IBM

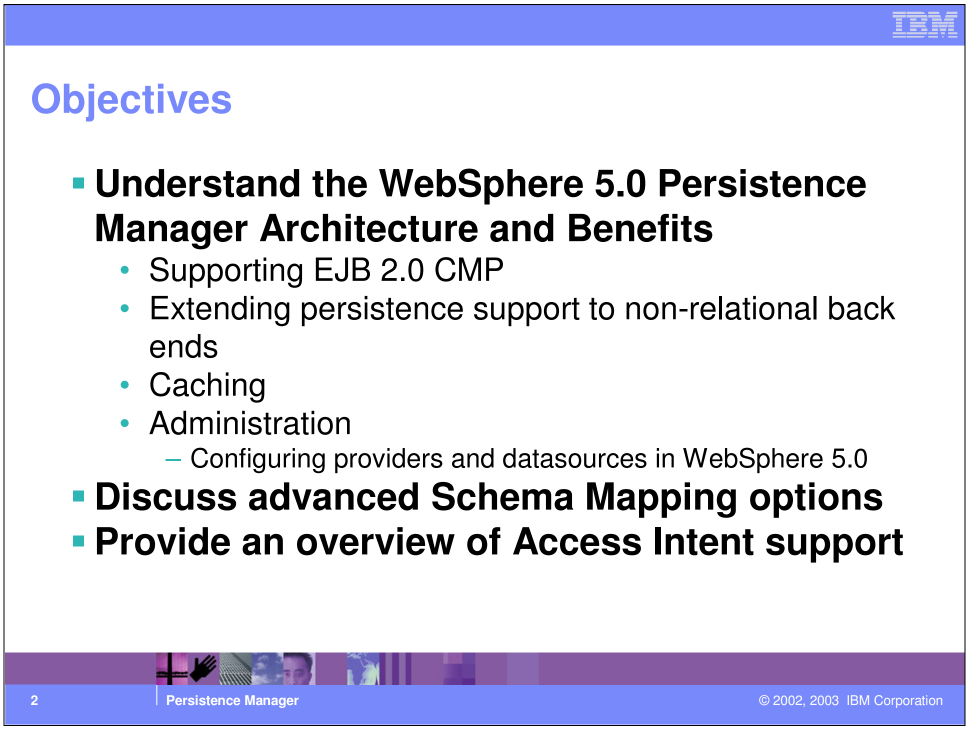
WebSphere Version 5.0

Persistence Manager

WebSphere. software

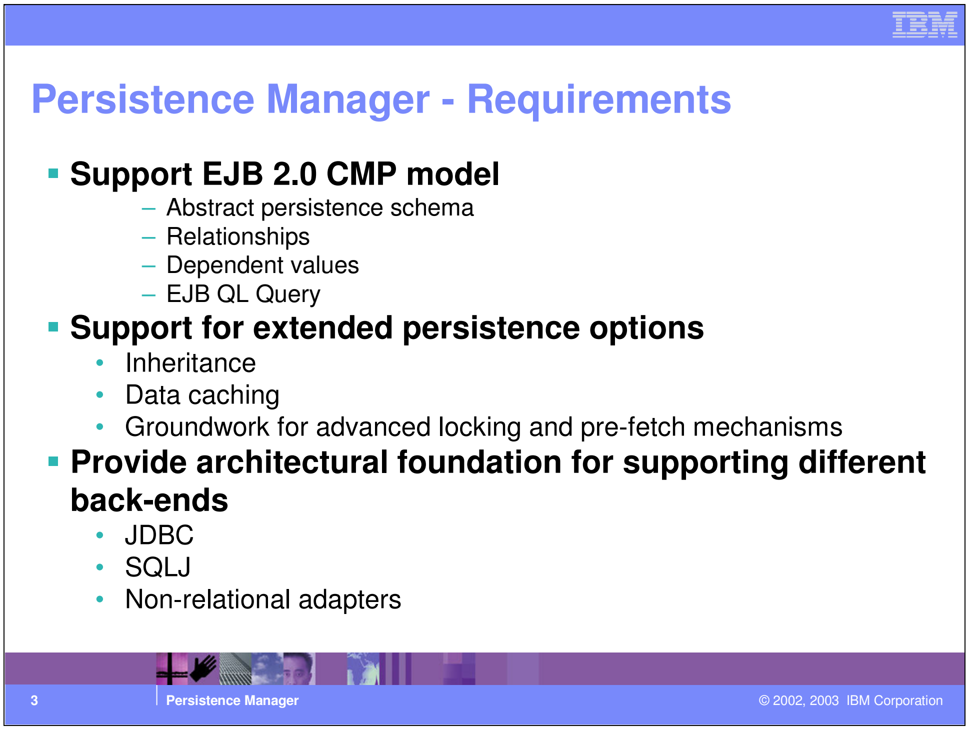
e business software

Updated 5/5/2003



Objectives

- **Understand the WebSphere 5.0 Persistence Manager Architecture and Benefits**
  - Supporting EJB 2.0 CMP
  - Extending persistence support to non-relational back ends
  - Caching
  - Administration
    - Configuring providers and datasources in WebSphere 5.0
- **Discuss advanced Schema Mapping options**
- **Provide an overview of Access Intent support**



Persistence Manager - Requirements

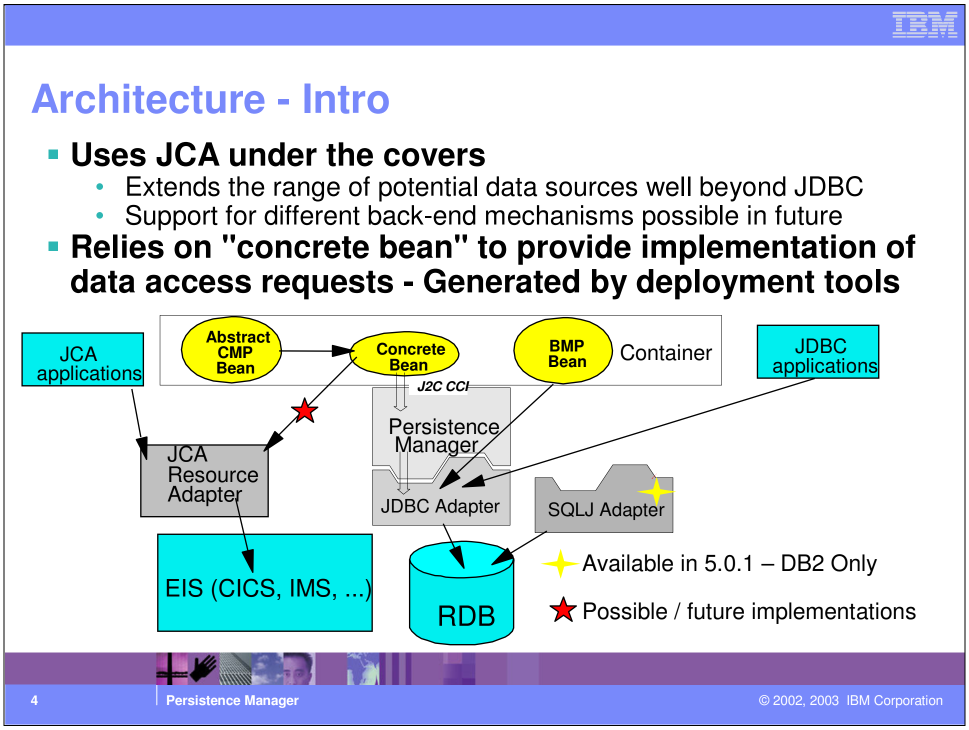
- **Support EJB 2.0 CMP model**
  - Abstract persistence schema
  - Relationships
  - Dependent values
  - EJB QL Query
- **Support for extended persistence options**
  - Inheritance
  - Data caching
  - Groundwork for advanced locking and pre-fetch mechanisms
- **Provide architectural foundation for supporting different back-ends**
  - JDBC
  - SQLJ
  - Non-relational adapters

The Persistence Manager in WebSphere 5.0 has a variety of objectives:

Support EJB 2.0 persistence which implies a series of responsibilities, such as implementing an abstract persistence schema, managing relationships and ensuring referential integrity, managing and persisting dependent values, and so on.

Support for "value add" persistence options such as handling inheritance, EJB data caching, pre-fetch mechanisms, and access intent.

In addition, the persistence manager wants to provide a new more open architecture that allows a wider variety of back ends, not limited to JDBC and possibly (in the future) even non-relational adapters.



Architecture - Intro

- **Uses JCA under the covers**
  - Extends the range of potential data sources well beyond JDBC
  - Support for different back-end mechanisms possible in future
- **Relies on "concrete bean" to provide implementation of data access requests - Generated by deployment tools**

The JDBC adapter uses JDBC APIs to access database dynamically. WebSphere supports JDBC to the DB2 family, Oracle, Sybase, Informix, MS SQLServer, Cloudscape and SQL99 standard vendor data sources.

SQLJ offers the advantage of static SQL and compiled access plans over JDBC, where the access plan needs to be created at runtime. This is implemented in WAS 5.0.1 Distributed and in z/OS 5.0 for DB/2 only, not for other databases. Support for other databases is expected in a later release.

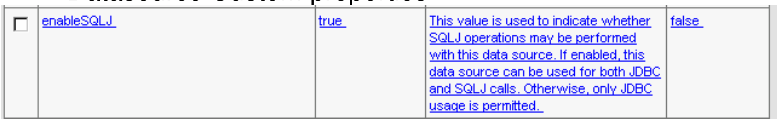
JCA Resource Adapters could be written to any number of Enterprise Information Systems (EIS), using the standard Java Connector Architecture (JCA). The appropriate code would then have to be written to generate the concrete bean implementation.

On the slide we have JCA Resource Adaptors to CICS and IMS. Two points to make here. These are not shipped as part of WAS, but are rather part of the CICS and IMS products that can be used with WAS. Secondly, the ability to use these from a CMP bean is just an architectural possibility, and there is no such implementation available today.

5.0.1→

SQLJ Support

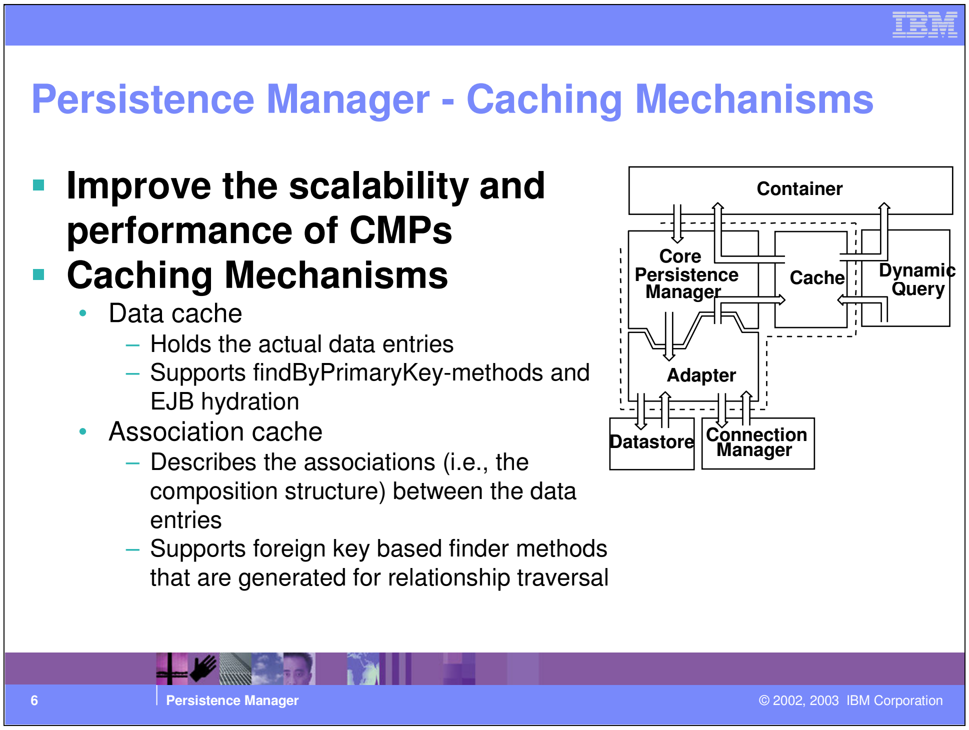
- **SQLJ advantages:**
  - Static SQL
  - Compiled access plans
    - JDBC access plan needs to be created at runtime
  - Performance
- **SQLJ Requirements:**
  - Configure DB2 Datasource to use SQLJ
    - Datasource Custom properties



□	enableSQLJ	true	This value is used to indicate whether SQLJ operations may be performed with this data source. If enabled, this data source can be used for both JDBC and SQLJ calls. Otherwise, only JDBC usage is permitted.	false

  - EAR file preparation
    - -sqlj flag on EJBDeploy command line tool

SQLJ offers static SQL, which is compiled in advance. This gives better runtime performance. To use SQLJ for EJB Persistence, two things are needed. The first is to configure the datasource to use SQLJ. The second is to configure the Enterprise Application to take advantage of SQLJ. This is done by generating EJB deployment code using the –sqlj option on the EJBDeploy command line tool.



Persistence Manager - Caching Mechanisms

- **Improve the scalability and performance of CMPs**
- **Caching Mechanisms**
  - Data cache
    - Holds the actual data entries
    - Supports findByPrimaryKey-methods and EJB hydration
  - Association cache
    - Describes the associations (i.e., the composition structure) between the data entries
    - Supports foreign key based finder methods that are generated for relationship traversal

Container

Core Persistence Manager

Cache

Dynamic Query

Adapter

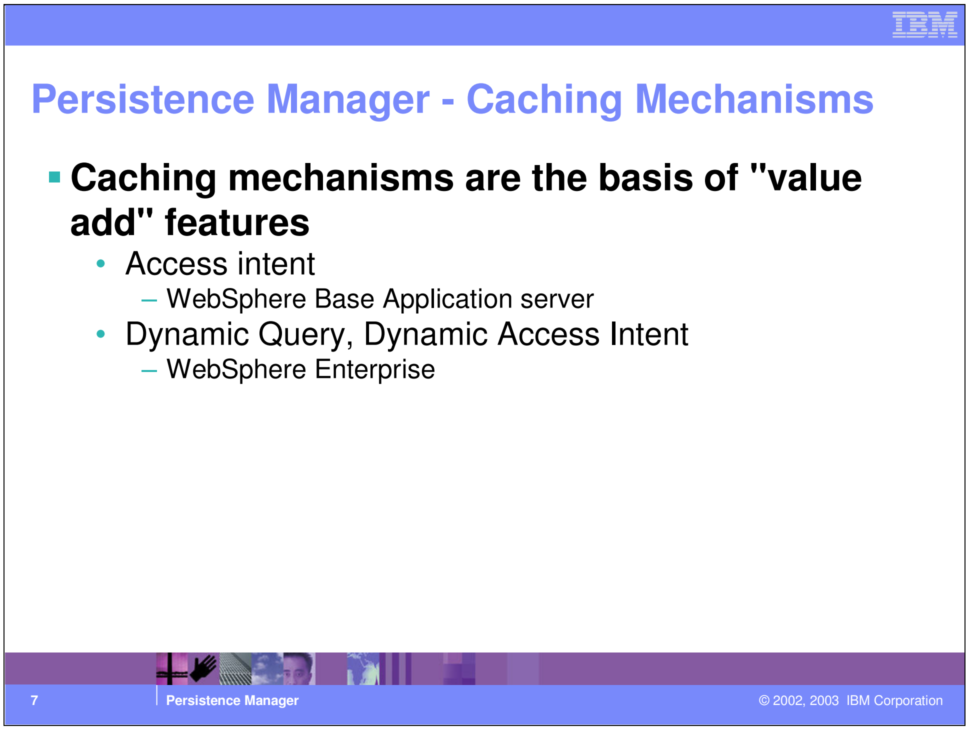
Datastore

Connection Manager

6 Persistence Manager

This chart complements the architectural view of the persistence manager. The PM is associated with a very rich caching implementation. There are three types of cache - besides an internal cache, which is used internally by the core persistence manager, each application server can rely on a high-level data cache where instances of the EJBs will actually be held - in order to minimize the physical database access.

