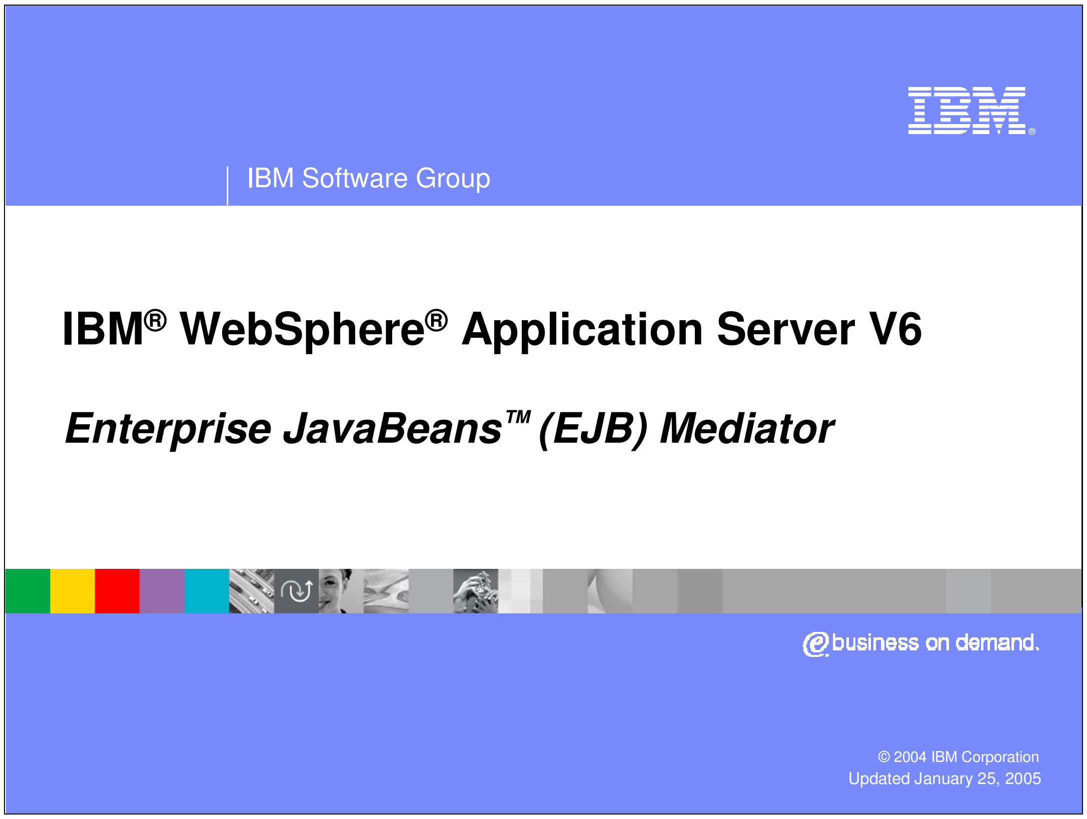


IBM Software Group

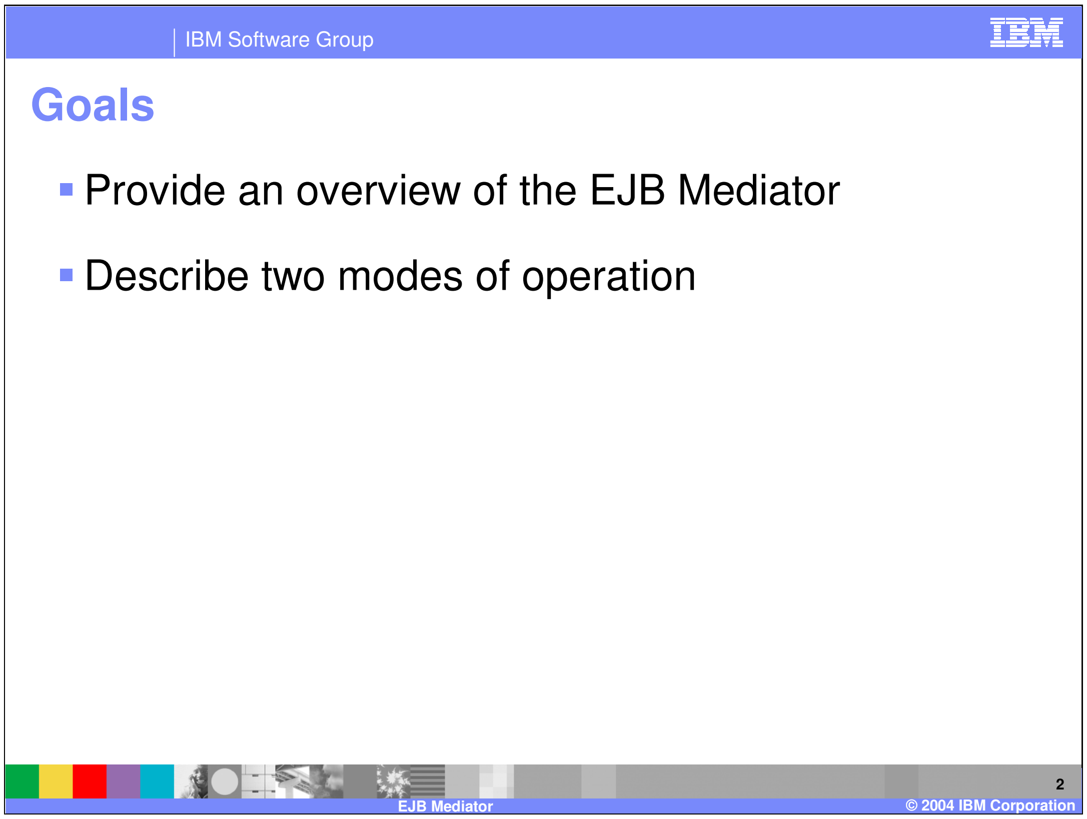
IBM® WebSphere® Application Server V6

Enterprise JavaBeans™ (EJB) Mediator

@business on demand.

