programming models are too complex
- Data-source independent data access
  - Today, there are many different models and APIs for data access in J2EE
  - A single model reduces complexity
- Data-centric data access
  - No behavior associated with data
- XML integrated
  - Easy to transfer data between tiers/Web Services
- Disconnected Model
  - Normal mode of operation for servlets and JSPs
  - Provides a performance advantage by reducing database round-trips

Definition

- Service Data Objects (SDO)

  "Service data objects is a specification for a programming model that unifies data programming across data source types, provides robust support for common application patterns, and enables applications, tools, and frameworks to more easily query, view, bind, update, and introspect data."
  Next Generation Data Programming: Service Data Objects, Beatty, Brodsky, Nally, Patel

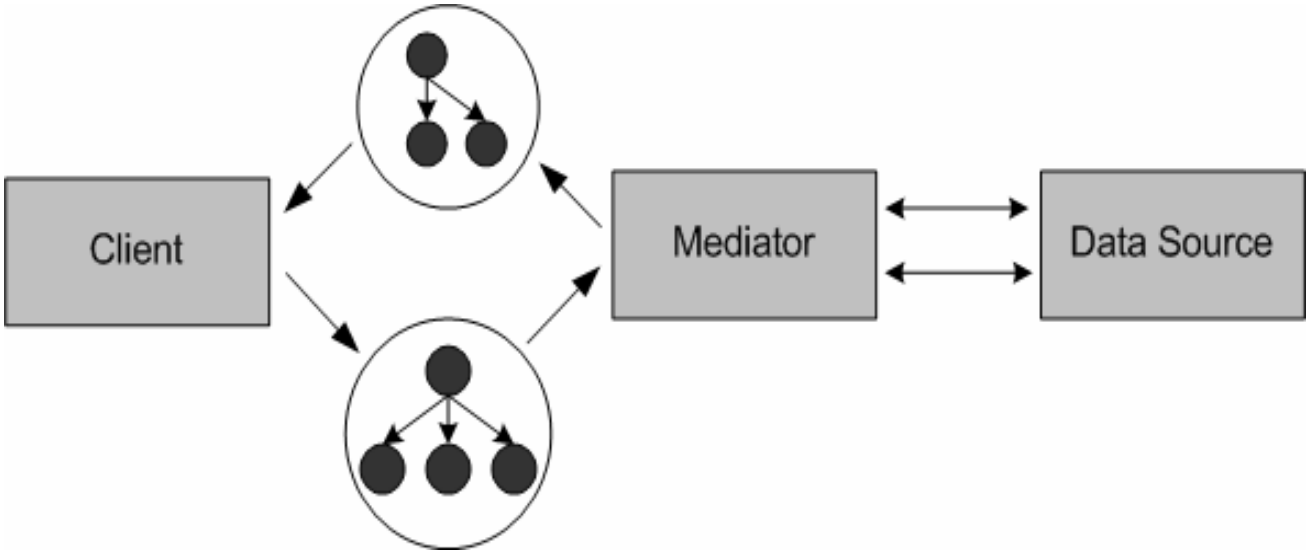
- Key messages:
  1. Programming model specification
  2. Unifies data programming across disparate data sources
  3. Enables standard application development patterns
  4. Enables tools and frameworks to be built to the consistent data model

- Note: SDO, as a unified data representation model, provides a myriad of different value propositions as a function of its rule in a given enterprise application scenario

Service Data Objects
Summary of Goals

1. Provide unified and consistent data access to heterogeneous data sources
   - Simplified programming model for the application programmer
   - Enable tools and frameworks to work consistently across heterogeneous data sources
   - Result:
     Dramatic simplification of the J2EE Programming Model
2. Provide robust programming model support for several J2EE best practice application patterns
   - Disconnected programming model
   - Custom data access layers based on common design patterns
3. Provide first class support for XML Schema, XML InfoSet, and XML data sources
   - XML/Java bindings (JAXB like capability)
   - JAX-RPC objects

Architecture



Architecture - DataGraph

- DataGraph is an “envelope” object.
  - Contains a tree or graph of DataObjects
  - Points to the schema for the DataObjects
- Can be used while disconnected from the original data source
  - Contains the change information for the DataObjects
  - Contains validation error information for the DataObjects
- Implementation based on Eclipse Metadata Framework

Architecture - Mediators

- Mediator responsibilities
  - Build a DataGraph and its associated schema
  - Apply changes stored in a DataGraph to the data source
- Current Mediators
  - JDBC
  - XML
  - EJB

XML and Data Modeling

- XML has changed the landscape dramatically
  - Late 1999/Early 2000
    - XML v1
    - DOM v1 and DOM v2
    - XPath v1
    - XSLT v1
  - Last couple of years and future
    - XML Information Set (InfoSet) v1 (10/01)
    - XML Schema v1 (05/01)
    - SOAP 1.2 (12/02)
    - XQuery 1
    - XPath 2
- Significant shift from viewing XML purely as a text-based format to a data representation formation