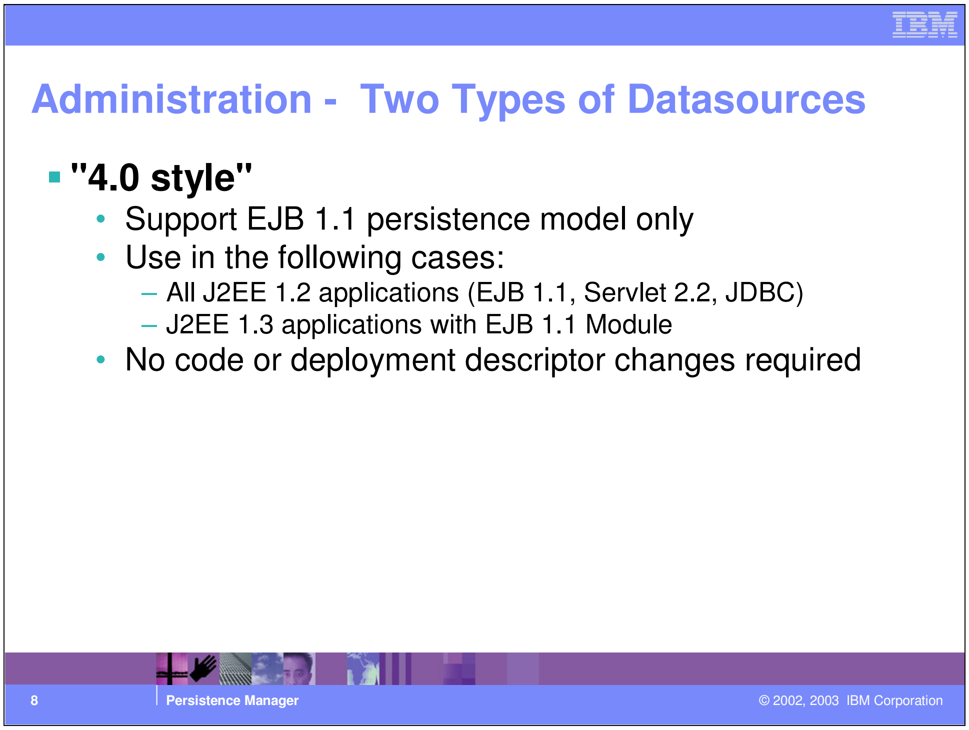
Also, the container can now cache the relationships so that we can optimize traversing a relationship tree.



Persistence Manager - Caching Mechanisms

- **Caching mechanisms are the basis of "value add" features**
  - Access intent
    - WebSphere Base Application server
  - Dynamic Query, Dynamic Access Intent
    - WebSphere Enterprise

This caching mechanisms are at the basis of "value add" features such as access intent, dynamic access intent, and dynamic query (the latter two provided as part of WebSphere Enterprise).

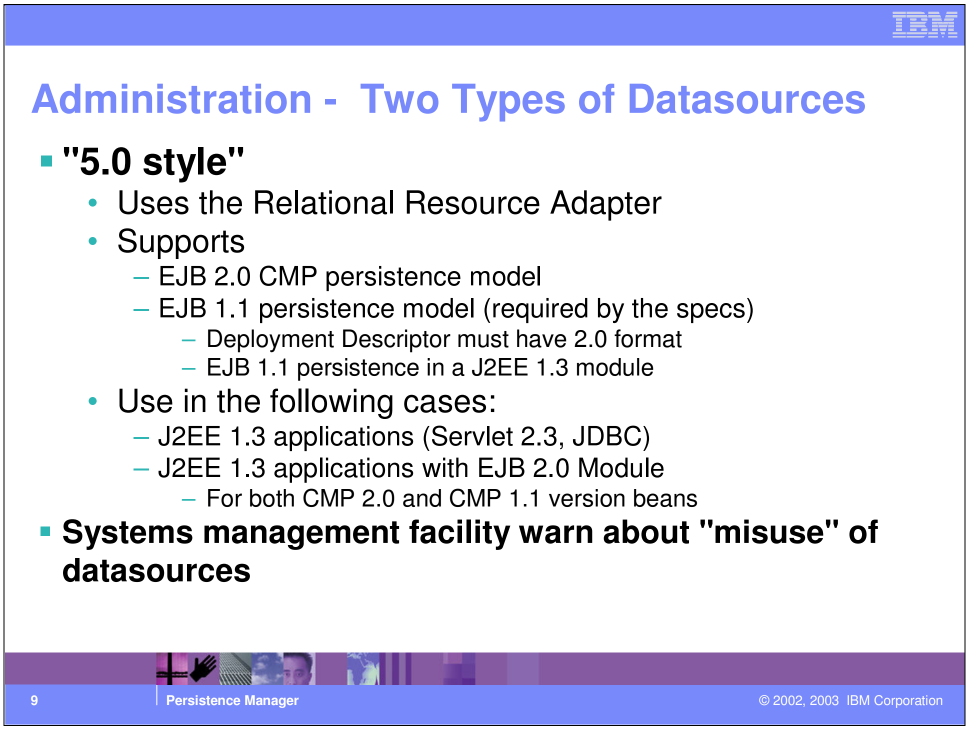


Administration - Two Types of Datasources

- **"4.0 style"**
  - Support EJB 1.1 persistence model only
  - Use in the following cases:
    - All J2EE 1.2 applications (EJB 1.1, Servlet 2.2, JDBC)
    - J2EE 1.3 applications with EJB 1.1 Module
  - No code or deployment descriptor changes required

WebSphere 5.0 offers two different types of datasources.

The 4.0 datasources are a carry over from the previous releases and only support EJB 1.1 persistence. They can be used with existing EJBs, so that you can run them as they are with no changes or need for redeployment. These datasources do not make use of the new architectural principles we outlined in the previous chart.

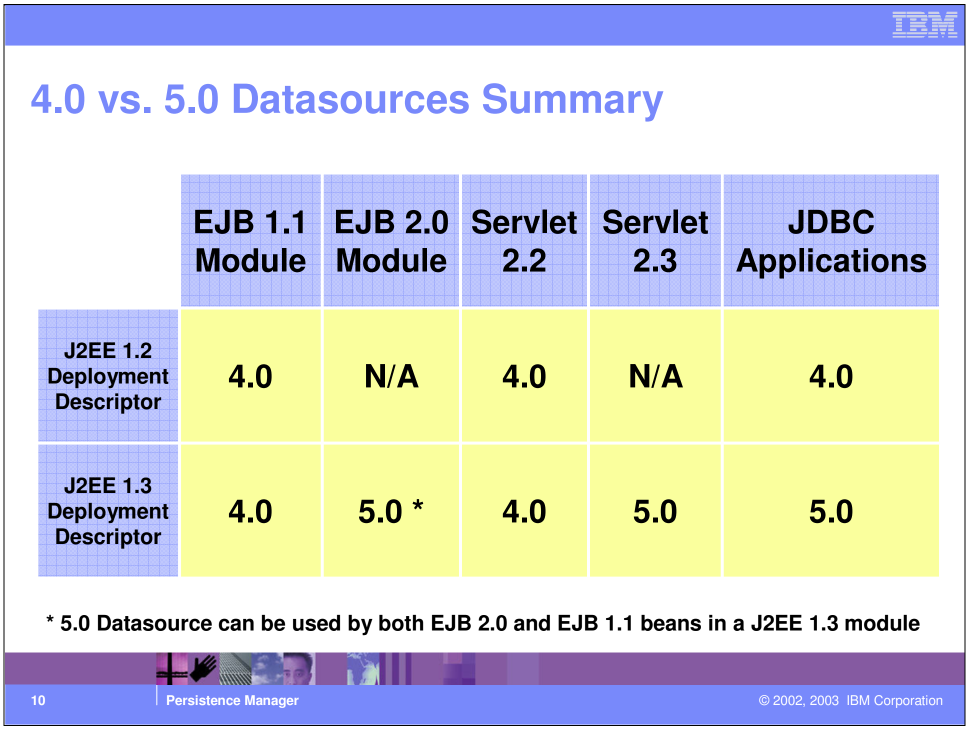


Administration - Two Types of Datasources

- **"5.0 style"**
  - Uses the Relational Resource Adapter
  - Supports
    - EJB 2.0 CMP persistence model
    - EJB 1.1 persistence model (required by the specs)
      - Deployment Descriptor must have 2.0 format
      - EJB 1.1 persistence in a J2EE 1.3 module
  - Use in the following cases:
    - J2EE 1.3 applications (Servlet 2.3, JDBC)
    - J2EE 1.3 applications with EJB 2.0 Module
      - For both CMP 2.0 and CMP 1.1 version beans
- **Systems management facility warn about "misuse" of datasources**

9 Persistence Manager

The 5.0 datasources are the ones that utilize the Relational Resource Adapter. They support both 2.0 and 1.1 persistence. However - they can be used only with EJB modules that have a 2.0 deployment descriptor. You can include EJBs with EJB 1.1 persistence in an EJB 2.0 module (the specifications support both persistence models).



4.0 vs. 5.0 Datasources Summary

	EJB 1.1 Module	EJB 2.0 Module	Servlet 2.2	Servlet 2.3	JDBC Applications
J2EE 1.2 Deployment Descriptor	4.0	N/A	4.0	N/A	4.0
J2EE 1.3 Deployment Descriptor	4.0	5.0 *	4.0	5.0	5.0

* 5.0 Datasource can be used by both EJB 2.0 and EJB 1.1 beans in a J2EE 1.3 module

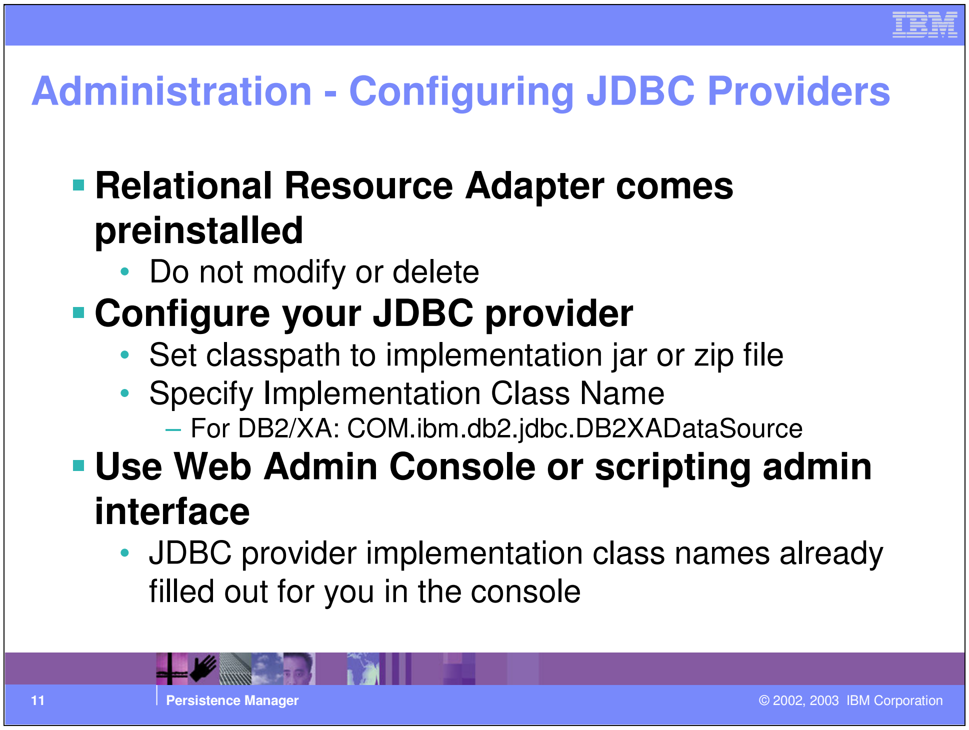
This chart outlines the possible combinations of specification APIs and datasources.

A J2EE 1.2 application module (which necessarily has EJB 1.1 persistence) needs to use the "old" WebSphere 4.0 datasources (in other words, applications carried over "as is" from WebSphere 4.0 will use these datasources).

A J2EE 1.3 application may contain both EJB 1.1 and EJB 2.0 persistence.

For EJB 2.0 persistence there is only one choice - the new datasources.

For EJB 1.1 persistence inside a J2EE 1.3 module you may choose between the 4.0 and the 5.0 datasources - although the latter are recommended (performance, caching, etc.).

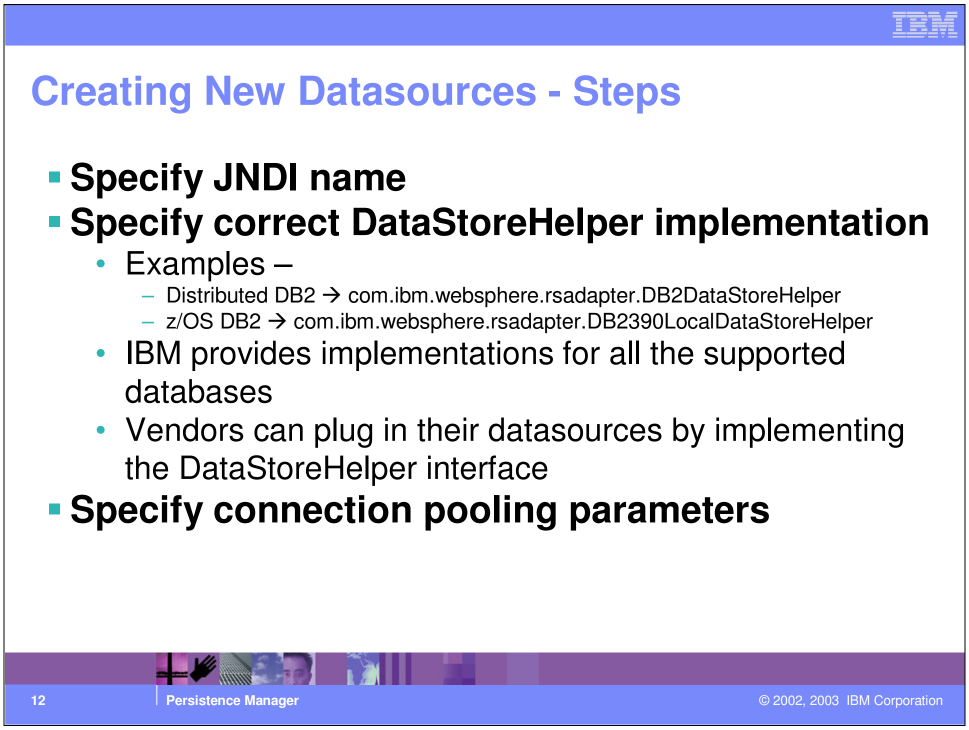


Administration - Configuring JDBC Providers

- **Relational Resource Adapter comes preinstalled**
  - Do not modify or delete
- **Configure your JDBC provider**
  - Set classpath to implementation jar or zip file
  - Specify Implementation Class Name
    - For DB2/XA: COM.ibm.db2.jdbc.DB2XADataSource
- **Use Web Admin Console or scripting admin interface**
  - JDBC provider implementation class names already filled out for you in the console

11 Persistence Manager © 2002, 2003 IBM Corporation

If you use the admin console to define your JDBC Provider, most of the parameters that are needed for the JDBC provider are filled out automatically by the GUI or available in drop-down lists.