Updated January 25, 2005

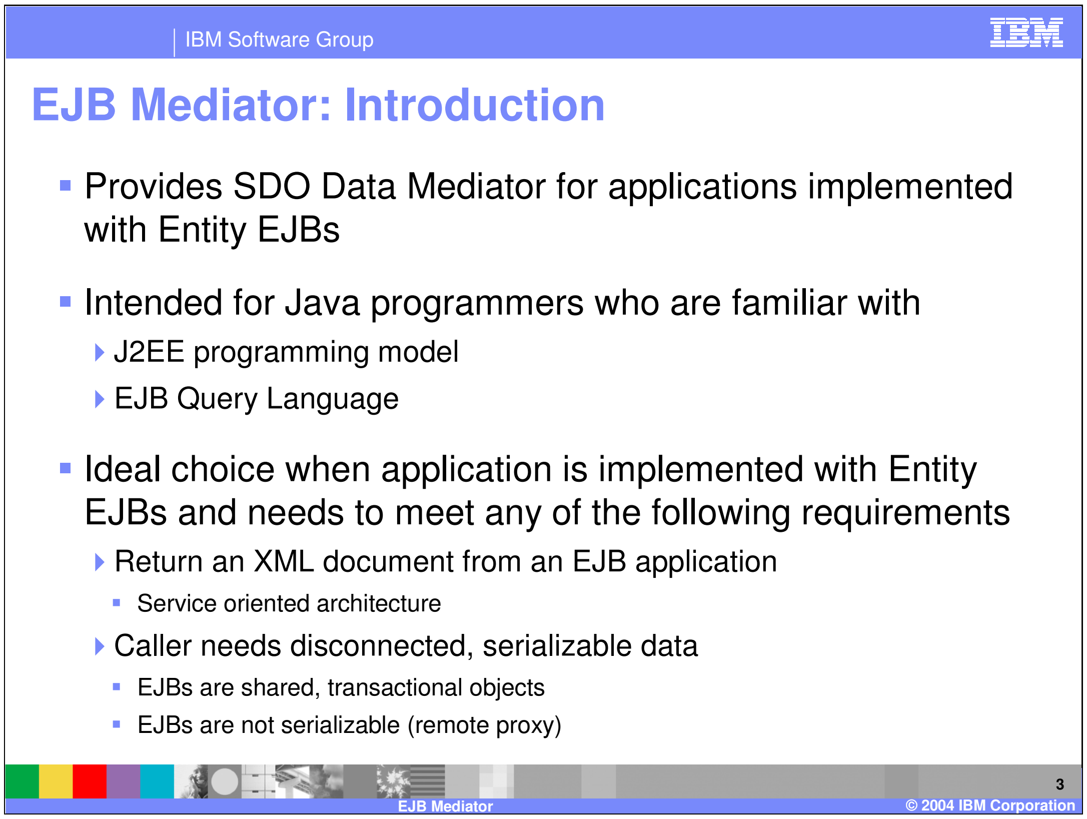
This presentation, will focus on providing an overview of the EJB Mediator.



Goals

- Provide an overview of the EJB Mediator
- Describe two modes of operation

The primary goal of this presentation is to provide an overview of the EJB Mediator.



EJB Mediator: Introduction

- Provides SDO Data Mediator for applications implemented with Entity EJBs
- Intended for Java programmers who are familiar with
  - J2EE programming model
  - EJB Query Language
- Ideal choice when application is implemented with Entity EJBs and needs to meet any of the following requirements
  - Return an XML document from an EJB application
    - Service oriented architecture
  - Caller needs disconnected, serializable data
    - EJBs are shared, transactional objects
    - EJBs are not serializable (remote proxy)

The EJB Mediator provides an SDO data mediator for applications developed using Entity EJBs (CMP type only). The EJB Mediator allows programmers to specify a compound EJB query statement to access entity EJB data through the DataGraph/DataObject interfaces. The EJB Mediator is targeted primarily for Java programmers that are already familiar with the J2EE programming model and the EJB Query Language.

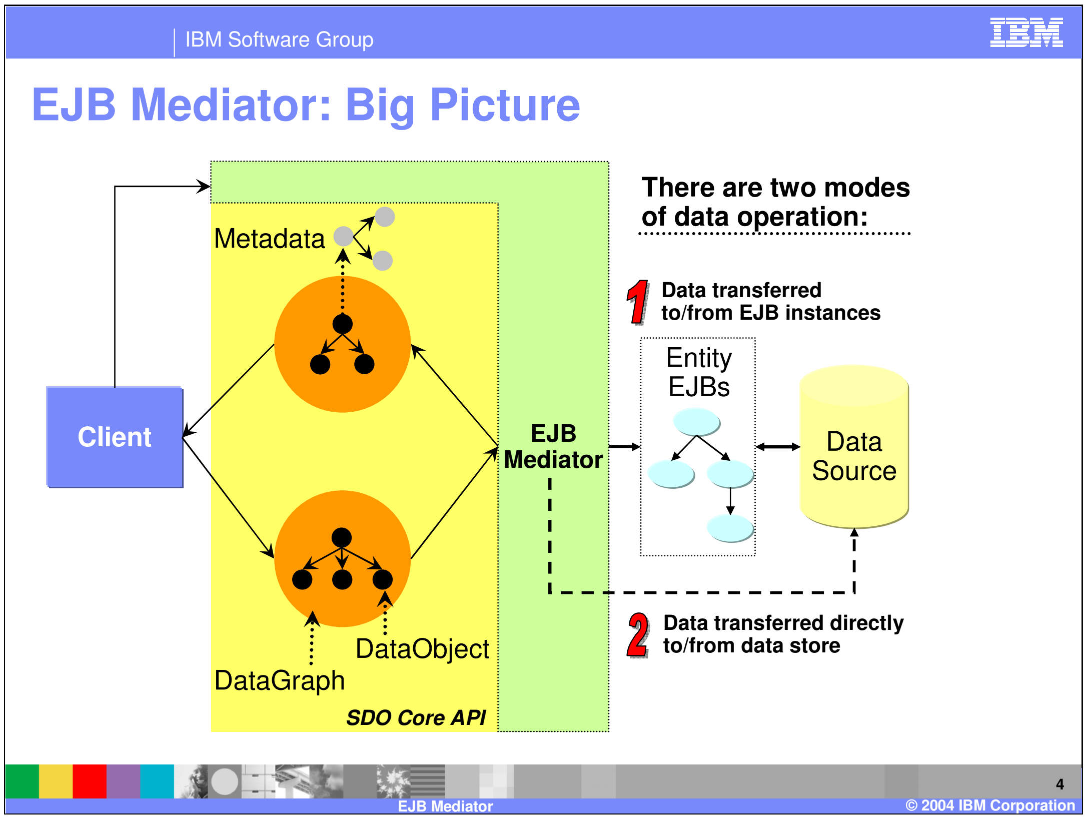
A natural question is, when would you use the EJB Mediator? The EJB mediator should be considered when an application is architected to use entity EJBs and any of the following items applies to the application design:

--> You need to return an XML document from an EJB application (for example, in the case of a service oriented architecture)

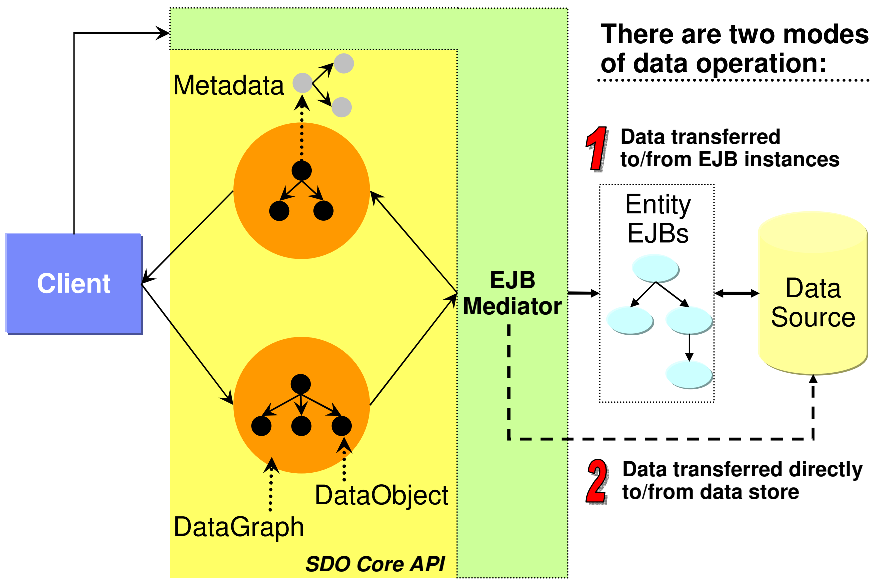
--> Caller needs disconnected, serializable data

--> Application is being built using tools that use DataObjects (for example, IBM Rational Application Developer V6 will have new EJB generation tools that will replace the EJB Access bean APIs with DataObjects and Mediators)

One final implementation note: Although the EJB Mediator facilitates access to entity EJBs, it is not itself an EJB.



EJB Mediator: Big Picture



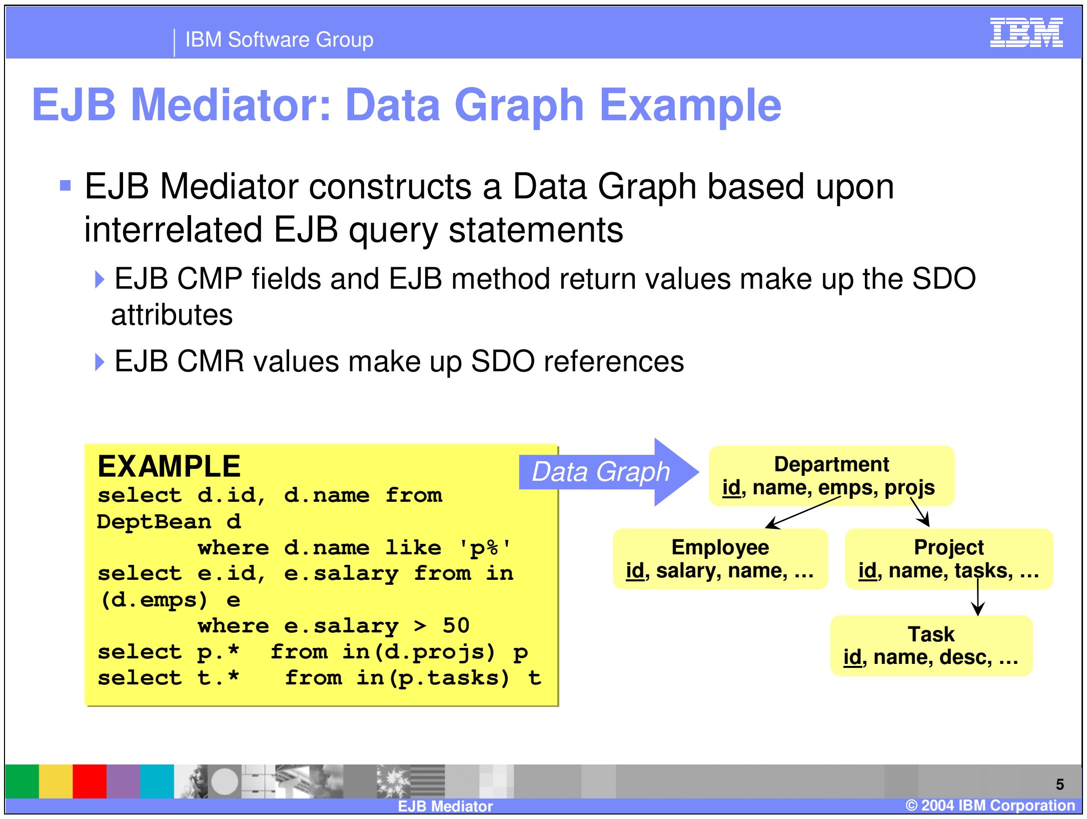
There are two modes of data operation:

1 Data transferred to/from EJB instances

2 Data transferred directly to/from data store

From the client perspective, the EJB mediator looks very similar to any other type of data mediator. Recall from previous discussions that all mediators provide a uniform view of the backend data through the SDO core APIs (DataObject and DataGraph). However, there are typically a small set of features that are specific to a particular type of mediator. The diagram shown on this slide is intended to highlight one of the important features provided with the EJB Mediator.

When using the EJB mediator, there are two modes of operation to work under when accessing entity EJB data. In the first mode of operation, the EJB mediator uses active EJB instances to transfer data to/from the data source. However, in the second mode of operation, the EJB mediator transfers data to and from the data store directly rather than activating the EJB.



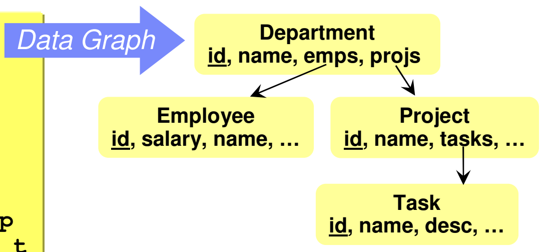
IBM Software Group

EJB Mediator: Data Graph Example

- EJB Mediator constructs a Data Graph based upon interrelated EJB query statements
  - EJB CMP fields and EJB method return values make up the SDO attributes
  - EJB CMR values make up SDO references

EXAMPLE

```
select d.id, d.name from
DeptBean d
       where d.name like 'p%'
select e.id, e.salary from in
(d.emps) e
       where e.salary > 50
select p.*  from in(d.projs) p
select t.*   from in(p.tasks) t
```



5

EJB Mediator © 2004 IBM Corporation

The EJB mediator allows a client to specify an EJB query statement that is used to build a data graph. The attributes for the DataObjects making up the DataGraph correspond to the EJB CMP fields or return values from EJB methods included in the EJB query statement. The SDO references (the way in which a DataObject in the graph refers to another DataObject) are related to EJB CMR values.

Included on this slide is an example EJB Query statement, along with visual representation of the DataGraph that would result from this query. The following lists the schema for this example:

---- EJB Schema ----

Department

CMP Fields (id, name, budget)

CMR (emps=>Collection of Employee, projs=>Collection of Project, mgr=>Employee)

Employee

CMP Fields (id, name, salary, bonus)

CMR (dept=>Department, tasks=>Collection of Task, manages=>Collection of Department)

Project

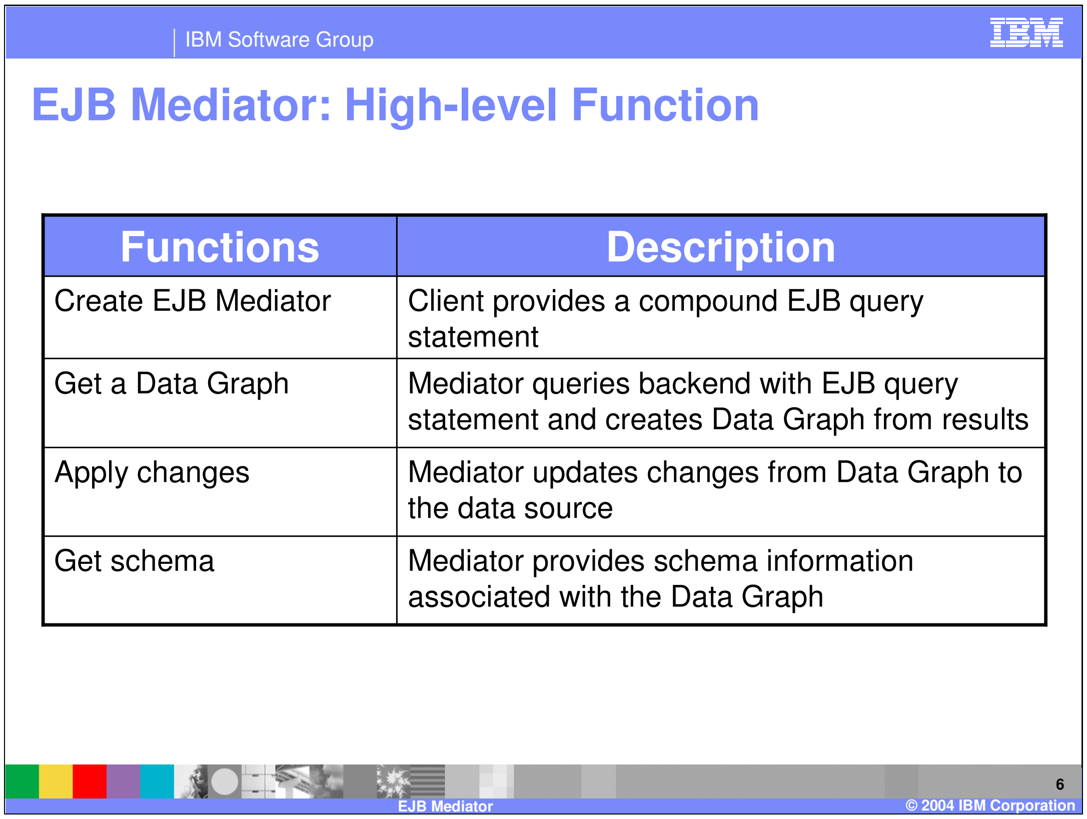
CMP Fields (id, name, cost)