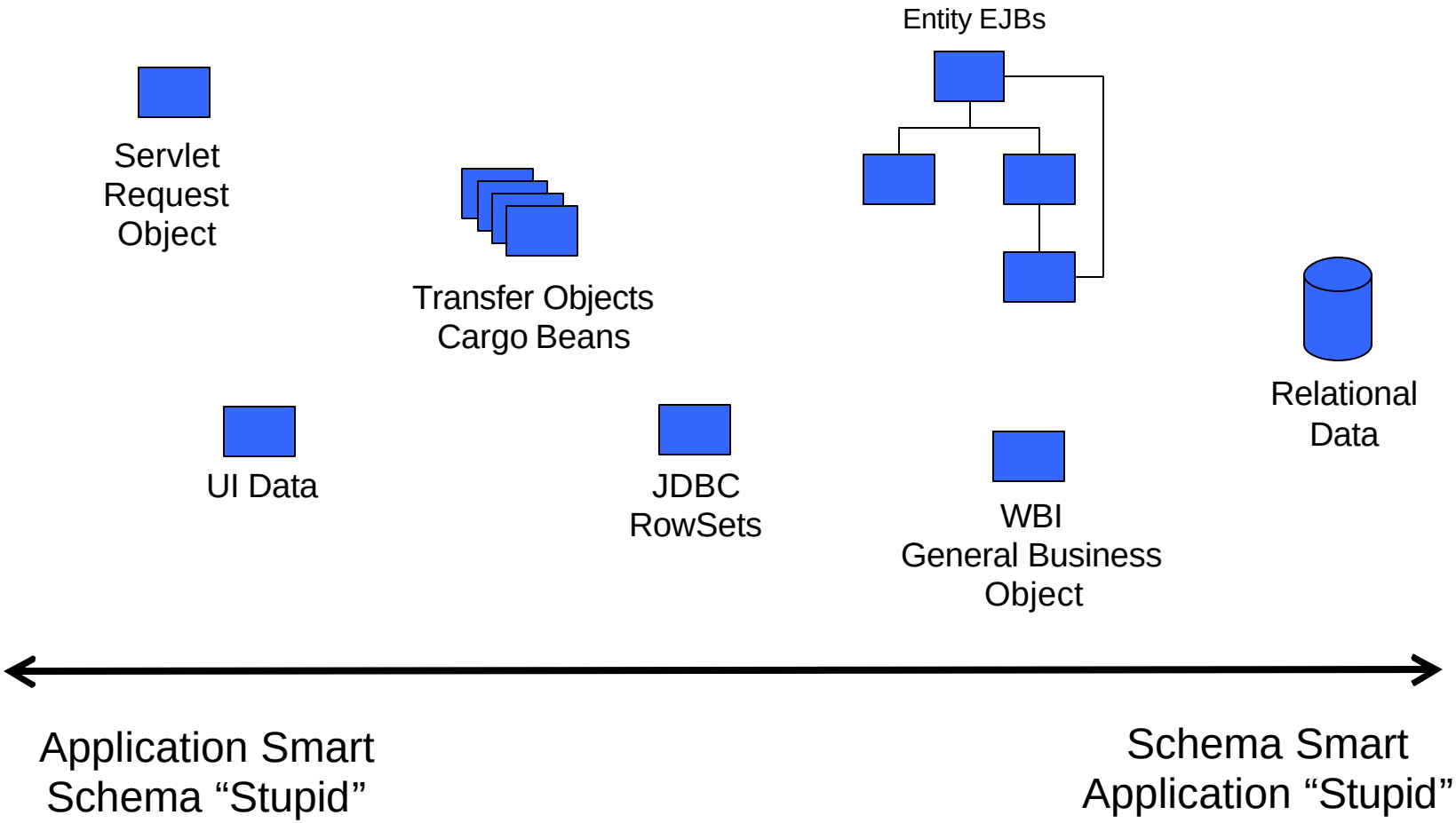
XML Data Model vs Relational Data Model

- Relational Data Model (RDM)
  - The way the data is written to disk is hidden
  - The “Relational Data Model” (primary keys, foreign keys, constraints, et al), represents the external view of the data
  - SQL provides a standard programming model for accessing the RDM
- XML Data Model (XDM)
  - With XML InfoSet, the way the data is written to disk is hidden
  - The XML Data Model represents the external view of the data
  - XQuery/XPath provides a standard programming model for accessing the XDM

XML Data Model and Java

- Common issue called “XML/Java Bindings”:
  - Define data using XML Schema
  - Generate a Java object from that XML Schema
  - *What is the shape and use model of the Java object?*
- Common usage scenarios:
  - JAX-RPC Web Services – what’s the form of the returned object?
  - Reading an XML file on disk into a Java object
- *Today – the IBM, BEA, and Microsoft programming models are all divergent with respect to this issue*

Application Data Spectrum



Classifying Data Representation Models
Connected vs Disconnected Data

- Connected data (or transactional data)
  - Data source dependent/aware (ie Connection object)
  - Examples:
    EJB Entity Beans, JDBC RowSets, and RDB Rows
  - Several J2EE technologies exist for either:
    (a) representing transactional/connected data or
    (b) to access transactional/connected data

- Disconnected data (or non-transactional data)
  - Data source independent/unaware
  - Examples:
    JavaBeans, Documentation Object Model (DOM), Eclipse Modeling Framework
  - Common best practice patterns:
    Transfer Objects, Cargo Beans, Data Objects, Non-transactional Data Objects, Replication Data Objects
  - Designed as a lightweight data container

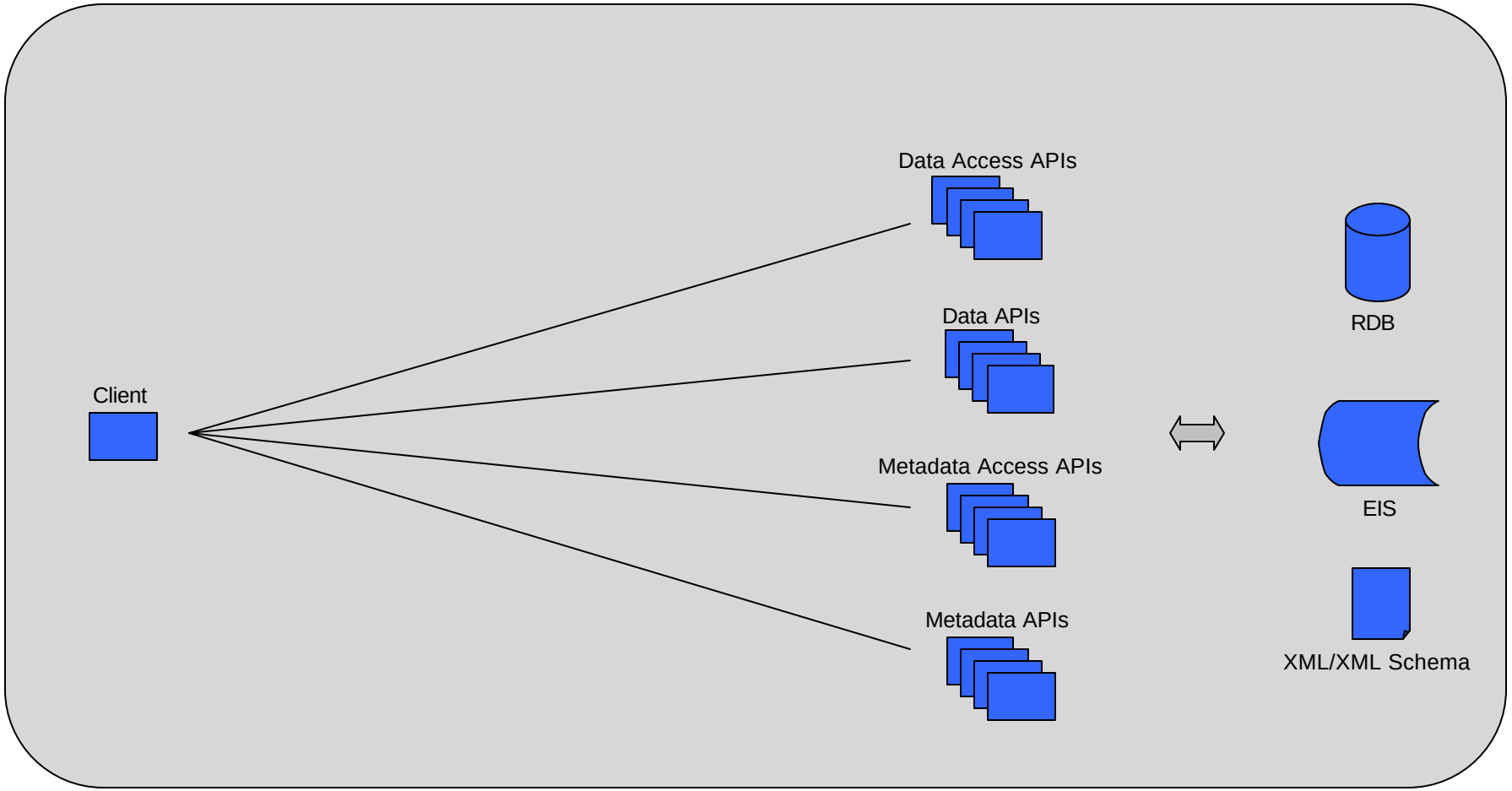
Classifying Data APIs
Static vs Dynamic

- Examples:
  - Entity EJB – static
  - JDBC Rowset - dynamic
- Traditional design tradeoffs:
  - Useability
  - Rigidity
  - Runtime overhead
  - Design time overhead (code generation)
  - Compile time checking
  - Code completion capability
  - Flexibility