Creating New Datasources - Steps

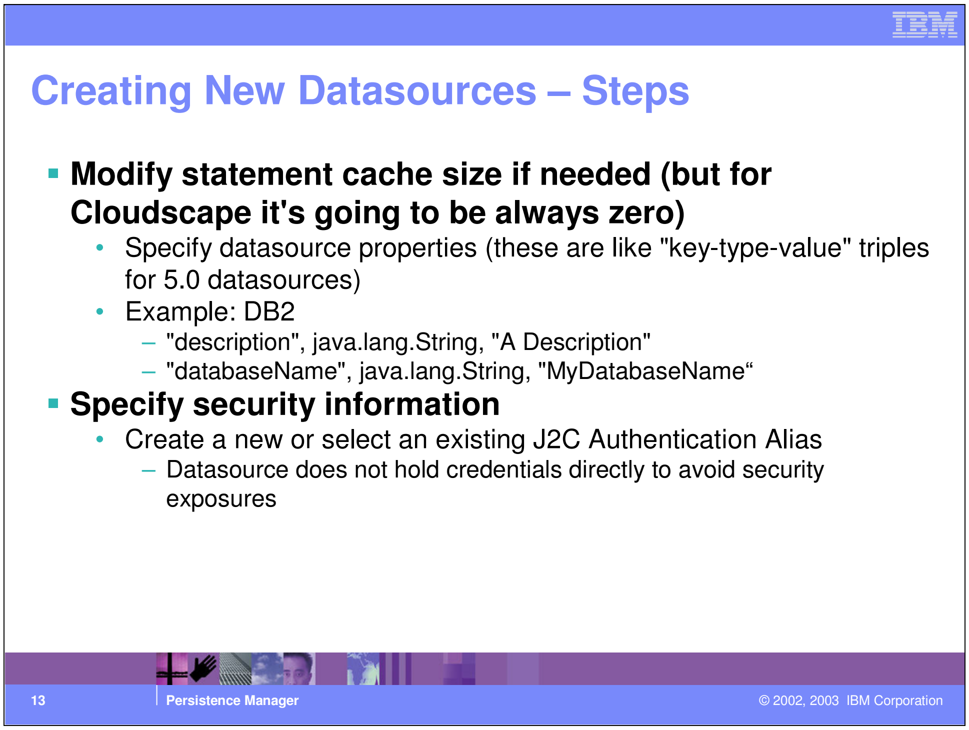
- **Specify JNDI name**
- **Specify correct DataStoreHelper implementation**
  - Examples –
    - Distributed DB2 → com.ibm.websphere.rsadapter.DB2DataStoreHelper
    - z/OS DB2 → com.ibm.websphere.rsadapter.DB2390LocalDataStoreHelper
  - IBM provides implementations for all the supported databases
  - Vendors can plug in their datasources by implementing the DataStoreHelper interface
- **Specify connection pooling parameters**

12 Persistence Manager

This chart describes the steps needed to configure a "new" data source (5.0 style). Most of the parameters (like the datastore implementer, etc.) are already filled out by the admin console as you try to create the datasource.

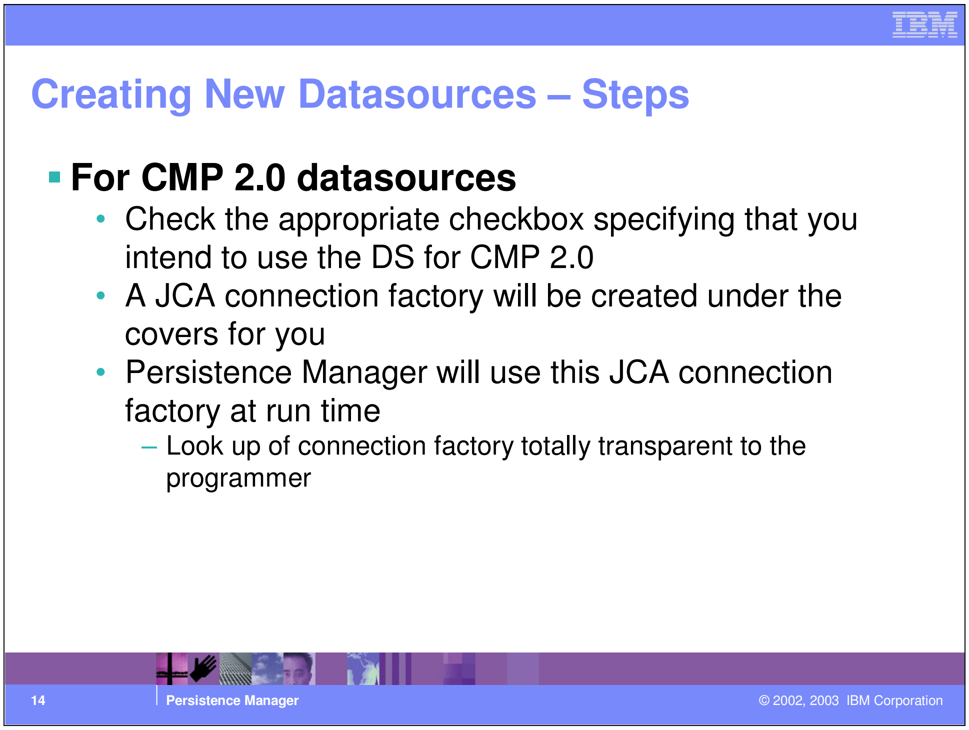
Some of the most commonly used properties are also predefined - you may just need to overwrite them.

As you create a datasource that uses CMP 2.0 you must make sure that you check the appropriate checkbox in the admin console - the systems management facilities will then create a new connection factory - which is needed, since the new datasources use JCA to get to the relational database.



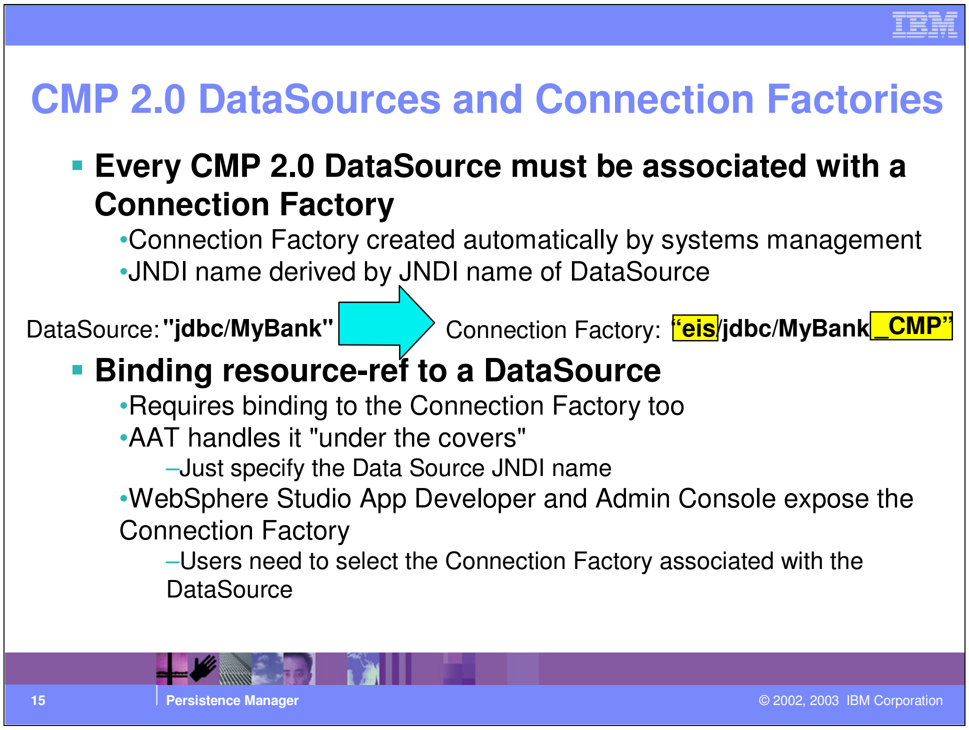
Creating New Datasources – Steps

- **Modify statement cache size if needed (but for Cloudscape it's going to be always zero)**
  - Specify datasource properties (these are like "key-type-value" triples for 5.0 datasources)
  - Example: DB2
    - "description", java.lang.String, "A Description"
    - "databaseName", java.lang.String, "MyDatabaseName“
- **Specify security information**
  - Create a new or select an existing J2C Authentication Alias
    - Datasource does not hold credentials directly to avoid security exposures



Creating New Datasources – Steps

- **For CMP 2.0 datasources**
  - Check the appropriate checkbox specifying that you intend to use the DS for CMP 2.0
  - A JCA connection factory will be created under the covers for you
  - Persistence Manager will use this JCA connection factory at run time
    - Look up of connection factory totally transparent to the programmer



CMP 2.0 DataSources and Connection Factories

- **Every CMP 2.0 DataSource must be associated with a Connection Factory**
  - Connection Factory created automatically by systems management
  - JNDI name derived by JNDI name of DataSource

DataSource: **"jdbc/MyBank"** → Connection Factory: **“eis/jdbc/MyBank_CMP”**

- **Binding resource-ref to a DataSource**
  - Requires binding to the Connection Factory too
  - AAT handles it "under the covers"
    - Just specify the Data Source JNDI name
  - WebSphere Studio App Developer and Admin Console expose the Connection Factory
    - Users need to select the Connection Factory associated with the DataSource

In order to be usable, CMP 2.0 datasources must be associated with a JCA Connection Factory. The JNDI name of the connection factory is derived from the name of the datasource by prepending "eis/" and appending "_CMP".

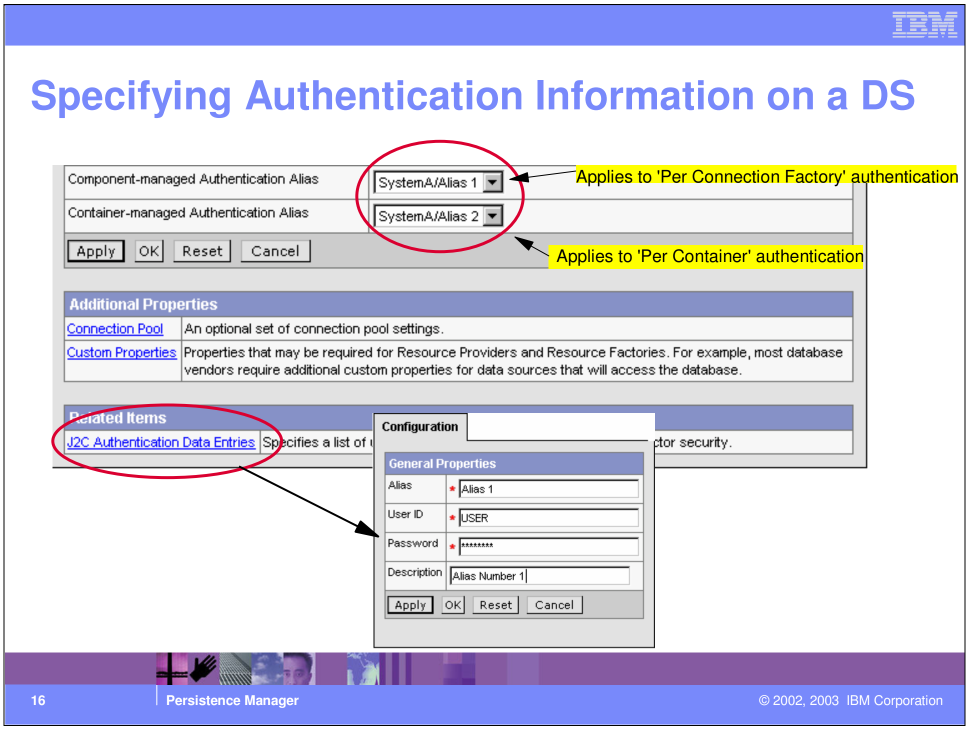
The administrative console automatically creates these Connection Factories for you - but only if you explicitly select the "Use for CMP 2.0" checkbox at creation time.

You need to make sure that when you bind to a datasource (for example, when you specify the datasource to use for a CMP Entity EJB) ultimately you will bind your component to the connection factory.

The AAT can do the translation for you. You can specify the JNDI name of the datasource for the EJB Entity CMP beans and the AAT will derive the CF name.

Other interfaces, like WebSphere Studio App Developer, or the admin console, still expose the connection factory JNDI name.

If the binding is specified incorrectly, or if the Connection Factory wasn't created appropriately, you will get a NamingException at runtime.



Specifying Authentication Information on a DS

Component-managed Authentication Alias	SystemA/Alias 1
Container-managed Authentication Alias	SystemA/Alias 2

Applies to 'Per Connection Factory' authentication

Applies to 'Per Container' authentication

Additional Properties

Connection Pool	An optional set of connection pool settings.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Related Items

J2C Authentication Data Entries	Specifies a list of ctor security.

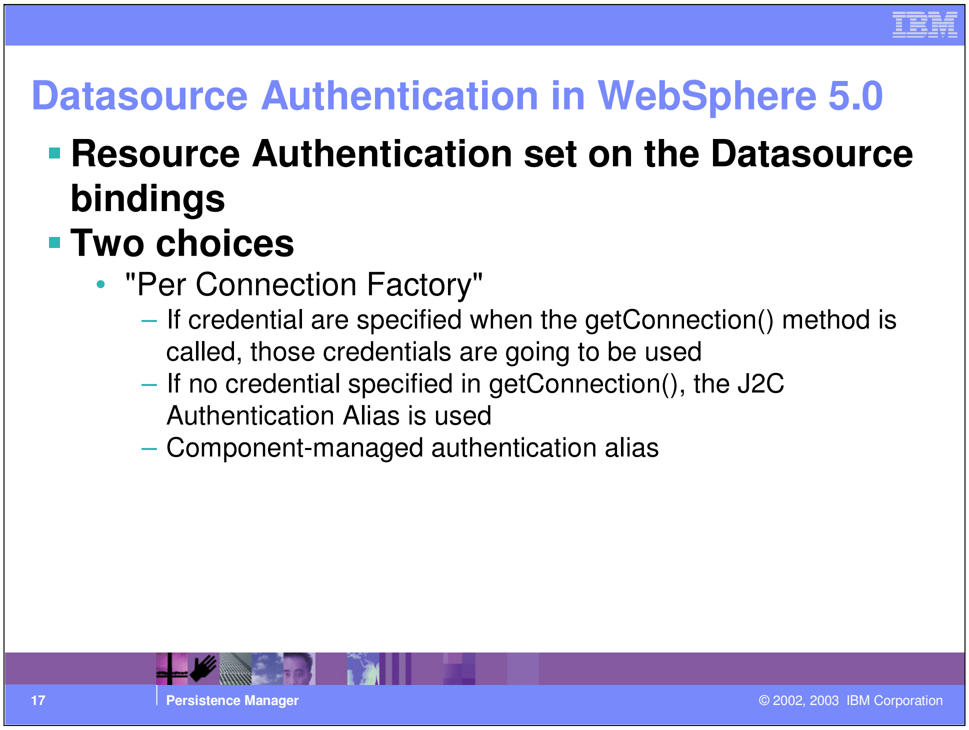
Configuration

General Properties

Alias	* Alias 1
User ID	* USER
Password	* ********
Description	Alias Number 1

Datasources do not hold credentials directly in their configuration information. They rather refer to an object that is stored in the security.xml file. Being able to look up a datasource doesn't guarantee the ability to establish a connection. You could use Java 2 Security to limit the access rights to the security information so that only certain applications or certain clients would be granted access.

There are two aliases that you can specify - the "Component Managed authentication alias" will be used for "Per Connection Factory" authentication and the "Container-managed authentication alias" will be used in the "Per Container" authentication.