CMR (tasks=>Collection of Task, dept=>Department)

Task

CMP Fields (id, name, desc)

CMR (proj=>Project, emps=>Collection of Employees)


IBM Software Group

EJB Mediator: High-level Function

Functions	Description
Create EJB Mediator	Client provides a compound EJB query statement
Get a Data Graph	Mediator queries backend with EJB query statement and creates Data Graph from results
Apply changes	Mediator updates changes from Data Graph to the data source
Get schema	Mediator provides schema information associated with the Data Graph

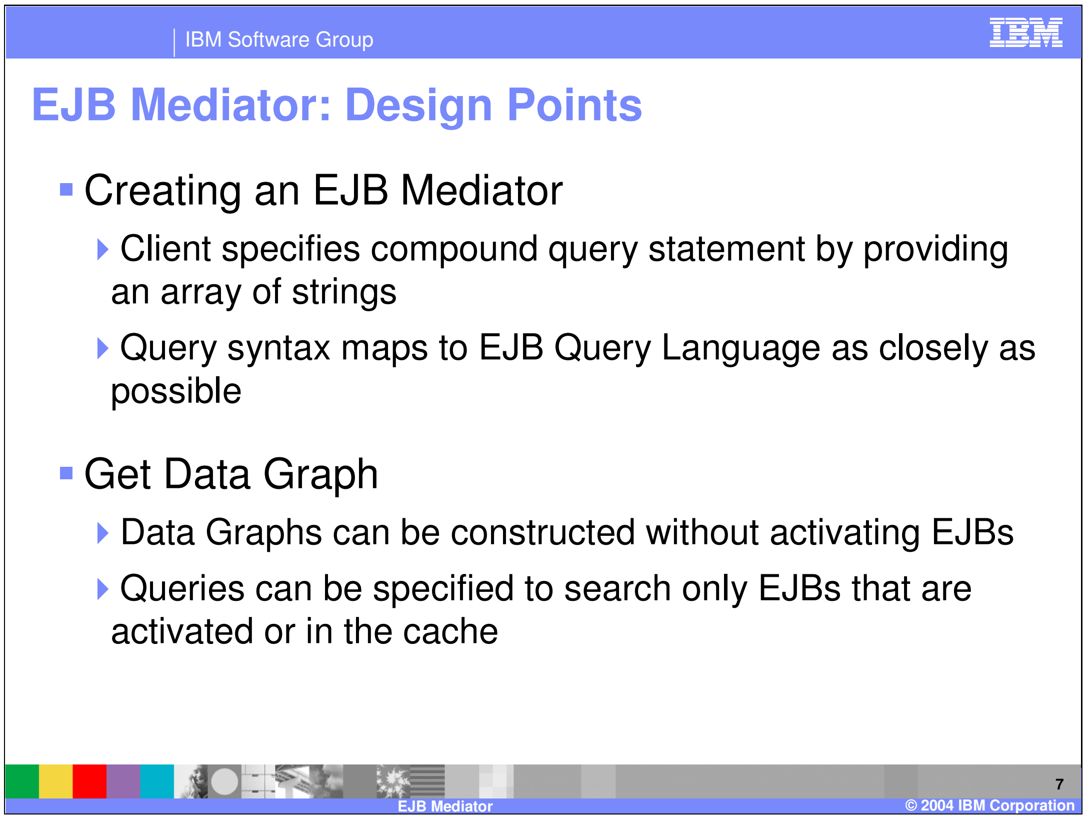
6

EJB Mediator

© 2004 IBM Corporation

The high level functions for the EJB Mediator are similar to the JDBC Mediator. These high level functions represent the functions that most mediators would be expected to support.

Note that the "Get Data Graph" and "Apply changes" functions can be applied directly to the database or through activated EJB instances as discussed previously.