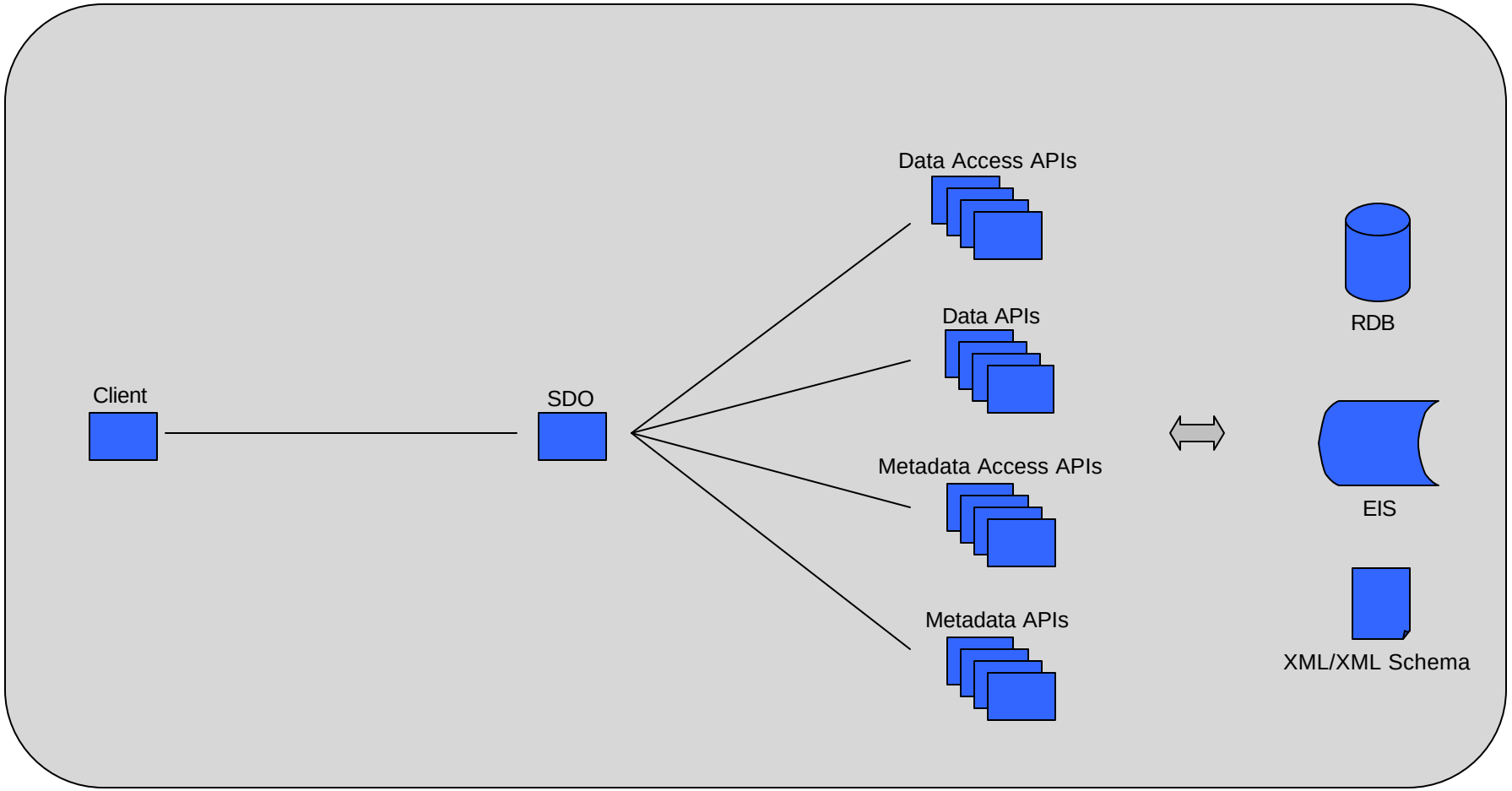
Data and Metadata APIs

	Model	API	Data Source	MetaData API	Query Language
SDO	Disconnected	Both	Any	SDO Metadata API, Java Introspection	Any
JDBC Rowset	Connected	Dynamic	Relational	Relational	SQL
JDBC Cached Rowset	Disconnected	Dynamic	Relational	Relational	SQL
Entity EJB	Connected	Static	Relational	Java Introspection	EJBQL
JDO	Connected	Static	Relational, Object	Java Introspection	JDOQL
JCA	Disconnected	Dynamic	Record-based	Undefined	Undefined
DOM and SAX	Disconnected	Dynamic	XML	XML InfoSet	XPath, XQuery
JAXB	Disconnected	Static	XML	Java Introspection	N/A
JAX-RPC	Disconnected	Static	XML	Java Introspection	N/A