Datasource Authentication in WebSphere 5.0

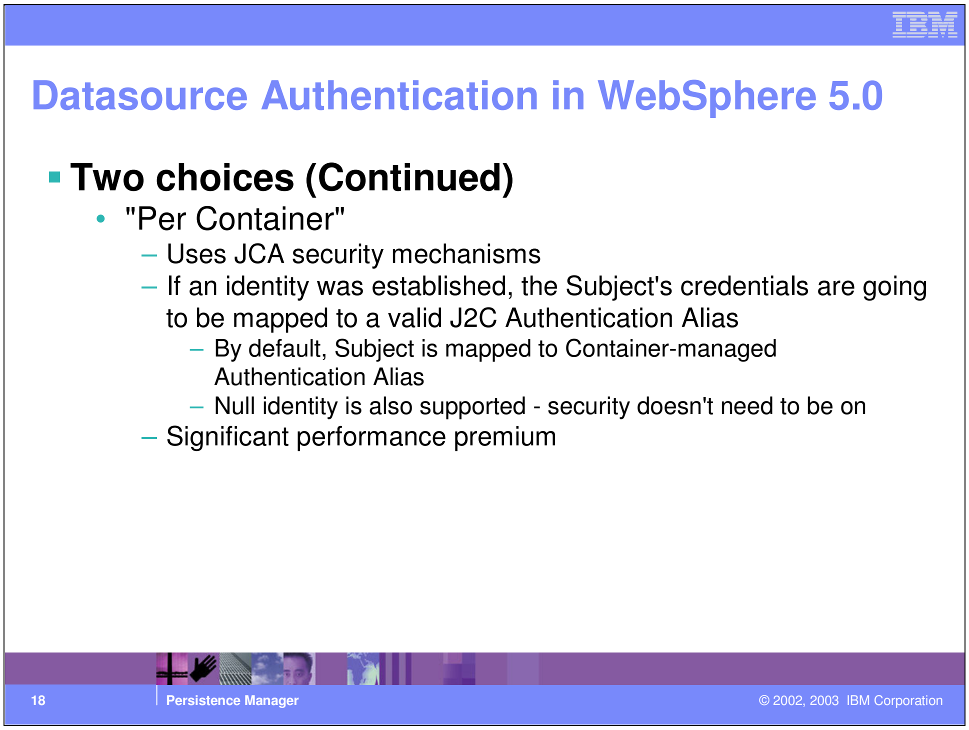
- **Resource Authentication set on the Datasource bindings**
- **Two choices**
  - "Per Connection Factory"
    - If credential are specified when the getConnection() method is called, those credentials are going to be used
    - If no credential specified in getConnection(), the J2C Authentication Alias is used
    - Component-managed authentication alias

17 Persistence Manager © 2002, 2003 IBM Corporation

There are two possible choices for the resource authentication in WebSphere 5.0 that are applicable to datasources.

"Per Connection Factory" (also called Per Resource in some contexts) specifies that if a user and password were explicitly set when calling the getConnection() method, those credentials will be used. If the getConnection method on the datasource is called with no parameters, the authentication will take place based on the user and password specified in the J2C Authentication Alias that was specified when configuring the datasource. The "Component-managed authentication alias" will be utilized.

With CMP EJBs, the getConnection() is always called with no parameters - no identity is passed with the connection and the Alias is always needed if the resource authentication method is of type "Per Connection Factory" .

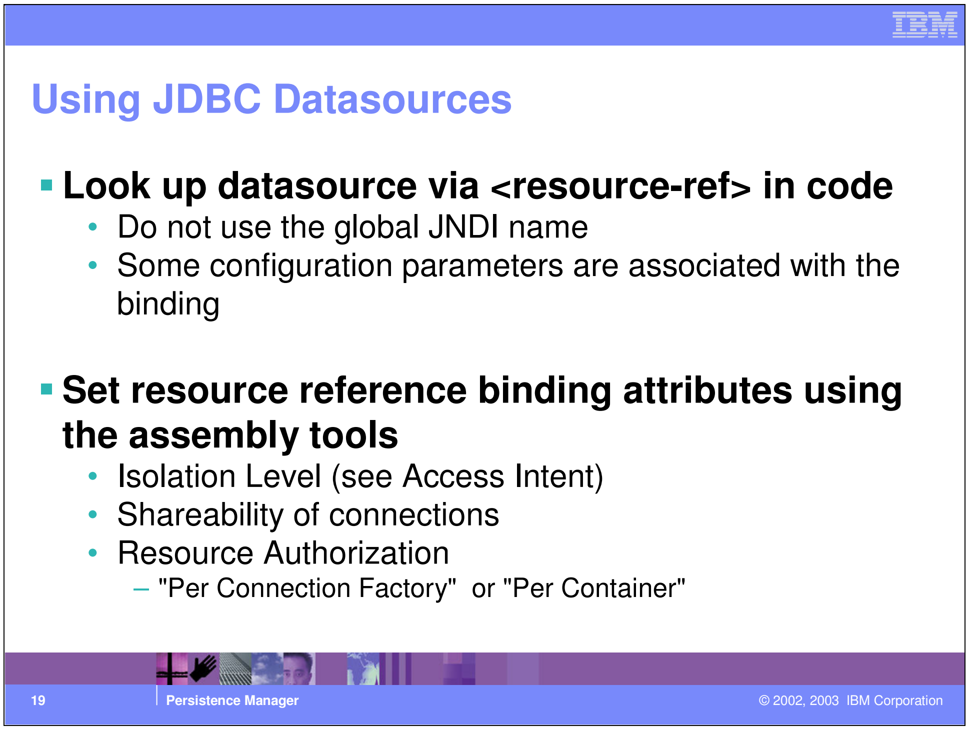


Datasource Authentication in WebSphere 5.0

- **Two choices (Continued)**
  - "Per Container"
    - Uses JCA security mechanisms
    - If an identity was established, the Subject's credentials are going to be mapped to a valid J2C Authentication Alias
      - By default, Subject is mapped to Container-managed Authentication Alias
      - Null identity is also supported - security doesn't need to be on
    - Significant performance premium

The "Per Container" Authentication technique involves some more complex authentication mechanisms. If the program that obtains the connection has established a client identity, the Persistence Management layer will be retrieve the "Subject" and map it to a J2C Authentication alias. The J2C Authentication alias is then used to establish the JDBC connection. By default, all identities (including the null identity) are mapped to the Container-managed authentication alias configured in the datasource. You can also plug in your own J2C Login Module and establish a different mapping.

"Per Container" is more resource-intensive than "Per Connection Factory" although it allows for more flexibility. It should be used only when necessary due to the performance implications.



Using JDBC Datasources

- **Look up datasource via <resource-ref> in code**
  - Do not use the global JNDI name
  - Some configuration parameters are associated with the binding
- **Set resource reference binding attributes using the assembly tools**
  - Isolation Level (see Access Intent)
  - Shareability of connections
  - Resource Authorization
    - "Per Connection Factory" or "Per Container"

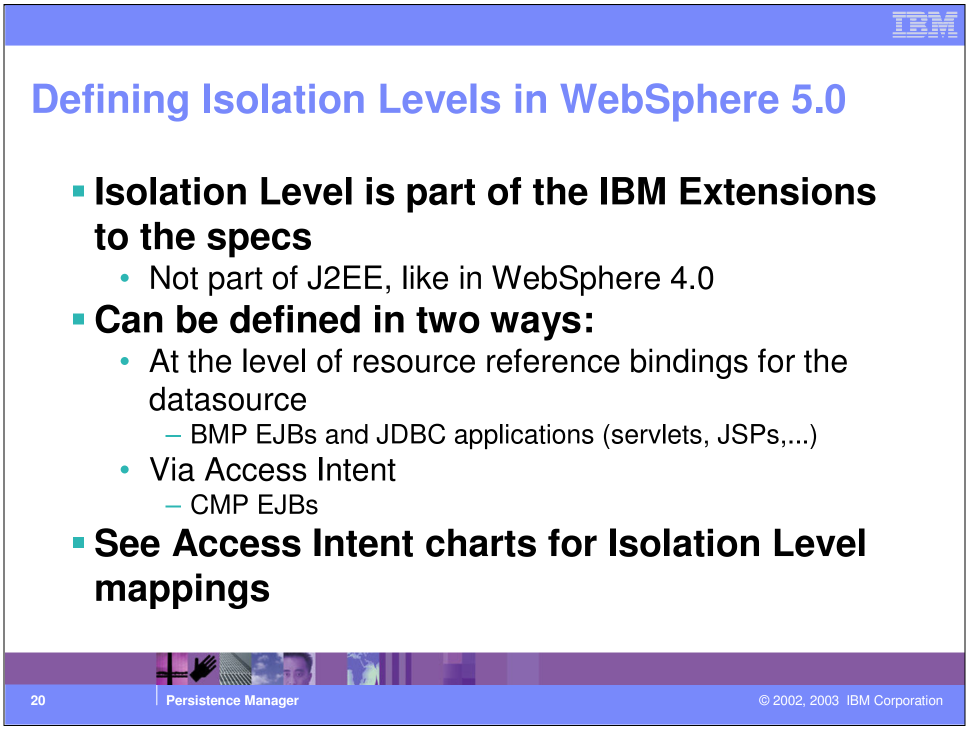
19 Persistence Manager

When you look up a datasource in your code (BMP EJB or servlet) you should be using the resource reference (java:comp/env/.....) rather than the global JNDI name. This is important in 5.0 - because some configuration options are associated to the binding (like for example the isolation level).

Conceivably, an application may look up the same datasource using two different references bound in two different ways. One reference may use a certain isolation level, another reference may be bound to the same datasource, but with a different isolation level.

The isolation level, the authorization mechanisms, and the shareability of the connections are determined by the binding.

If you use a direct lookup, no bindings will be involved in the process. In that case, WebSphere will assign a default set of values for the parameters mentioned above. A message is generated in the log file (systemout.log) mentioning the fact that a direct lookup was used.



Defining Isolation Levels in WebSphere 5.0

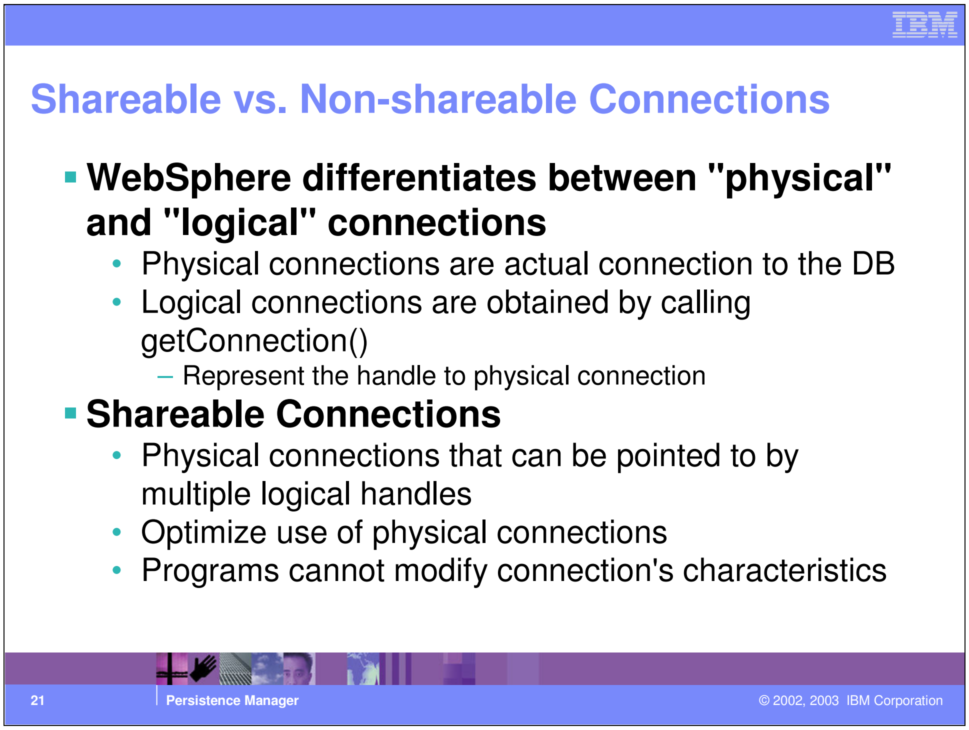
- **Isolation Level is part of the IBM Extensions to the specs**
  - Not part of J2EE, like in WebSphere 4.0
- **Can be defined in two ways:**
  - At the level of resource reference bindings for the datasource
    - BMP EJBs and JDBC applications (servlets, JSPs,...)
  - Via Access Intent
    - CMP EJBs
- **See Access Intent charts for Isolation Level mappings**

20 Persistence Manager

The isolation level in WebSphere 5.0 can be defined in two different ways.

For BMP and Web components, the isolation level can be defined at the datasource level - in fact, it is defined in the resource-reference bindings.

For CMPs, the isolation level needs to be defined through the Access Intent parameter - an IBM extension to the J2EE 1.3 deployment descriptors. The Access Intent is meant to encapsulate more than just the isolation level though.



Shareable vs. Non-shareable Connections

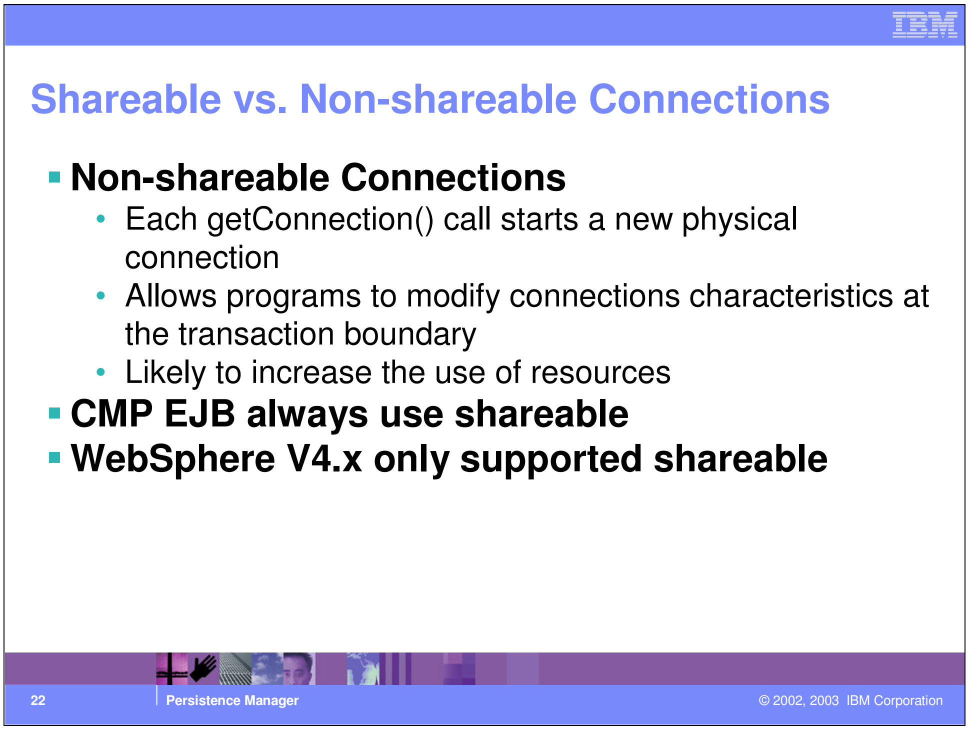
- **WebSphere differentiates between "physical" and "logical" connections**
  - Physical connections are actual connection to the DB
  - Logical connections are obtained by calling getConnection()
    - Represent the handle to physical connection
- **Shareable Connections**
  - Physical connections that can be pointed to by multiple logical handles
  - Optimize use of physical connections
  - Programs cannot modify connection's characteristics

21 Persistence Manager © 2002, 2003 IBM Corporation

In the resource-ref binding tab of JDBC connections, you can specify if you want to obtain a shareable or unshareable connection.