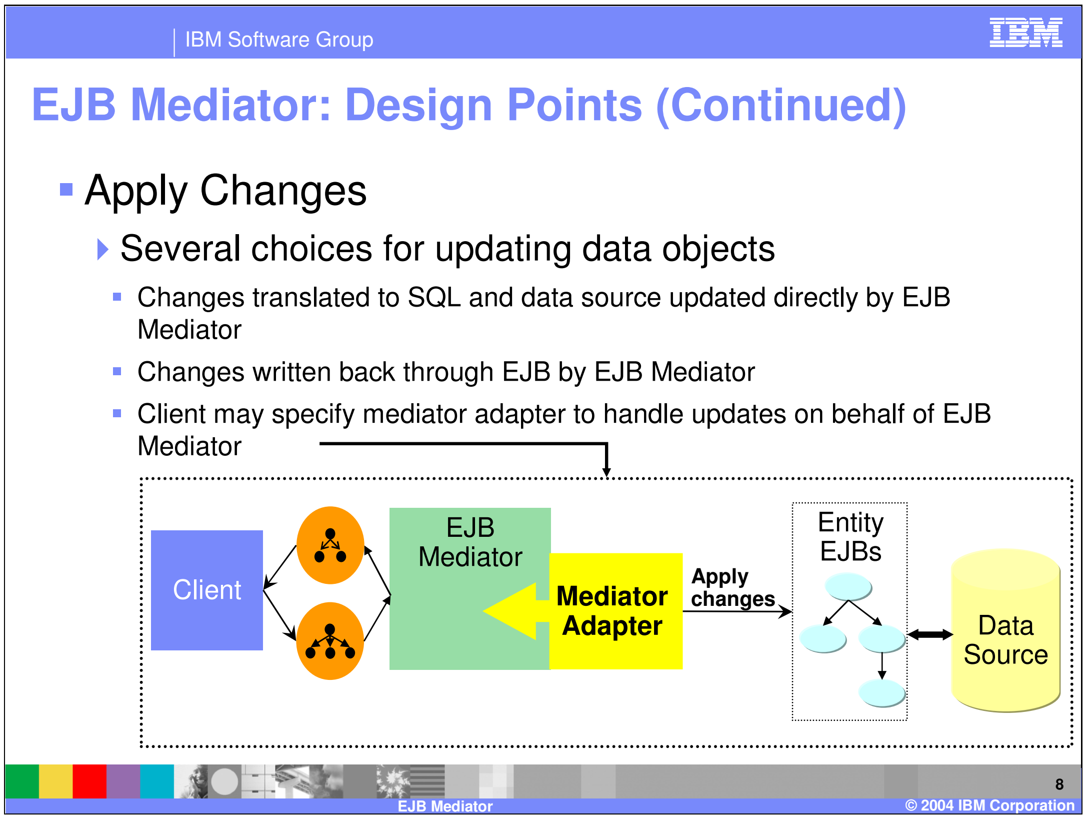
EJB Mediator: Design Points

- Creating an EJB Mediator
  - Client specifies compound query statement by providing an array of strings
  - Query syntax maps to EJB Query Language as closely as possible
- Get Data Graph
  - Data Graphs can be constructed without activating EJBs
  - Queries can be specified to search only EJBs that are activated or in the cache

This slide and the next highlights several design points for the EJB Mediator.

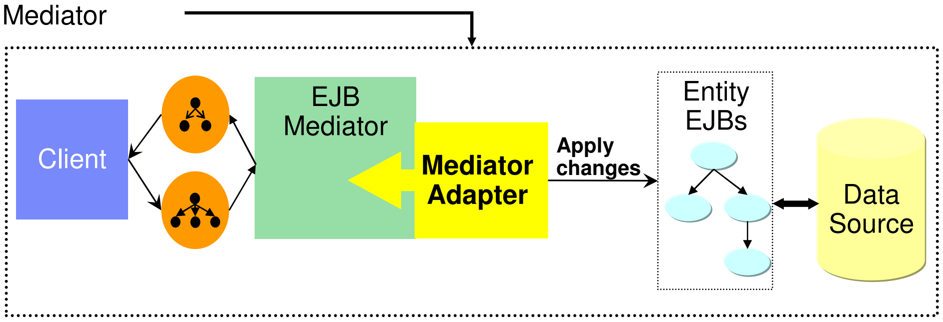
When creating an instance of the EJB Mediator, the client needs to specify a compound query statement. This is done by providing an array of strings that represent the individual queries that make up the compound EJB query statement. It is important to note that every effort has been made to map the syntax of the compound query statement passed in to create the EJB mediator to the EJB Query Language. However, there are several modifications included to support various functions of the EJB Mediator. An example where the EJB QL syntax has been augmented is with respect to using the EJB mediator to only query EJBs from a user supplied collection of activated EJBs.

One of the design features of the EJB Mediator is that for the get Data Graph function the client has a choice to have the data graph constructed without activating the EJBs. In this case, the mediator accesses the data store directly to build the data graph rather than activating the EJBs. In addition to this, the client can specify to the EJB Mediator only to search EJBs that are currently activated or are in the cache. The advantage to these features is that avoiding EJB activation can improve performance.



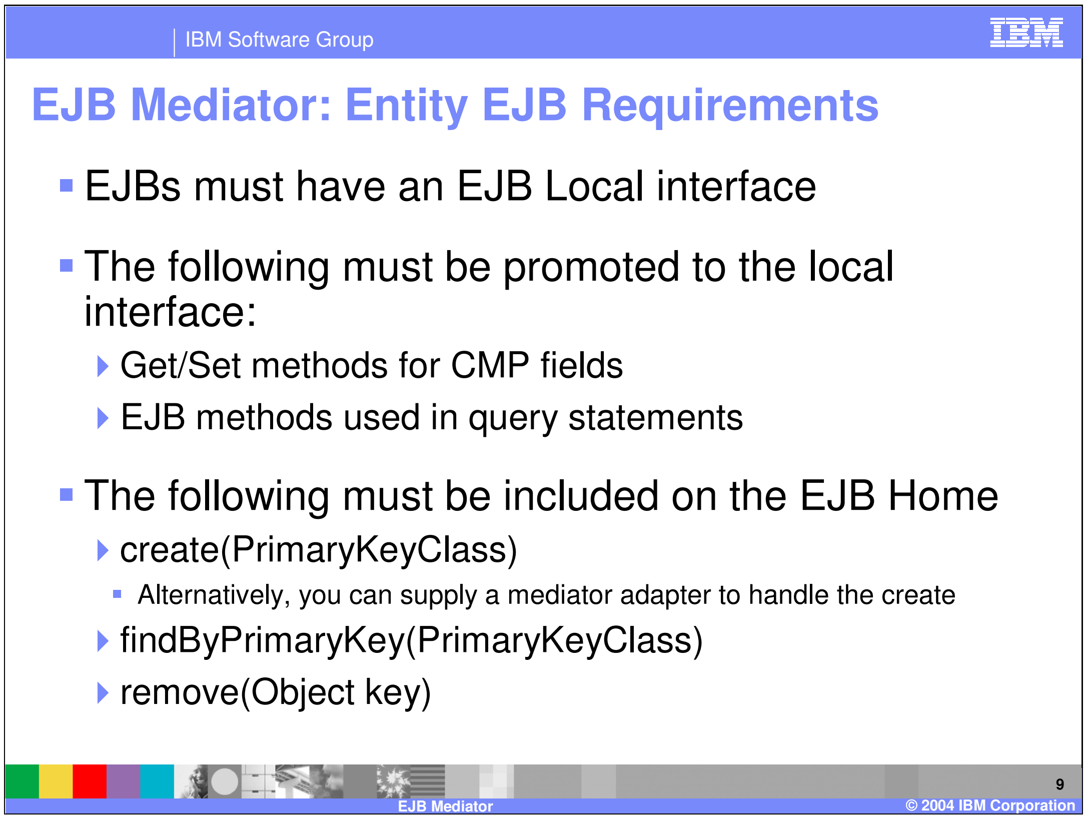
EJB Mediator: Design Points (Continued)