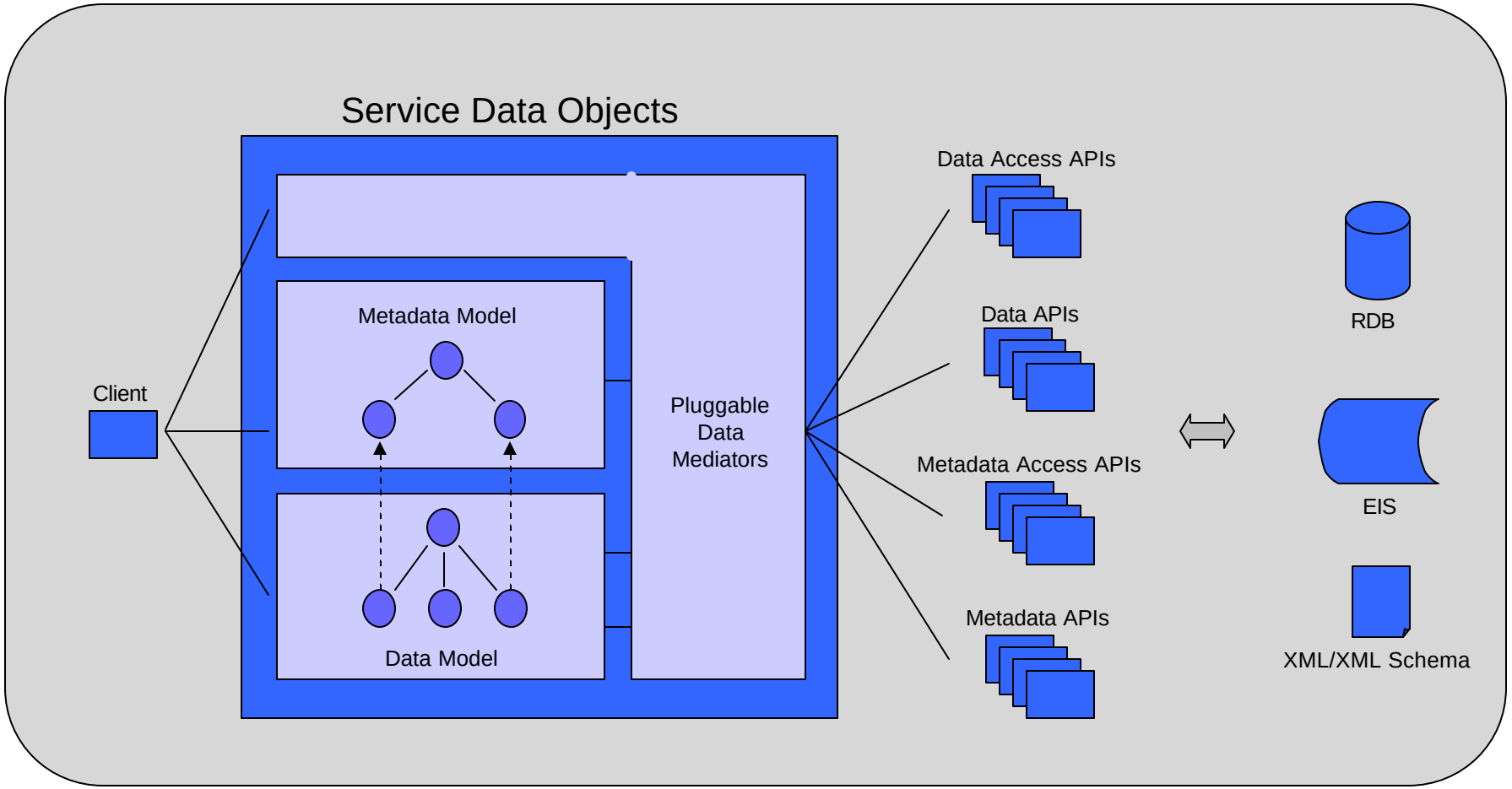
Existing J2EE Architecture



SDO Architecture
Mediator Pattern



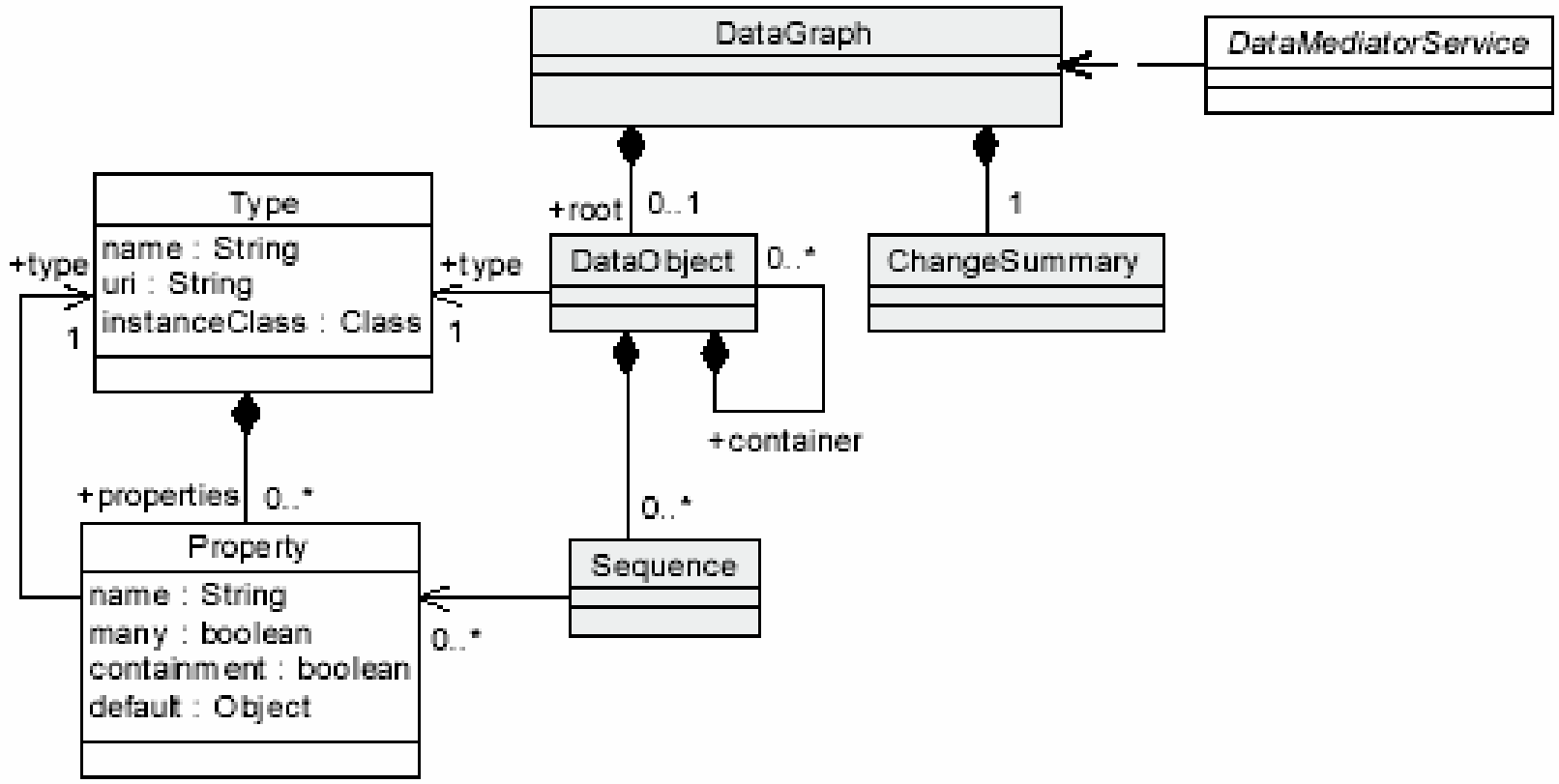
Service Data Object Runtime Architecture



SDO Design and Runtime Components

1. SDO Core
   - Data Objects
   - Data Graphs
   - Introspection APIs
2. SDO Data Mediator Services
   - Query back end data source
   - Create data graphs
   - Manage optimistic concurrency
3. SDO Tools
   - Code generators
   - Metamodel converters
   - Schema converters
   - Data modeling tools
   - Schema modeling tools
4. SDO Enabled Runtimes and Frameworks
   - Data binding to UI

SDO Core
UML Model



SDO Core
DataObjects (Page 1 of 2)

- Next generation JavaBeans
- Purpose is as a source independent data container
  - Primitives (Java/XML Schema like primitive types)
  - References to other data objects
- Does not a provider of business logic methods

- Examples:
  - XML Schema
    DataObject would represent a complex type, with attributes being represented as primitives, and child complex type elements represented as references
  - Relational Database
    DataObject might represent a row of data

SDO Core
DataObjects (Page 2 of 2)

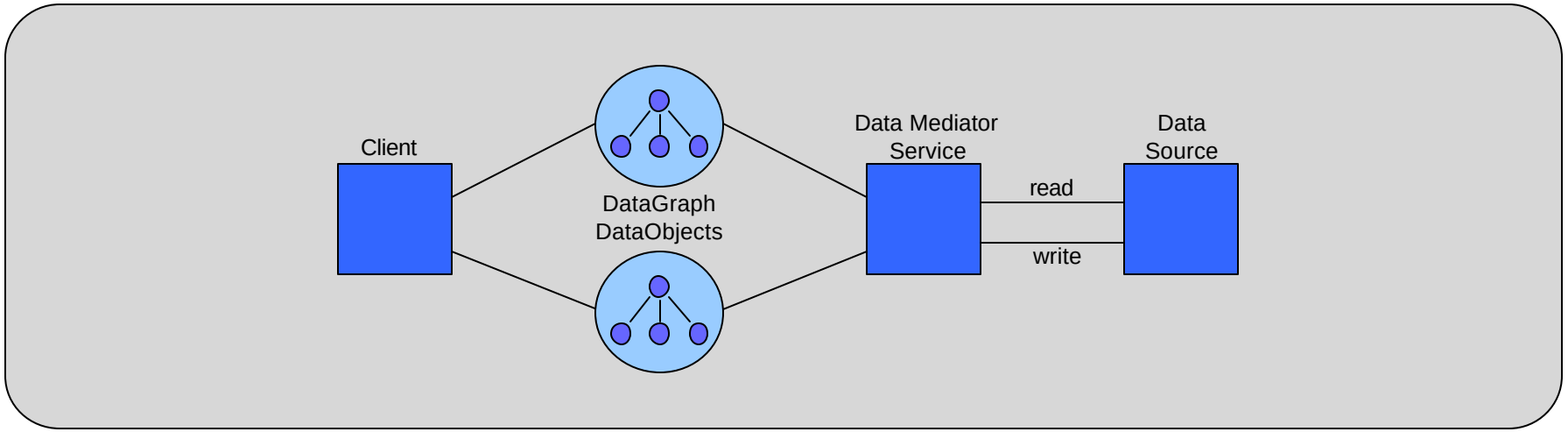
- Metadata introspection capabilities
  - Enables access to types, relationships, and constraints
  - Metadata can be generated from XML Schema, Java interfaces, XMI, et al
- Dynamic interface or you can generate a statically-typed interface from metadata
- Rich relationship integrity management
  - Supports 1:1, 1:n, and n:m relationships
  - Auto-manages inverse relationships
  - Supports containment and reference semantics
- Event management facilities
- XML friendly: supports XML Schema for metadata and XML as a data source, and supports XPath expressions to get/set values (looking at XQuery)

SDO Core
DataGraph

- Contains a single DataObject
- References the schema for the DataObjects
- Records change summary information accessible by mediators to provide optimistic concurrency control semantics
- Flows as an XML Message (eg Datagraph.xsd)