Shareable connections are physical connections that can be pointed to by a number of logical connections.

When the getConnection() method is called, WebSphere will determine whether an existing connection with the required characteristics already exists (isolation level and security settings must match). If it does, the getConnection() method will return a handle pointing to the existing connection.



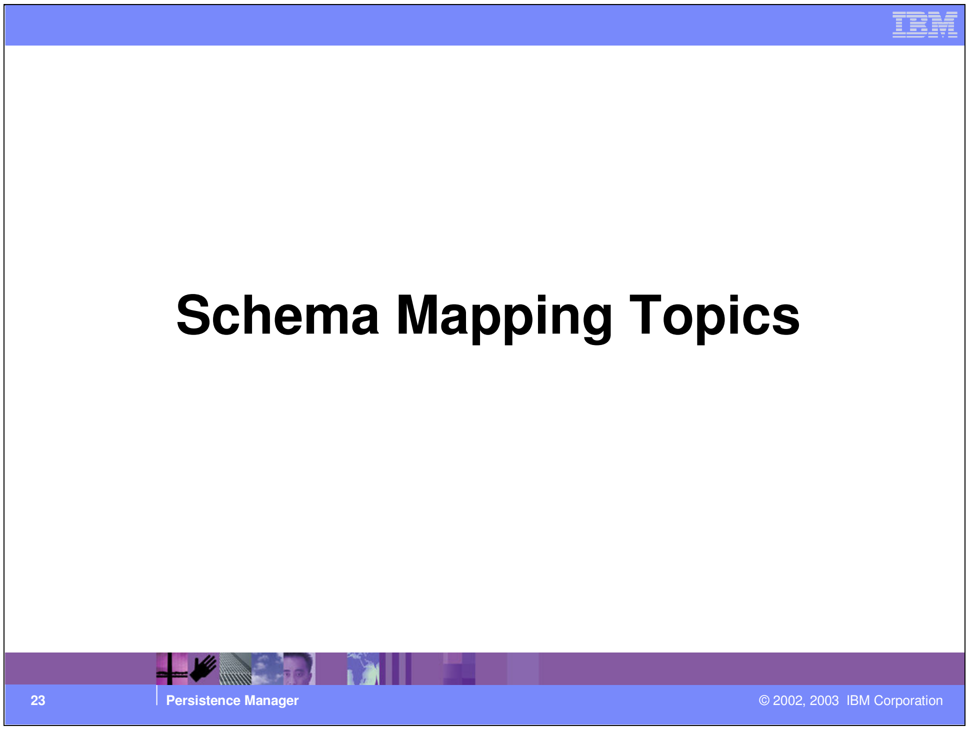
Shareable vs. Non-shareable Connections

- **Non-shareable Connections**
  - Each getConnection() call starts a new physical connection
  - Allows programs to modify connections characteristics at the transaction boundary
  - Likely to increase the use of resources
- **CMP EJB always use shareable**
- **WebSphere V4.x only supported shareable**

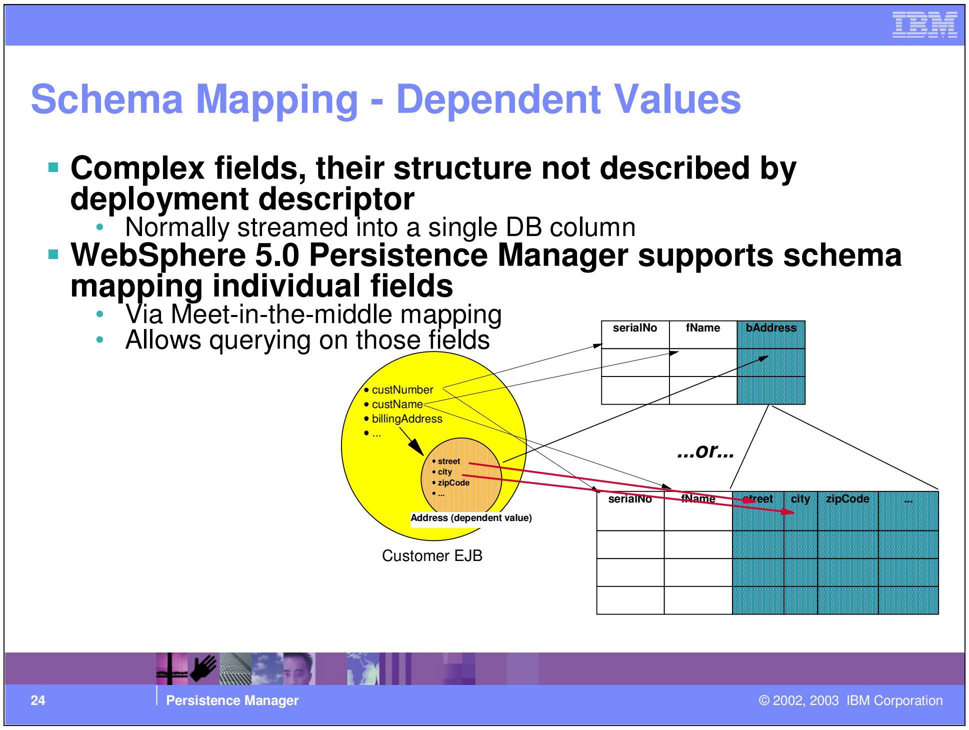
If non-shareable connections are specified in the bindings - we'll start a new physical connection with each call of getConnection().

The tradeoff between the two types is as follows: if you get a shareable connection, you won't be able to modify its characteristics (isolation level, security) once you have obtained it.

With non-shareable connections, you can change the connection's characteristics at the transaction boundaries, but you probably end up with a larger number of physical connections being created.



Schema Mapping Topics



Schema Mapping - Dependent Values

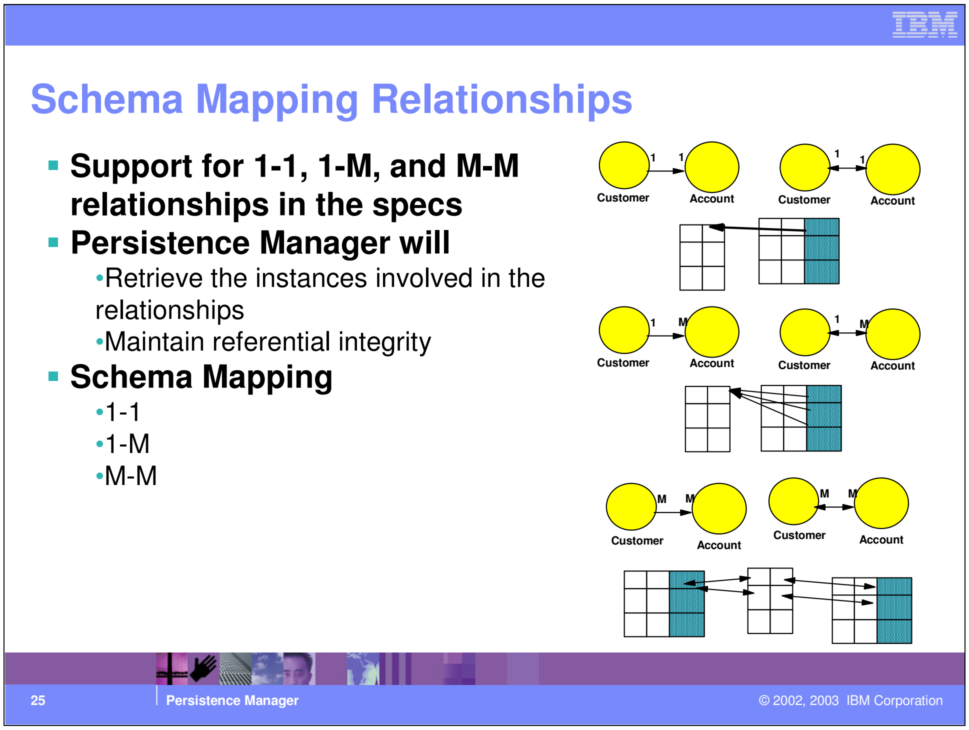
- **Complex fields, their structure not described by deployment descriptor**
  - Normally streamed into a single DB column
- **WebSphere 5.0 Persistence Manager supports schema mapping individual fields**
  - Via Meet-in-the-middle mapping
  - Allows querying on those fields

serialNo	fName	bAddress

serialNo	fName	street	city	zipCode	...

WebSphere schema mapping and EJB deployment tools support mapping each individual field of a dependent value in separate columns.

The default mapping will stream the dependent value in a single column - but using the meet-in-the-middle techniques you can map individual fields. The extended EJB QL query support for can then take advantage of this by allowing you to query on fields located inside a dependent.


1
1
Customer
Account
1
1
Customer
Account
1
M
Customer
Account
1
M
Customer
Account
M
M
Customer
Account
M
M
Customer
Account

Schema Mapping Relationships

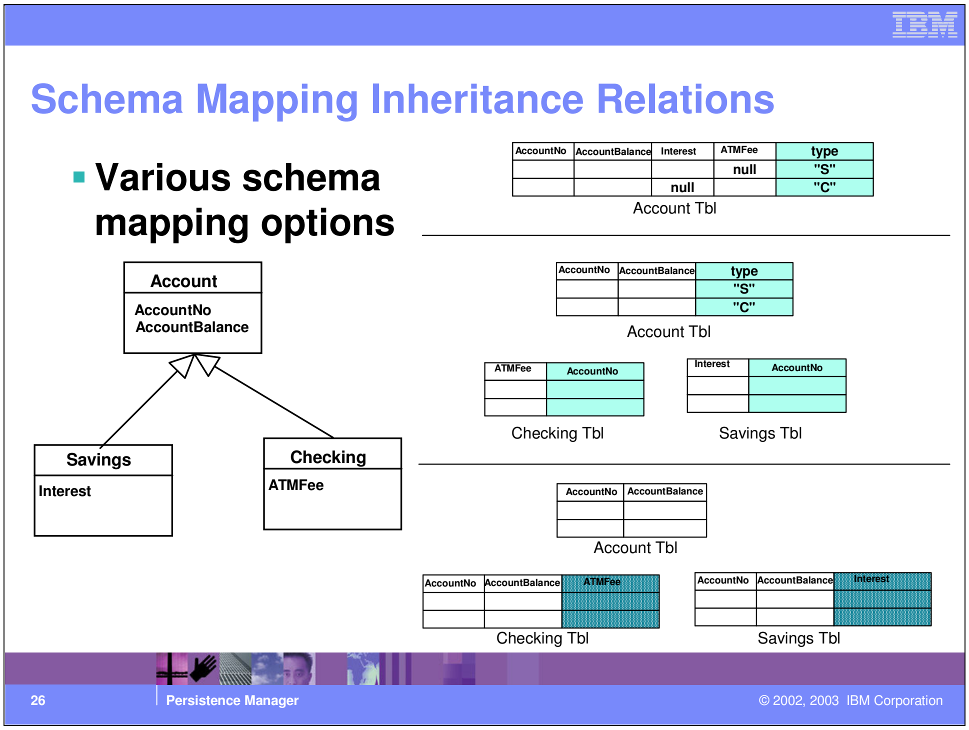
- **Support for 1-1, 1-M, and M-M relationships in the specs**
- **Persistence Manager will**
  - •Retrieve the instances involved in the relationships
  - •Maintain referential integrity
- **Schema Mapping**
  - •1-1
  - •1-M
  - •M-M

25 Persistence Manager © 2002, 2003 IBM Corporation

The chart shows various types of relationships and their correspondent schema mapping. Notice that the table layout is not affected by the uni- or bi-directional nature of the relationship.

In the case of a Many-to-Many, there is the need for a relationship table to maintain the information about the related objects.

The container will make sure that referential integrity is preserved when creating, deleting, or moving EJBs around.



Schema Mapping Inheritance Relations

- Various schema mapping options

Account
AccountNo
AccountBalance

Savings
Interest

Checking
ATMFee

AccountNo	AccountBalance	Interest	ATMFee	type
			null	"S"
		null		"C"

Account Tbl

AccountNo	AccountBalance	type
		"S"
		"C"

Account Tbl

ATMFee	AccountNo

Checking Tbl

Interest	AccountNo

Savings Tbl

AccountNo	AccountBalance

Account Tbl

AccountNo	AccountBalance	ATMFee

Checking Tbl

AccountNo	AccountBalance	Interest

Savings Tbl

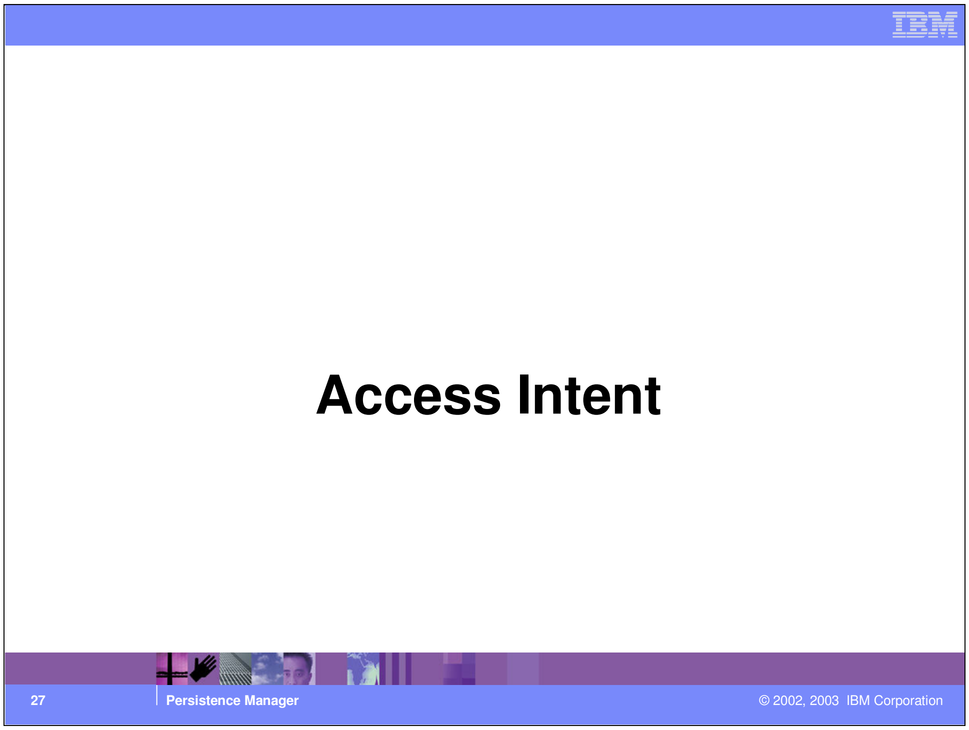
26 Persistence Manager © 2002, 2003 IBM Corporation

WebSphere supports inheritance through a variety of schema mapping options:

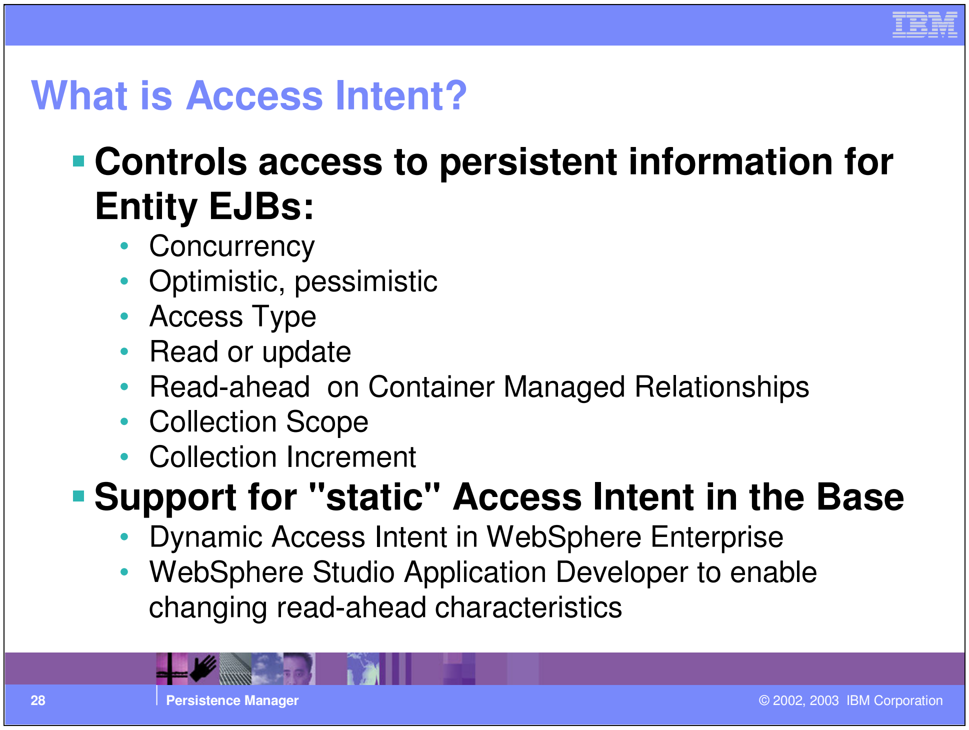
1) All classes (super- and sub-classes) will coexist in the same table. The table layout is a superset of all the attributes.