- Apply Changes
  - Several choices for updating data objects
    - Changes translated to SQL and data source updated directly by EJB Mediator
    - Changes written back through EJB by EJB Mediator
    - Client may specify mediator adapter to handle updates on behalf of EJB Mediator



When changes to the data graph are written back to the data store, there are several choices. The first choice is to let the EJB mediator handle the updates completely. In this case, the changes are translated to SQL and the data source is updated directly. The next choice is to have the mediator write back the changes through the EJBs. And finally, the last choice allows the client to provide a mediator adapter that allows the client to handle the updates on behalf of the mediator. These updates are written back through the EJBs. This is a pluggable interface that is implemented by the client to handle the updates.

Note: The mediator adapter can handle creates, updates, and deletes on behalf of the mediator. For more information on the mediator adapter see the com.ibm.websphere.sdo.mediator.ejb.MediatorAdapter interface.

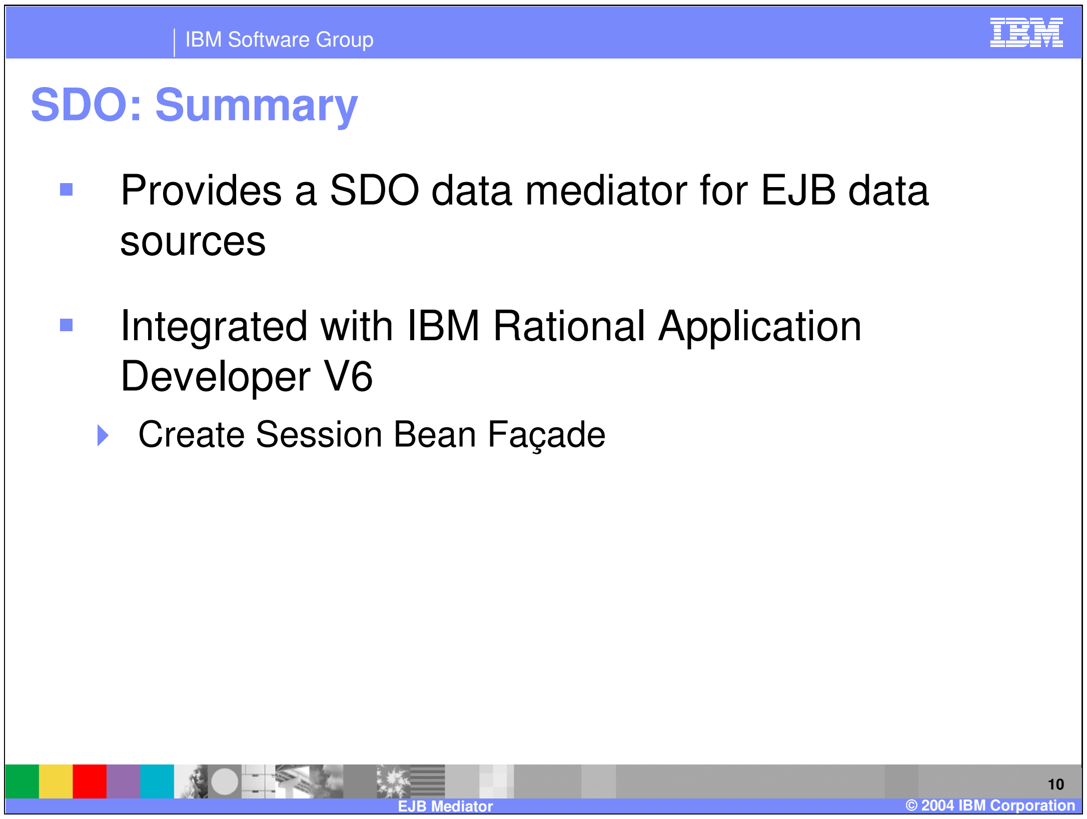


EJB Mediator: Entity EJB Requirements

- EJBs must have an EJB Local interface
- The following must be promoted to the local interface:
  - Get/Set methods for CMP fields
  - EJB methods used in query statements
- The following must be included on the EJB Home
  - create(PrimaryKeyClass)
    - Alternatively, you can supply a mediator adapter to handle the create
  - findByPrimaryKey(PrimaryKeyClass)
  - remove(Object key)

Because the EJB Mediator uses local interfaces for EJBs there are several important points to remember when using this mediator. First, getter and setter methods for CMP fields or EJB methods included in compound query statements used by the mediator must be promoted to the local interface.