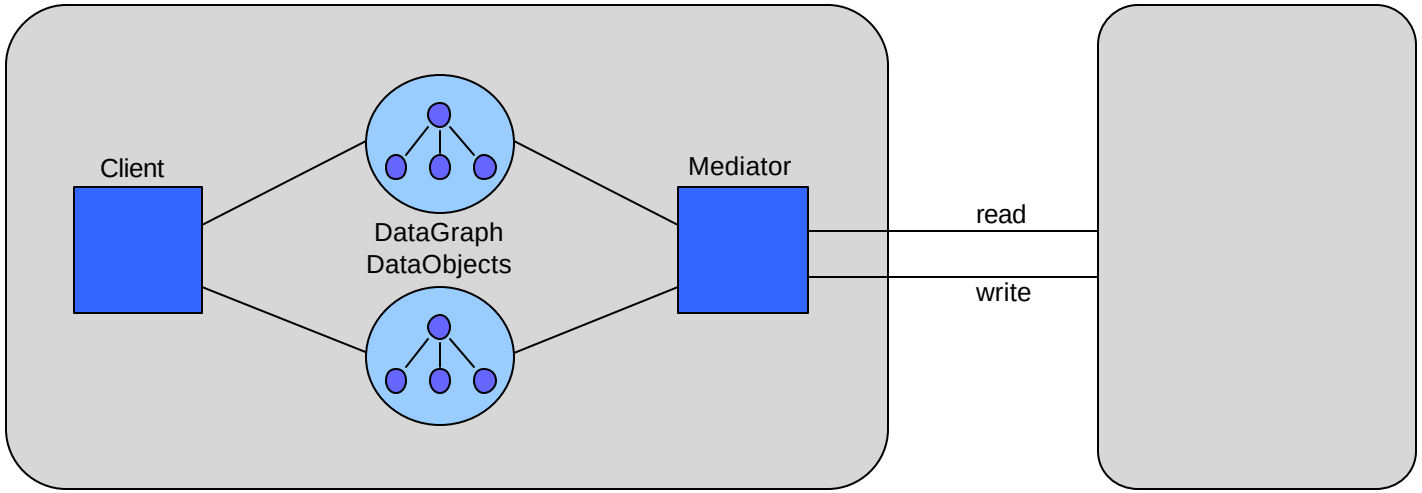
SDO
Data Mediator Services

- Responsibilities
  - Query data source
  - Creating graphs of data containing data objects
  - Looks to see if concurrency control was violated
  - Applies data graph changes back to the data source



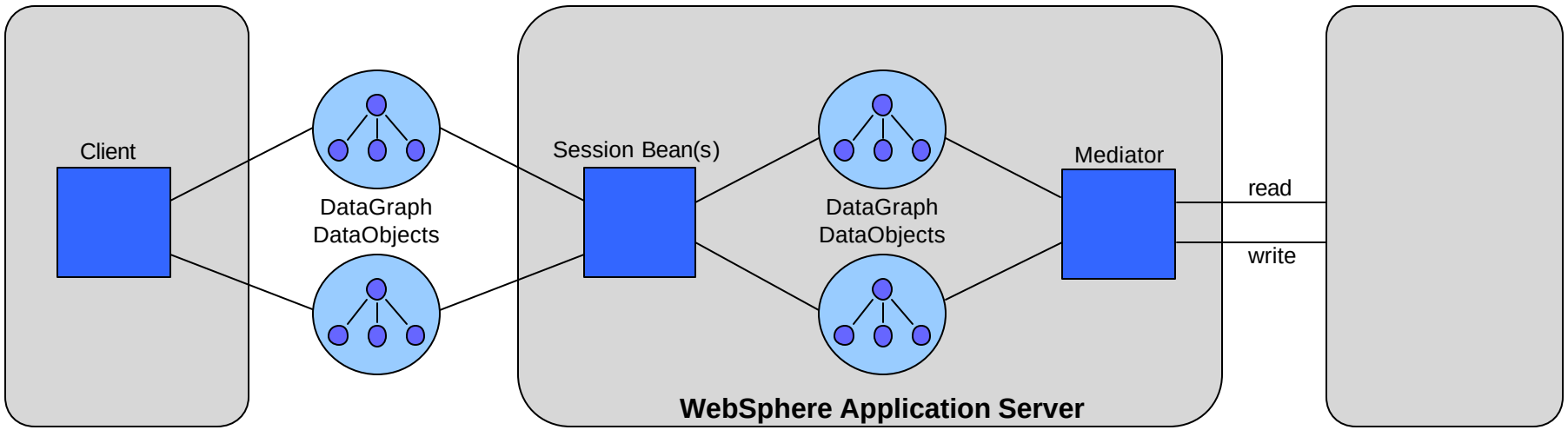
Topology 1

- Client talks directly to an in process mediator



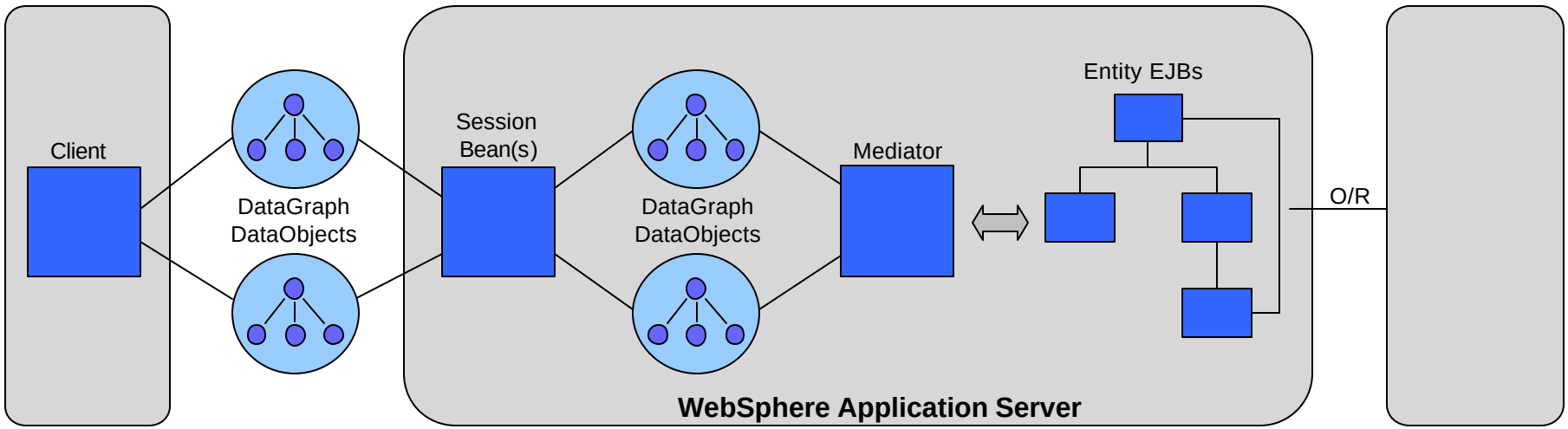
Topology 2

- Client talks to a service, in this case a stateless session bean
- The stateless session bean(s) talk to the mediator to access the data source
- In this case, the mediator goes remotely to a data source