2) Root-leaf mapping (side tables): the Account table has the "common" attributes, plus a column that indicates the subclass. The "leaves" or side tables have the additional attributes. There is a side table for every subclass.

3)One table per class - no coexistence of different object types.



Access Intent



What is Access Intent?

- **Controls access to persistent information for Entity EJBs:**
  - Concurrency
  - Optimistic, pessimistic
  - Access Type
  - Read or update
  - Read-ahead on Container Managed Relationships
  - Collection Scope
  - Collection Increment
- **Support for "static" Access Intent in the Base**
  - Dynamic Access Intent in WebSphere Enterprise
  - WebSphere Studio Application Developer to enable changing read-ahead characteristics

28 | Persistence Manager © 2002, 2003 IBM Corporation

The Access Intent can be specified on EJB methods to control the way data access is going to occur.

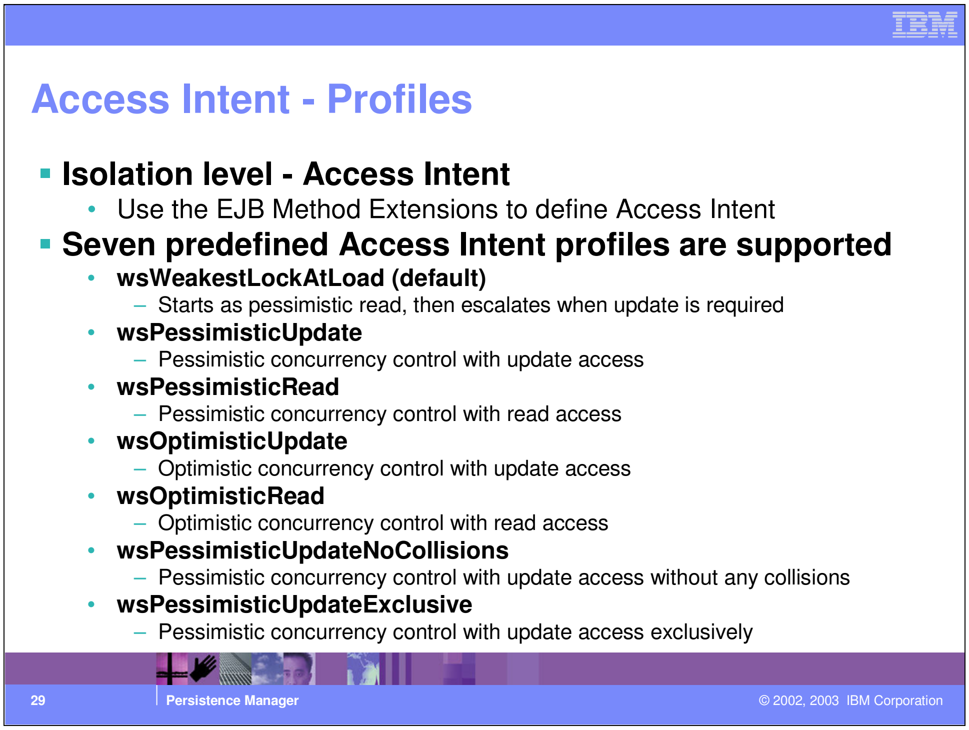
Generally speaking the access intent encapsulates a number of aspects of data access, including locking, access type, isolation level, and so on.

The Read ahead parameter controls how the collections should be navigated

Collection scope could be Transaction, Activity Session, or a fixed Timeout.

However- only the Enterprise Edition provides a way to fully customize the Access Intent

In the base, the WSAD will allow to modify the read-ahead setting - other than that, you will be able to select one of the seven predefined access intents.



Access Intent - Profiles

- **Isolation level - Access Intent**
  - Use the EJB Method Extensions to define Access Intent
- **Seven predefined Access Intent profiles are supported**
  - **wsWeakestLockAtLoad (default)**
    - Starts as pessimistic read, then escalates when update is required
  - **wsPessimisticUpdate**
    - Pessimistic concurrency control with update access
  - **wsPessimisticRead**
    - Pessimistic concurrency control with read access
  - **wsOptimisticUpdate**
    - Optimistic concurrency control with update access
  - **wsOptimisticRead**
    - Optimistic concurrency control with read access
  - **wsPessimisticUpdateNoCollisions**
    - Pessimistic concurrency control with update access without any collisions
  - **wsPessimisticUpdateExclusive**
    - Pessimistic concurrency control with update access exclusively

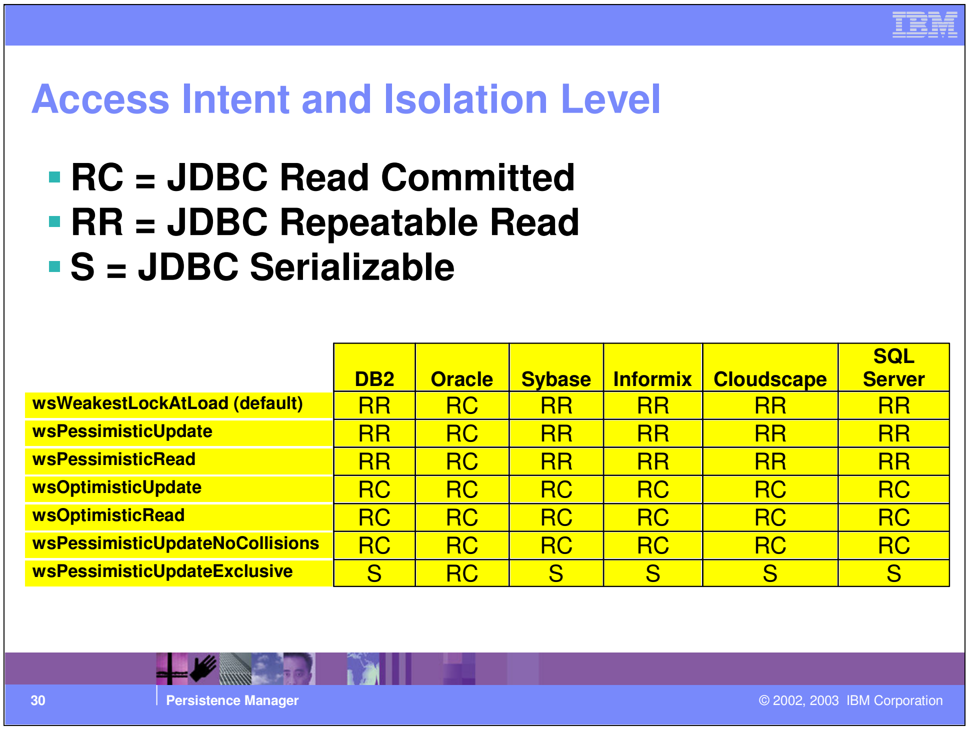
All optimistic access intents have the collection increment set to 25.

All access intents have the read-ahead parameter set to blank (no read-ahead)

All pessimistic access intents have collection increment set to 1, except for wsPessimisticUpdateNoCollisions where it is 25. In this case, though, you need to make sure that the WebSphere Persistence Manager is the only user of those database tables - or data loss and inconsistency may occur.

The wsWeakestLockAtLoad is pessimistic update - but it will initiate data access in a read mode, and will try to upgrade the lock level as needed.

In general Optimistic locking results into better concurrency - but the applications have to be engineered to catch optimistic exceptions and to avoid deadlocks.



Access Intent and Isolation Level

- RC = JDBC Read Committed
- RR = JDBC Repeatable Read
- S = JDBC Serializable

	DB2	Oracle	Sybase	Informix	Cloudscape	SQL Server
wsWeakestLockAtLoad (default)	RR	RC	RR	RR	RR	RR
wsPessimisticUpdate	RR	RC	RR	RR	RR	RR
wsPessimisticRead	RR	RC	RR	RR	RR	RR
wsOptimisticUpdate	RC	RC	RC	RC	RC	RC
wsOptimisticRead	RC	RC	RC	RC	RC	RC
wsPessimisticUpdateNoCollisions	RC	RC	RC	RC	RC	RC
wsPessimisticUpdateExclusive	S	RC	S	S	S	S

30 Persistence Manager

This chart shows the mapping between the various access intents and the corresponding isolation levels on the supported relational databases.

The access intents may result in one of three isolation levels:

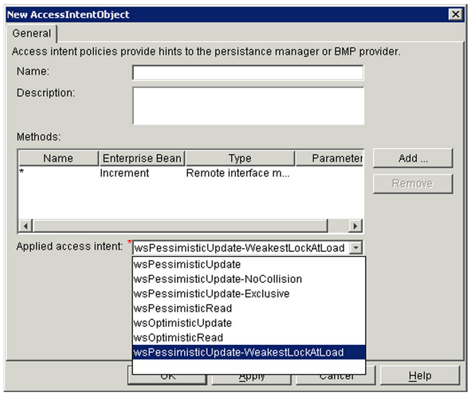
Read Committed - this level ensures that only committed data is read (no "dirty reads"). However, no locks on the read data are maintained for the duration of the transaction. This implies that another process may change and commit the data we are reading (non-repeatable read) or insert/delete a row that would match our selection criteria (phantom read).

Repeatable Read is like read committed, except that it ensure repeatable reads. In other words, another process would NOT be able to update a row that we obtained until our transaction is complete. Phantom reads are still possible though, as another process might insert a row that matches the selection criteria (if we repeat the query, we would get the newly inserted row)

Serializable ensures that repeatable read is enforced and disallows phantom reads too, by preventing other processes to insert/remove rows that would modify the result set until the serializable transaction has completed.

Setting Access Intent

- **Application Assembly Tool**
- **At the EJB module level, down to the individual EJB method**
- **WebSphere Studio App Developer allows modifying Read Ahead parameter**



New AccessIntentObject

General

Access intent policies provide hints to the persistance manager or BMP provider.

Name:

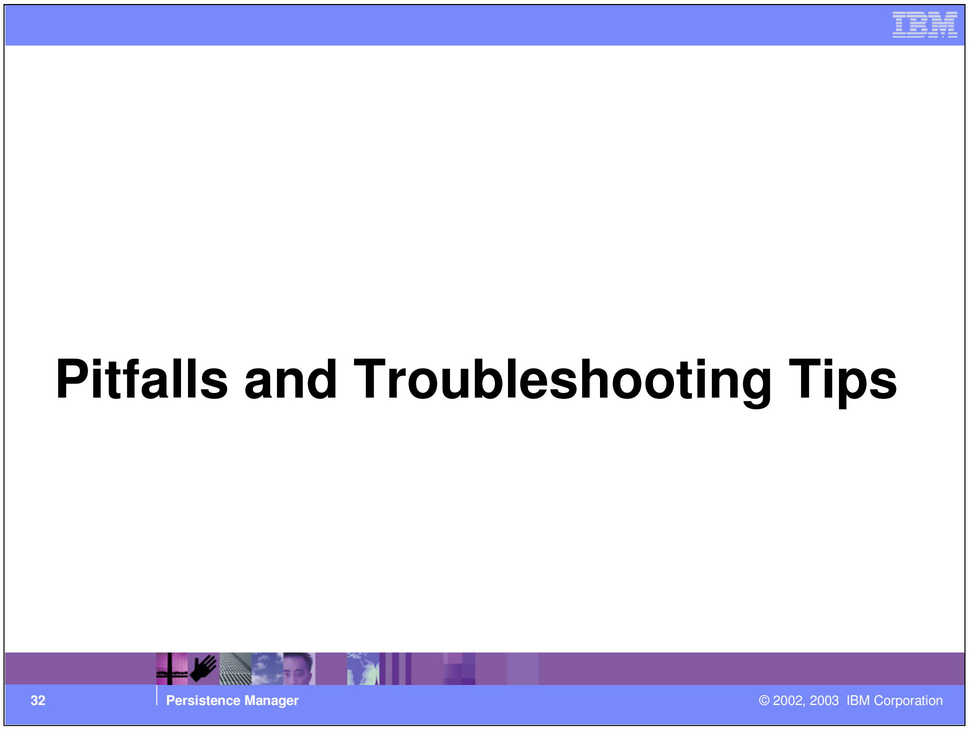
Description:

Methods:

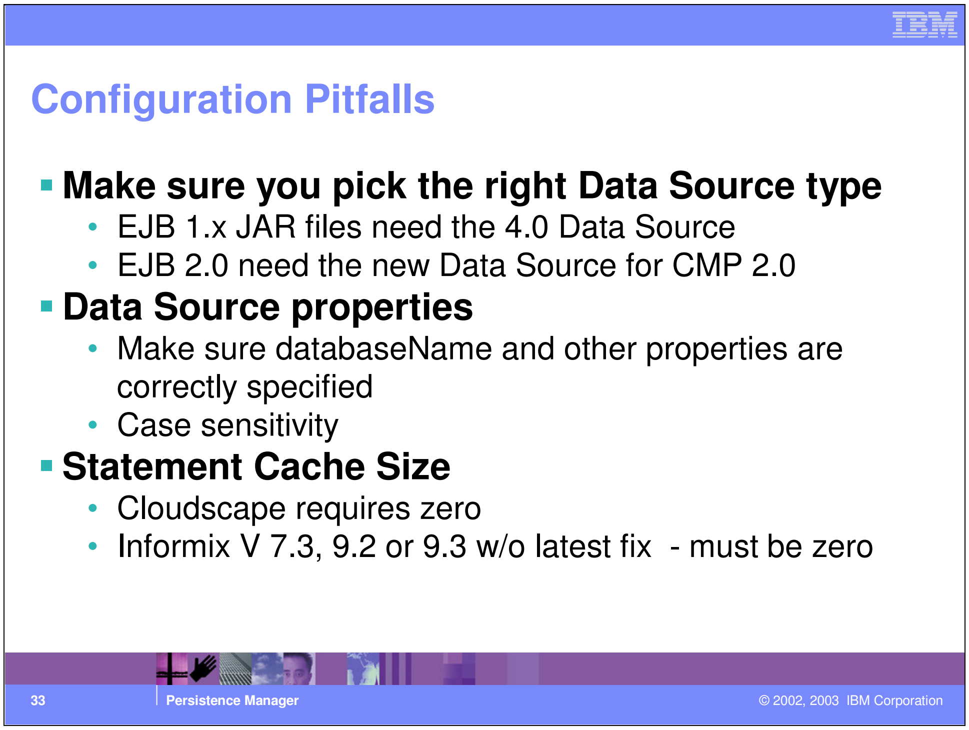
Name	Enterprise Bean	Type	Parameter
*	Increment	Remote interface m...	