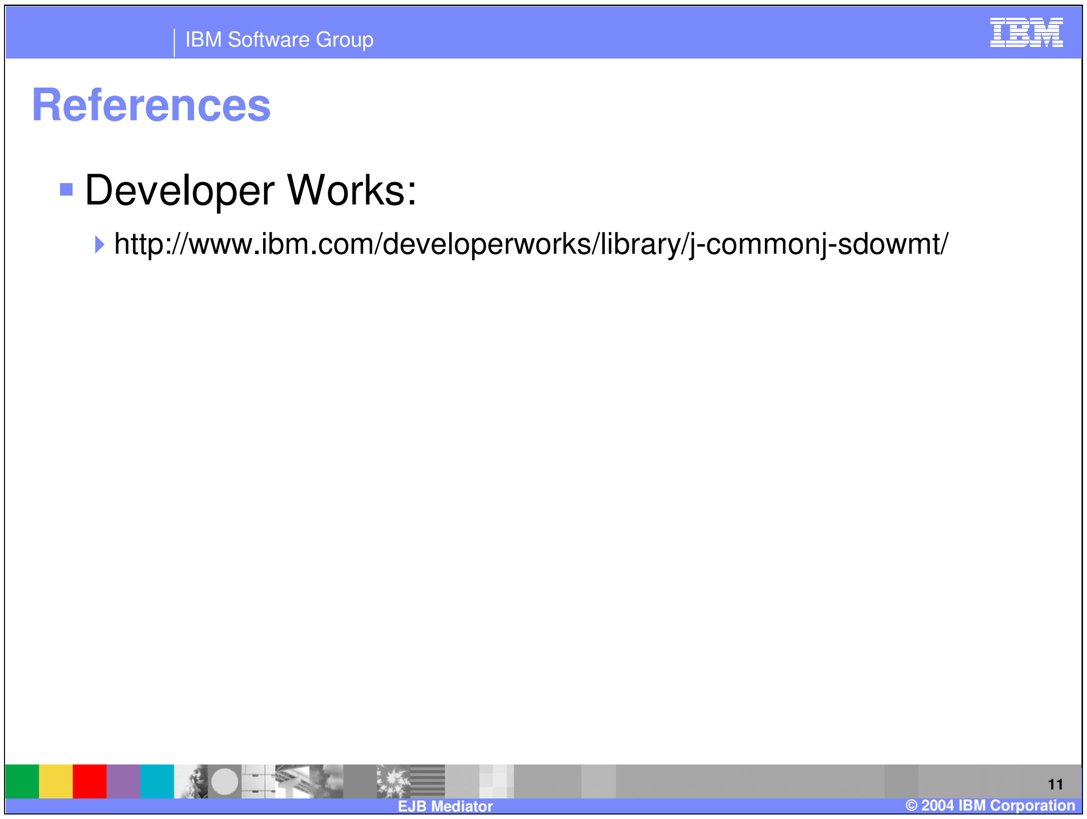
Also, if the EJB Mediator is used by the client to create an EJB, the client must be sure that the create(Primary_Key) method is included on the EJB Home. Alternatively, if this is not the case, you may supply a mediator adapter to handle the EJB create on behalf of the EJB Mediator.



SDO: Summary

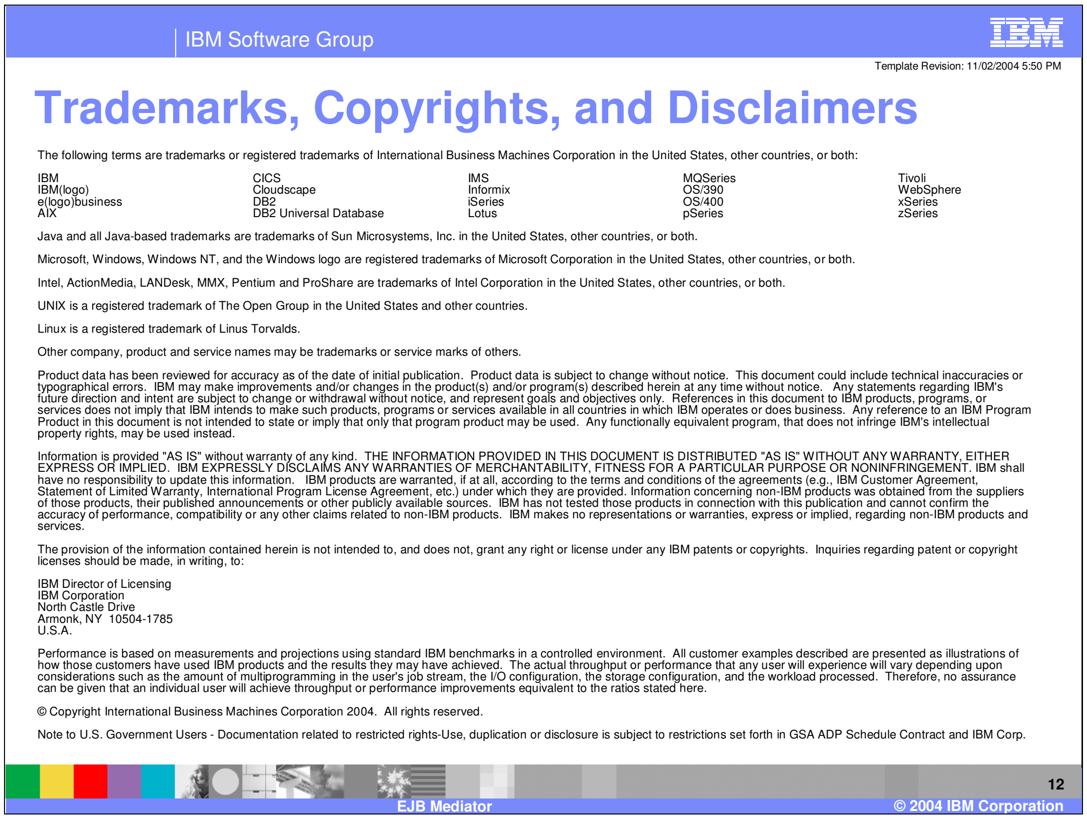
- Provides a SDO data mediator for EJB data sources
- Integrated with IBM Rational Application Developer V6
  - Create Session Bean Façade

In this presentation you learned about the SDO-based EJB Mediator. This technology is a new feature in the WebSphere Application Server V6 runtime environment and is also supported in IBM Rational Application Developer V6. Specifically, the EJB Mediator functionality is available as the Create Session Bean Façade from the context menu for Entity EJBs. The resulting session bean can be added to a JSF page from a Session Bean data source from the Page Data view.



References

- Developer Works:
  - http://www.ibm.com/developerworks/library/j-commonj-sdowmt/



Template Revision: 11/02/2004 5:50 PM

Trademarks, Copyrights, and Disclaimers