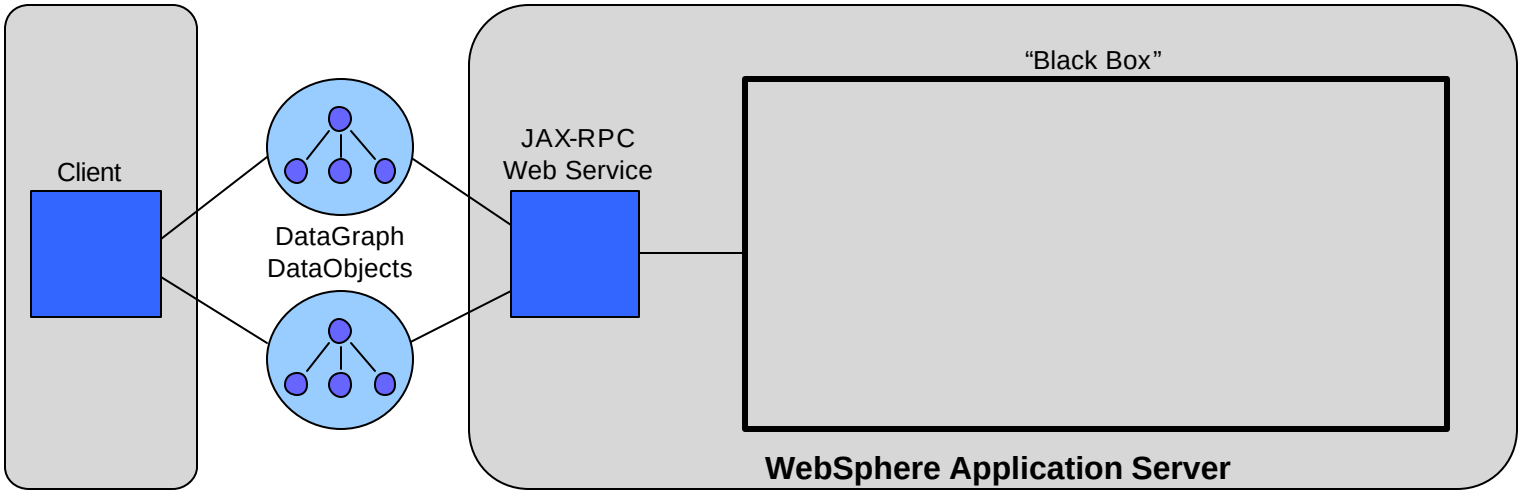
Topology 3

- Client talks to a service, in this case a stateless session bean
- The stateless session bean(s) talk to the mediator to access the data source (Note: Although the persistence form changes dramatically from Topology 2, the source code in the session bean(s) changes very little – the Mediator is acting as a data facade)
- In this case, the mediator is mediating access to Entity EJBs
- The WebSphere Entity EJB Container manages the Object/Relational mapping of the Entity to the data source



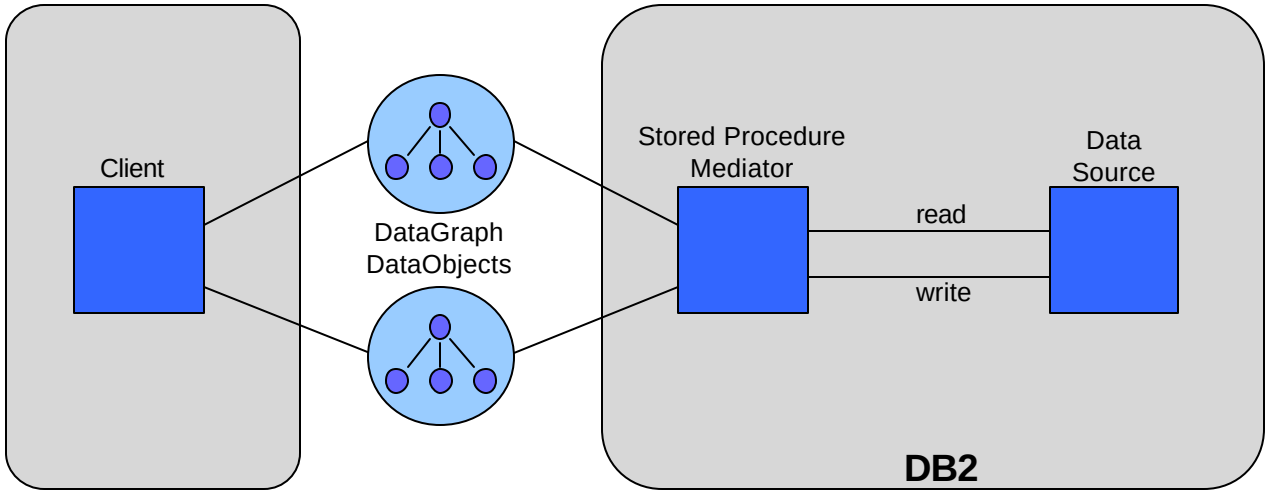
Topology 4

- Client talks to a JAX-RPC based web service passing a DataGraph/DataObjects
- The web service does its processing a returns DataGraph/DataObjects back to the client
- It is quite possible that the black box looks like Topology 2 or 3 above, where the web service is assuming the same role as the session bean



Topology 5

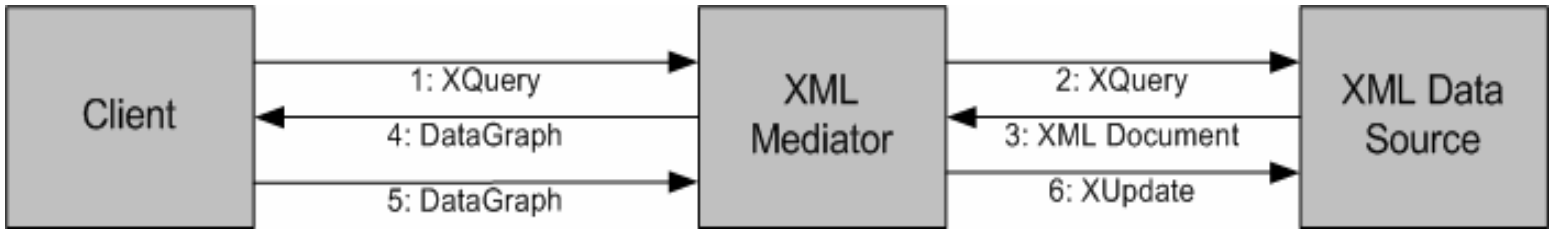
- Mediator potentially running as a stored procedure
- Returns DataGraph/DataObjects back to the client
- Takes DataGraph/DataObjects as parameters



Use Cases - XML

- XML programming capabilities far exceed what is available today
- Provides Java-XML data binding and XPath- and XQuery-based querying
- Disconnected data operations
- Under this architecture, the XML data source could be either an XML file, a native XML data store, or a relational database with XML features.