Applied access intent: * wsPessimisticUpdate-WeakestLockAtLoad

This is how the AAT allows you to set the Access Intent. The granularity of Access Intent is down to the individual EJB method.



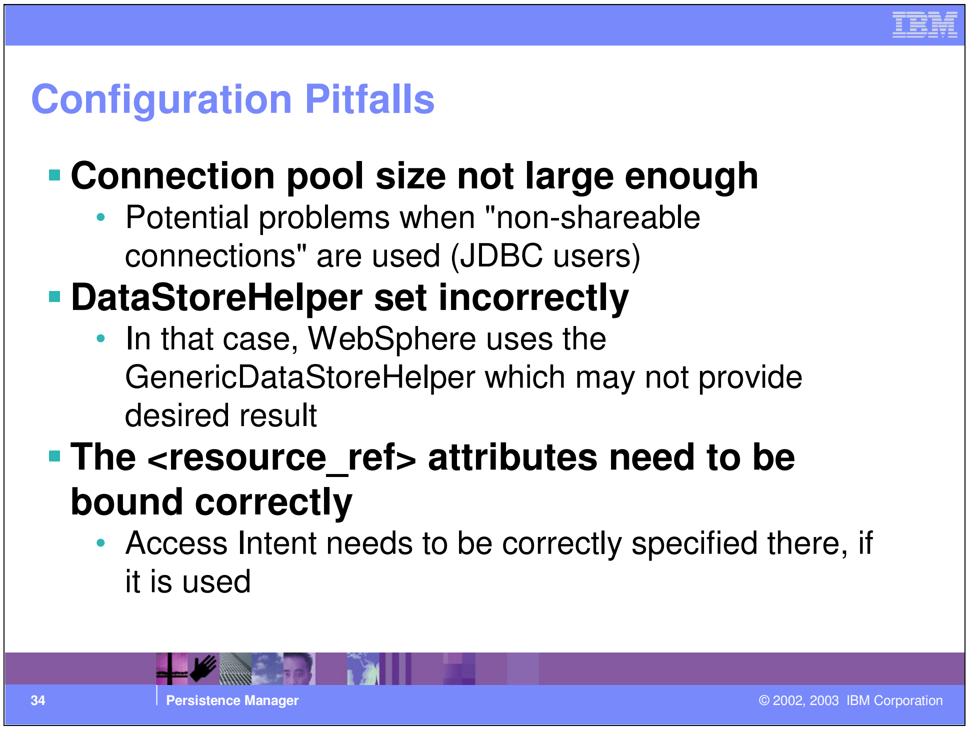
Pitfalls and Troubleshooting Tips



Configuration Pitfalls

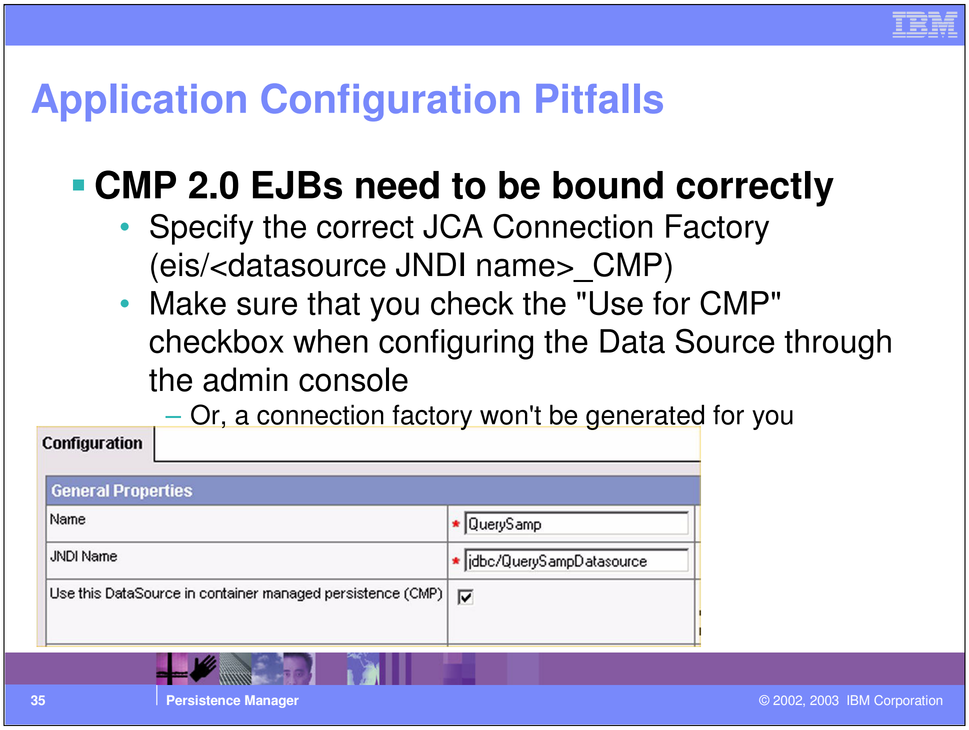
- **Make sure you pick the right Data Source type**
  - EJB 1.x JAR files need the 4.0 Data Source
  - EJB 2.0 need the new Data Source for CMP 2.0
- **Data Source properties**
  - Make sure databaseName and other properties are correctly specified
  - Case sensitivity
- **Statement Cache Size**
  - Cloudscape requires zero
  - Informix V 7.3, 9.2 or 9.3 w/o latest fix - must be zero

These are some common error situations. The admin console is structured in a way that it minimizes the chances you have to specify wrong configuration parameters for the datasources and the JDBC provider.



Configuration Pitfalls

- **Connection pool size not large enough**
  - Potential problems when "non-shareable connections" are used (JDBC users)
- **DataStoreHelper set incorrectly**
  - In that case, WebSphere uses the GenericDataStoreHelper which may not provide desired result
- **The <resource_ref> attributes need to be bound correctly**
  - Access Intent needs to be correctly specified there, if it is used



Application Configuration Pitfalls

- **CMP 2.0 EJBs need to be bound correctly**
  - Specify the correct JCA Connection Factory (eis/<datasource JNDI name>_CMP)
  - Make sure that you check the "Use for CMP" checkbox when configuring the Data Source through the admin console
    - Or, a connection factory won't be generated for you

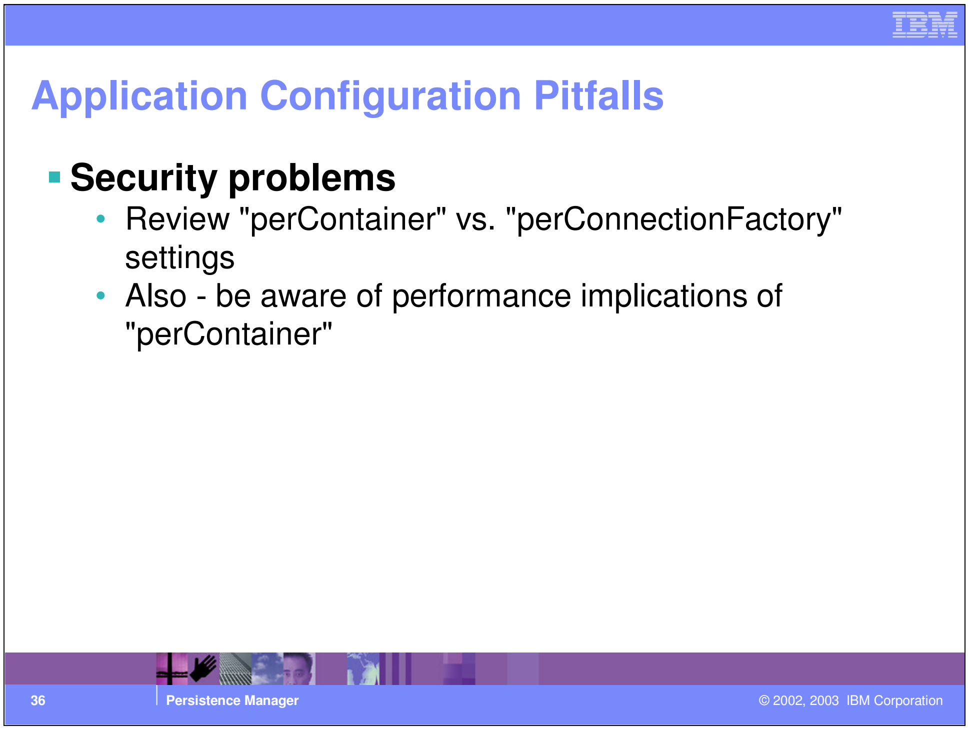
Configuration

General Properties

Name	* QuerySamp
JNDI Name	* jdbc/QuerySampDatasource
Use this DataSource in container managed persistence (CMP)	☑

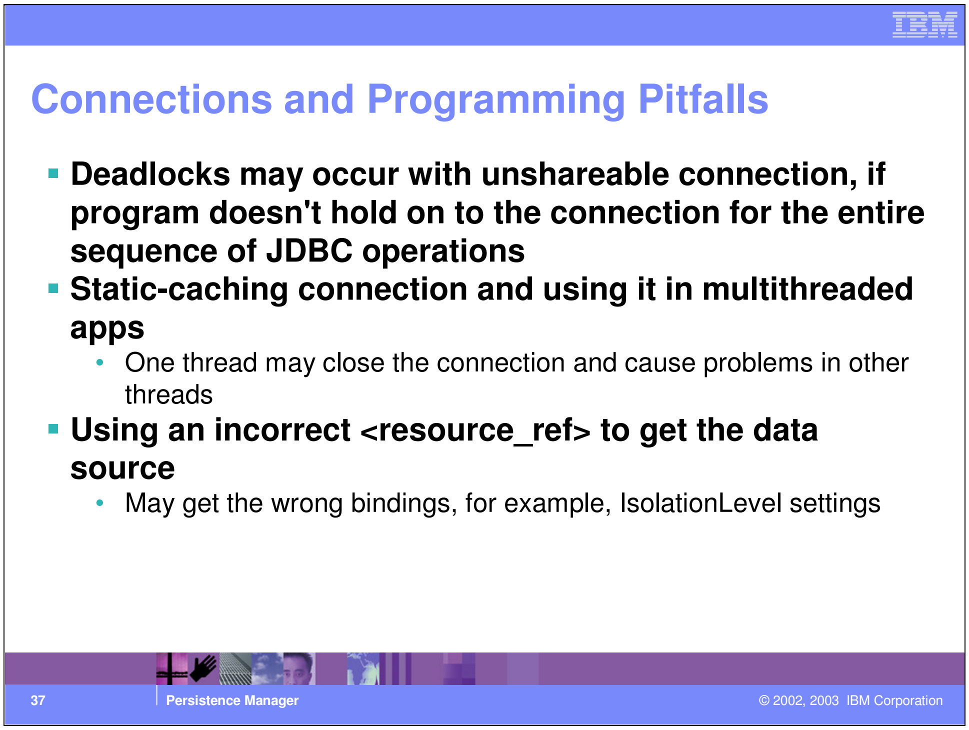
35 Persistence Manager

This is a crucial step when defining CMP 2.0 datasources. If you forget to check the "Use this DataSource in CMP" checkbox the connection factory will not be generated resulting in problems as the container starts the EJBs.



Application Configuration Pitfalls

- **Security problems**
  - Review "perContainer" vs. "perConnectionFactory" settings
  - Also - be aware of performance implications of "perContainer"

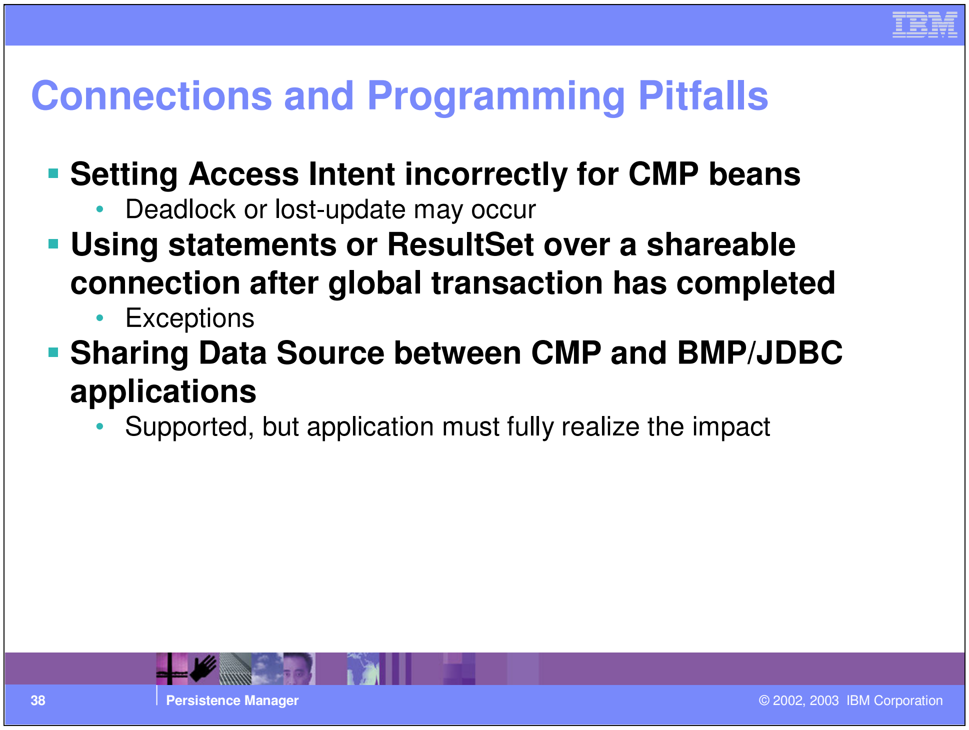


Connections and Programming Pitfalls

- **Deadlocks may occur with unshareable connection, if program doesn't hold on to the connection for the entire sequence of JDBC operations**
- **Static-caching connection and using it in multithreaded apps**
  - One thread may close the connection and cause problems in other threads
- **Using an incorrect <resource_ref> to get the data source**
  - May get the wrong bindings, for example, IsolationLevel settings

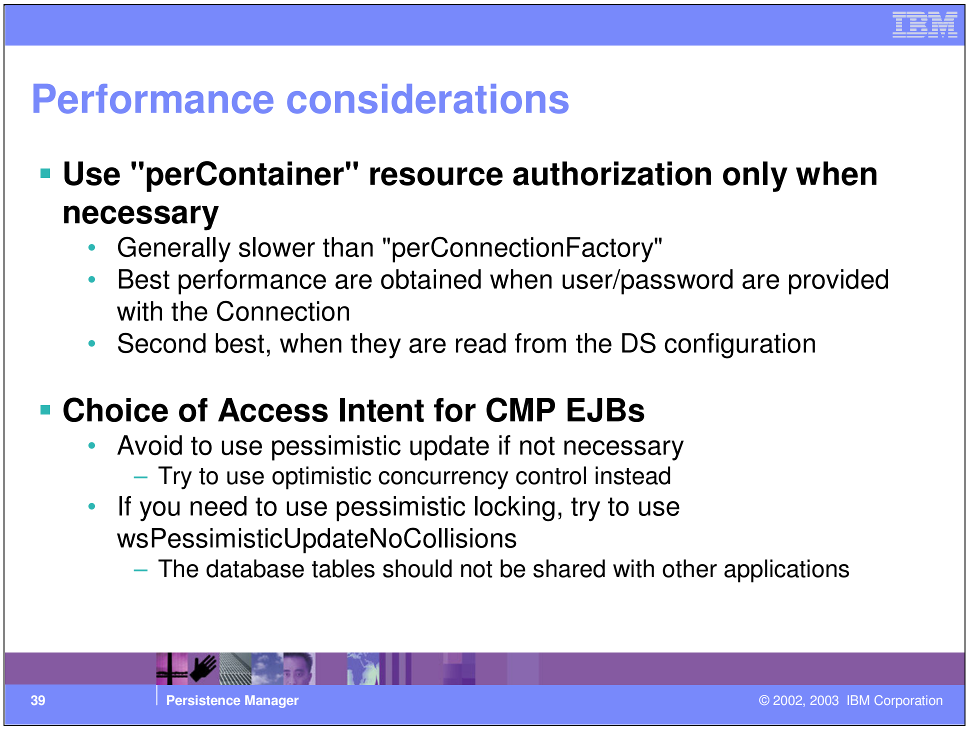
37 Persistence Manager

These are some more guidelines and recommendations related to the use of datasources and to the way data access needs to occur in WebSphere Application Server 5.0.