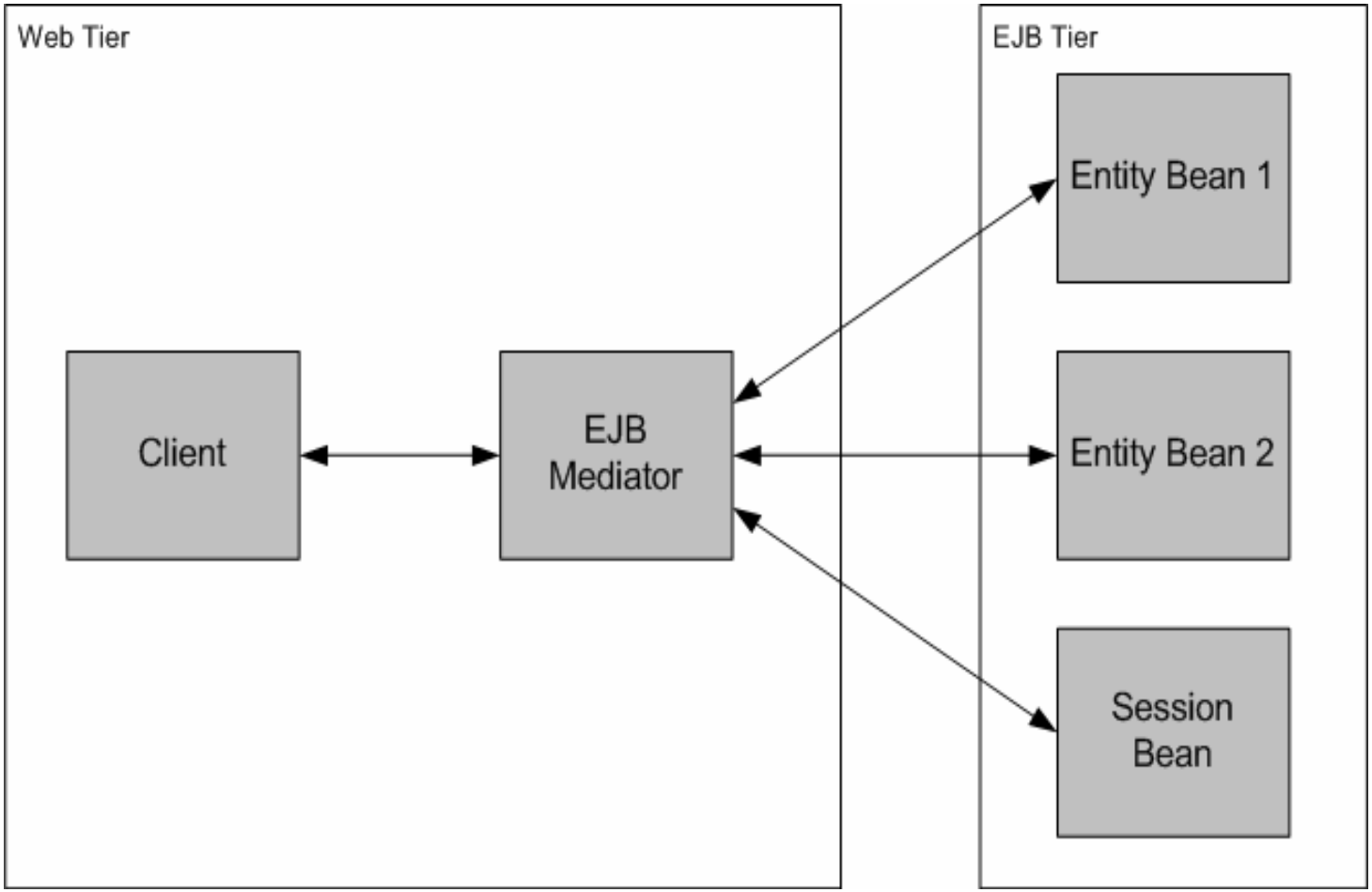
Use Cases - XML



Use Cases - EJB

- The “Data Transfer Object” and “Data Access Object” design patterns are often used with EJBs
- Data Transfer Objects are used to pass data between the presentation tier, the business tier and the persistence tier
- The Data Transfer Object represents the data in a way that is independent of the underlying persistence technology
- Data Access Objects abstract and encapsulate a data source by creating and using Data Objects as the neutral form of data across applications and data sources.
- Today, developers implement these patterns manually.

Use Cases - EJB



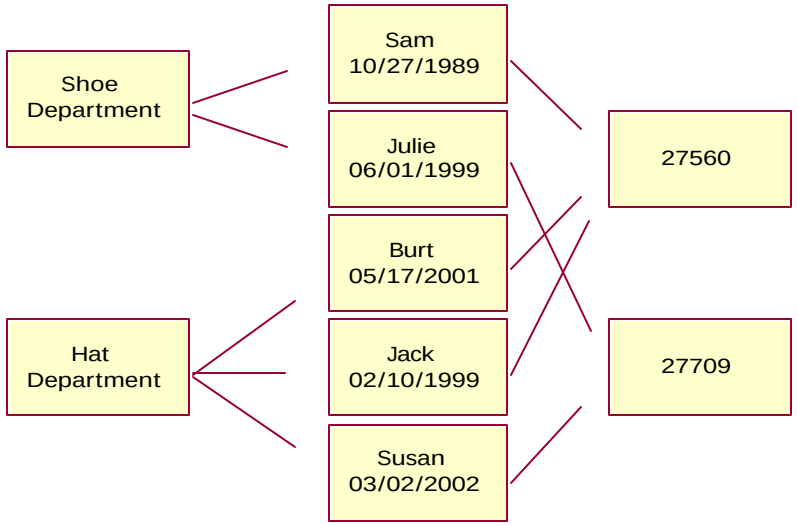
Use Cases - JDBC

- JDBC returns results in a flat, tabular format that does not resemble the database structure

Department	Name	Hire Date	Zip Code
Shoe Department	Sam	10/27/1989	27560
Shoe Department	Julie	06/01/1999	27709
Hat Department	Burt	05/17/2001	27560
Hat Department	Jack	02/10/1999	27560
Hat Department	Susan	03/02/2002	27709

Use Cases - JDBC

- The JDBC Mediator normalizes the data so that the underlying database relationships are represented



Use Cases - JDBC

- Old Way to get all shoe department employees:

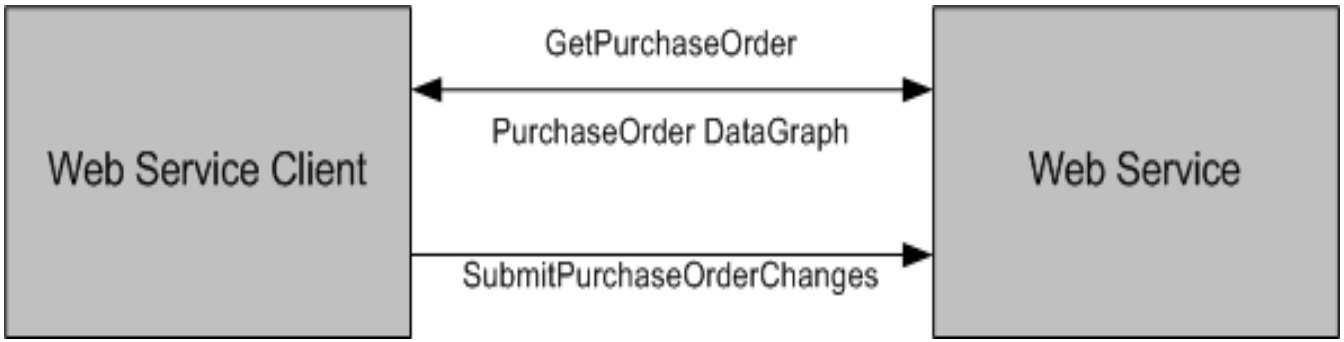
```
while (resultSet.next()) {
    String department = resultSet.getString(1);
     if ( department.equals("Shoe Department") ) {
                String employeeName = resultSet.getString(2);
       employees.add(employeeName);
     }
}
```

- New way:

```
List employees =shoeDepartment.getList("employees");
```

Use Cases – Web Services

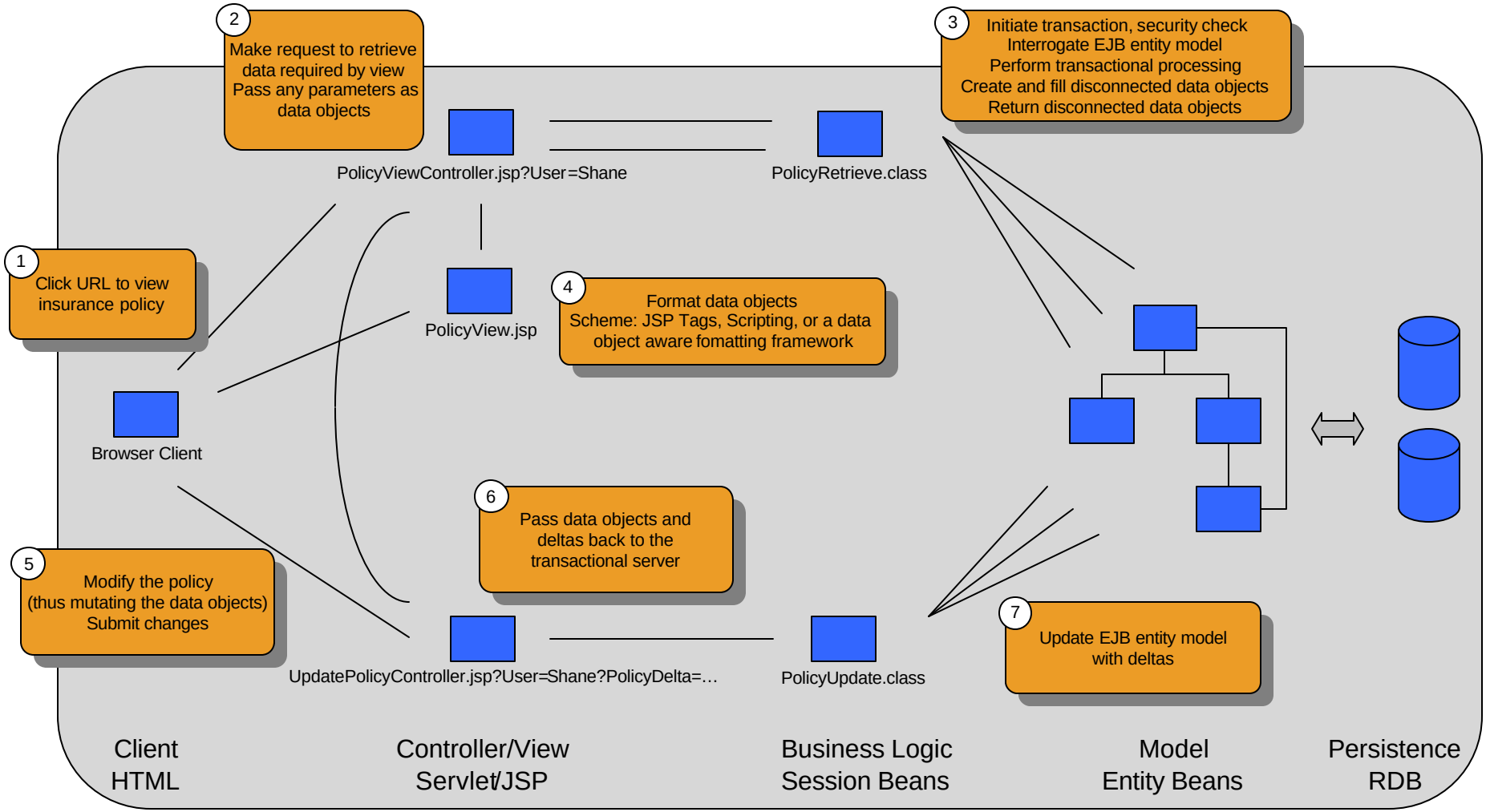
- DataGraphs can be used for transporting data over the wire:



Sample Scenario

- Prototypical J2EE technology set – servlet/session bean/entity bean
- Insurance policy retrieval, and modification
- Assume policy is viewable by end consumer as well as agent

Sample Scenario – Viewing/Updating an Insurance Policy
Traditional Best Practices using Connected/Disconnected Data



Disconnected Programming Model

- Insurance policy view/update scenario demonstrates a traditional web based disconnected model:
  1. Client makes policy view reqeust
  2. Controller requests policy data from transactional server
  3. Server starts transaction
  4. Server retrieves data from transactional resource
  5. Server copies data into non-transactional disconnected data objects
  6. Server commits transaction
  7. Controller combines data objects with render objects (widgets, tags, et al) to produce client view
  8. Client updates non-transactional disconnected data objects, submits changes
  9. Controller delegates changes to the transactional server
  10. Server starts transaction
  11. Server validates data concurrency semantics (see OCC)
  12. Server persists changes back to the transactional resource
  13. Server commits transaction

- Data is “checked out” of the data store for some “period” of time (but not locked)
  - Possibility exists that the data might become stale
  - Possibility exists that someone else might change the same data elsewhere
- The disconnect data scenario applies to several other enterprise architecture use models:
  - Offline mode (Lotus Notes replication semantics, PDA synchronization, et al)
  - B2B SCM/PRM – Company A obtains data from company B, modifies, returns updated data

Optimistic Concurrency Control (OCC)
Popular strategy

- Scenario:
  My insurance agent and I are updating my insurance policy concurrently
- The optimistic concurrency control strategy supports this use model
  - Increased concurrency
  - Increased throughput
- Collision detection strategy
  - Disconnected data objects maintain primary key, old value, and new value
  - When disconnected data object changes are written to the database, the old value is first checked to make sure it is still the same, before the new value is applied
  - If the old value in the DataObject differs from the current value in the transactional data store, an error is thrown to the application, otherwise the update is completed
- Note: there are multiple strategies for managing OCC

Several best practice patterns and strategies

- Separation of concerns between data fetch and data render logic
- Session façade pattern with EJB Entity CMP to represent business model
- Actions aggregated into a single transaction providing ACID reads/updates
- Transfer objects used between view/controller and the transactional tier
- Application of data render objects to transfer objects for efficient and reusable user interface programming
- Server transaction and connection resource utilization minimized
- Data Object change summary used to optimize transactional update operation
- Optimistic concurrency pattern leveraged to enable high transaction throughput

- Note: Items affected by SDO are in the Blue font

Applying SDO to the Insurance View/Update Scenario

- Transfer objects used between view/controller and the transactional tier
  - The transfer objects could use the rich DataGraph/DataObject data types
  - The view/controller could communicate directly with an Entity CMP mediator in order to obtain the disconnected data. If processed needed to occur in addition to the data fetch – the session bean could use the Entity CMP mediator.
- Application of data render objects to transfer objects for efficient and reusable user interface programming
  - SDO enables frameworks like JSF to bind their widget set to a single unified data representation format
  - Since SDO is likely to enable constrain frameworks, those can be leveraged by the user interface componentry to provide client side cascading delete semantics, field uniqueness, read-only semantics, et al
- Disconnected and OCC programming
  - The change set automatically maintained by the DataGraph enables the data mediators to provide optimistic concurrency control for the application programmer
  - Maintaining change information on the client enables the application to only send back the change sets to the server optimizing network bandwidth usage
- How much of the model would have to change if the data source changed from Relational to IMS (or vice versa)?
  - Leveraging the mediator pattern and the unified DataGraph/DataObject representation scheme minimizes the ripple effect resulting from either an data access API or data representation change
- Although not mentioned above, SDO provides several application server caching opportunities to further enhance application performance and scalability

SDO Release Status

- WSADIE 5.1
  - EMF Project – Can import XML Schema, Java interface, et al, and generate statically typed EMF Java objects
  - Precursor of SDO (without detached programming support, XML serialization, enhanced XML schema support, et al)
- WSAD 5.1.2
  - Contains JSF user interface components and tooling
  - SDO DataGraph/DataObject under the covers
  - Relational mediator (dynamically typed DataObjects only)

Competitive Landscape

- BEA
  - XML Beans
  - Transition to SDO
- Microsoft
  - ADO.Net (.Net v1.1 System.Data classes)
    - Relational data model
  - XML Mediators (.Net v2 System.Xml classes)
    - XML data model
  - Microsoft ADO.Net disconnected data set interoperability
    - ADO.Net provides a Dataset – much like Datagraph, but it is constrained to relational semantics (RDM)
    - ADO.Net provides an on the wire format called Diffgram
    - Potential exists to enable some degree of "on the wire" interoperability between Microsoft .Net and J2EE detached data set objects flowing between services
    - Many details need to be looked at…

Resource References

- IBM DeveloperWorks
  - http://www-106.ibm.com/developerworks/java/library/j-commonj-sdowmt/
  - SDO Specification
  - SDO Whitepaper
- Eclipse Modeling Framework (EMF)
  - http://www.eclipse.org/emf
- Announcement press
  - http://xml.coverpages.org/ni2003-11-25-a.html
  - http://www.infoworld.com/article/03/11/25/HNibmbeajava_1.html
  - http://zdnet.com.com/2100-1104_2-5111567.html
  - http://www.techweb.com/wire/story/TWB20031125S0010
- Transfer Object Pattern
  - http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html
- AlphaWorks ETTK – SDO Technology Preview
  - http://www.alphaworks.ibm.com
- The Eclipse Modeling Framework
  - Budinsky, Merks, et al
- A First Look at ADO.NET and System.Xml v2.0
  - Homer, Sussman, Fussell

Questions?