Connections and Programming Pitfalls

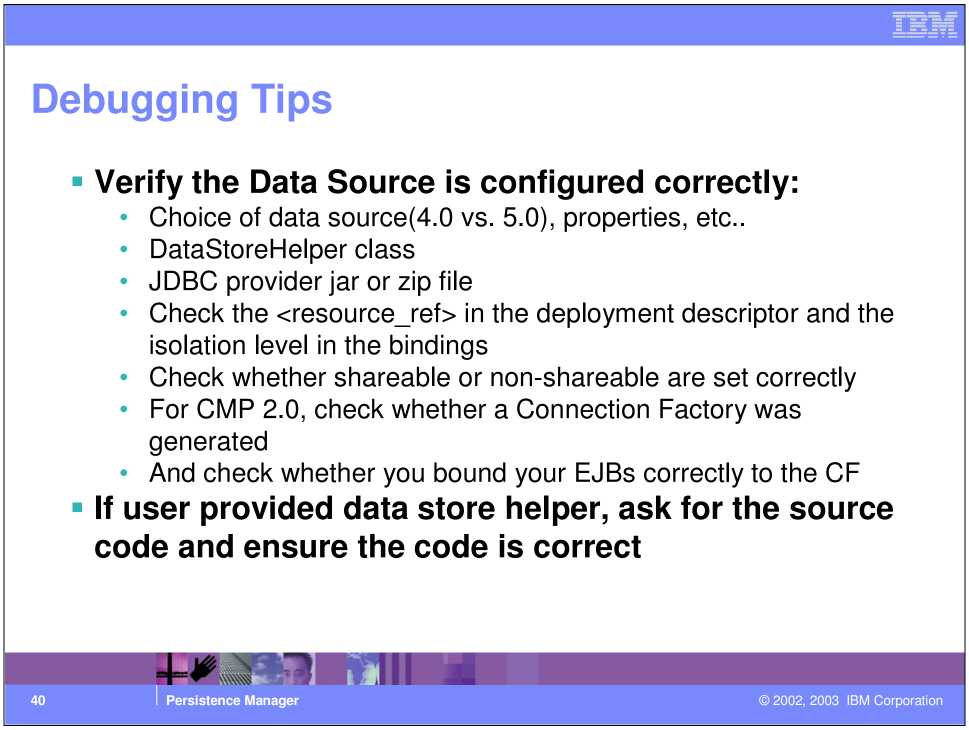
- **Setting Access Intent incorrectly for CMP beans**
  - Deadlock or lost-update may occur
- **Using statements or ResultSet over a shareable connection after global transaction has completed**
  - Exceptions
- **Sharing Data Source between CMP and BMP/JDBC applications**
  - Supported, but application must fully realize the impact



Performance considerations

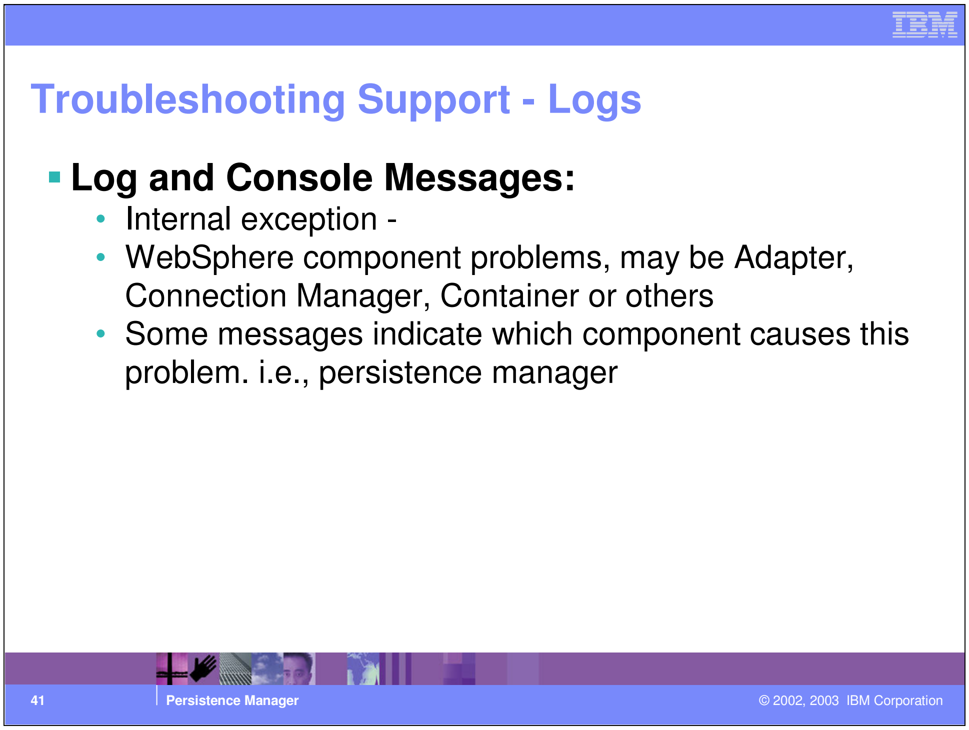
- **Use "perContainer" resource authorization only when necessary**
  - Generally slower than "perConnectionFactory"
  - Best performance are obtained when user/password are provided with the Connection
  - Second best, when they are read from the DS configuration

- **Choice of Access Intent for CMP EJBs**
  - Avoid to use pessimistic update if not necessary
    - Try to use optimistic concurrency control instead
  - If you need to use pessimistic locking, try to use wsPessimisticUpdateNoCollisions
    - The database tables should not be shared with other applications



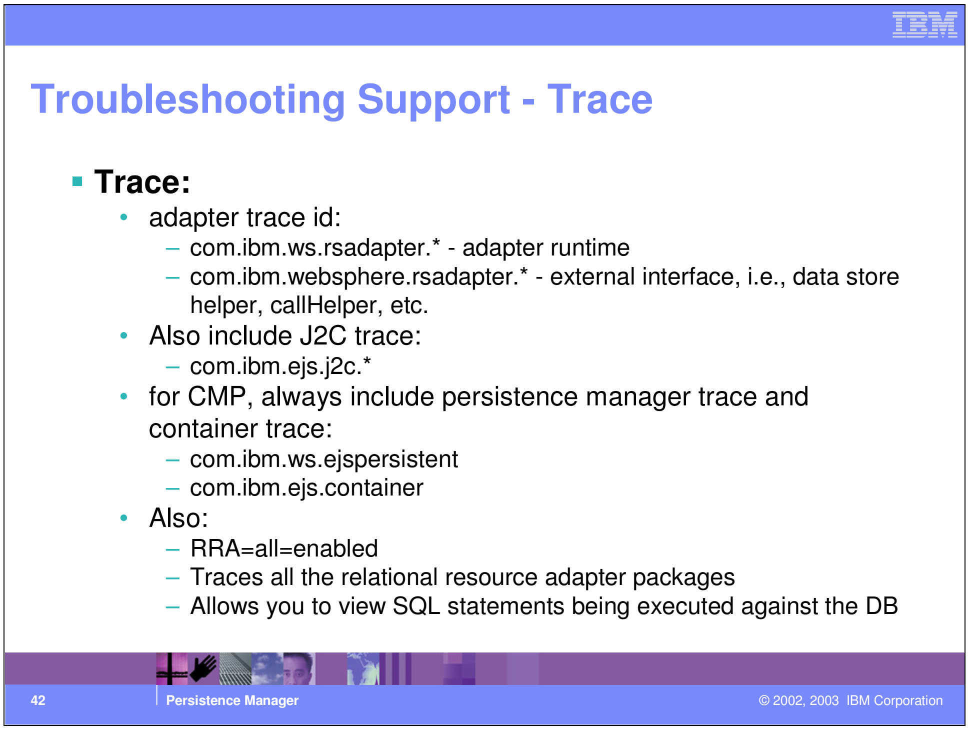
Debugging Tips

- **Verify the Data Source is configured correctly:**
  - Choice of data source(4.0 vs. 5.0), properties, etc..
  - DataStoreHelper class
  - JDBC provider jar or zip file
  - Check the <resource_ref> in the deployment descriptor and the isolation level in the bindings
  - Check whether shareable or non-shareable are set correctly
  - For CMP 2.0, check whether a Connection Factory was generated
  - And check whether you bound your EJBs correctly to the CF
- **If user provided data store helper, ask for the source code and ensure the code is correct**



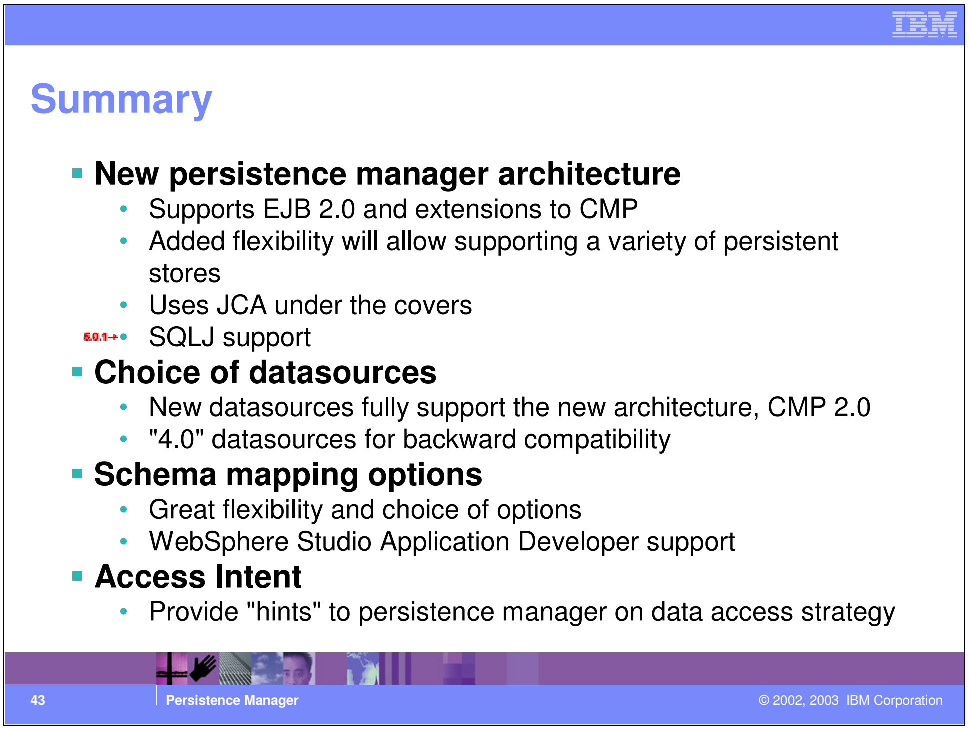
Troubleshooting Support - Logs

- **Log and Console Messages:**
  - Internal exception -
  - WebSphere component problems, may be Adapter, Connection Manager, Container or others
  - Some messages indicate which component causes this problem. i.e., persistence manager



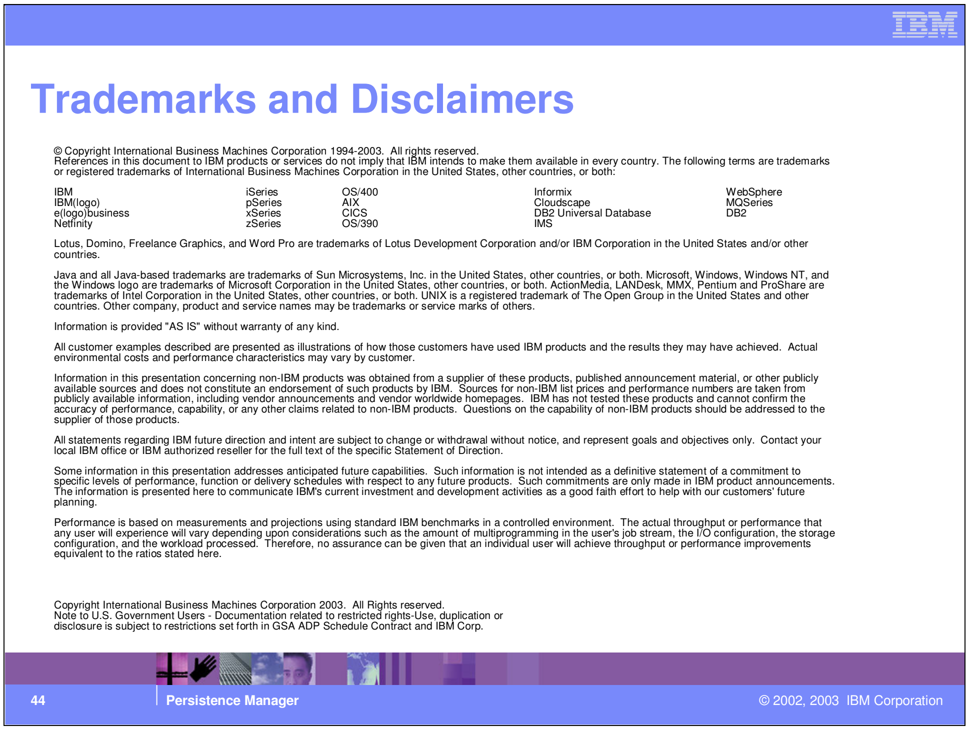
Troubleshooting Support - Trace

- **Trace:**
  - adapter trace id:
    - com.ibm.ws.rsadapter.* - adapter runtime
    - com.ibm.websphere.rsadapter.* - external interface, i.e., data store helper, callHelper, etc.
  - Also include J2C trace:
    - com.ibm.ejs.j2c.*
  - for CMP, always include persistence manager trace and container trace:
    - com.ibm.ws.ejspersistent
    - com.ibm.ejs.container
  - Also:
    - RRA=all=enabled
    - Traces all the relational resource adapter packages
    - Allows you to view SQL statements being executed against the DB



Summary

- **New persistence manager architecture**
  - Supports EJB 2.0 and extensions to CMP
  - Added flexibility will allow supporting a variety of persistent stores
  - Uses JCA under the covers
  - 6.0.1→ SQLJ support
- **Choice of datasources**
  - New datasources fully support the new architecture, CMP 2.0
  - "4.0" datasources for backward compatibility
- **Schema mapping options**
  - Great flexibility and choice of options
  - WebSphere Studio Application Developer support
- **Access Intent**
  - Provide "hints" to persistence manager on data access strategy



Trademarks and Disclaimers

References in this document to IBM products or services do not imply that IBM intends to make them available in every country. The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
IBM(logo)
e(logo)business
Netfinity

iSeries
pSeries
xSeries
zSeries

OS/400
AIX
CICS
OS/390

Informix
Cloudscape
DB2 Universal Database
IMS

WebSphere
MQSeries
DB2

Lotus, Domino, Freelance Graphics, and Word Pro are trademarks of Lotus Development Corporation and/or IBM Corporation in the United States and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries. Other company, product and service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.