WebSphere Adapter for JDBC Version 6.0.2.2 Fix Pack Guide

Contents

About this document

This document describes the features of IBM® WebSphere® Adapter for JDBC that are contained in fix pack 6.0.2.2. It contains new information and updates to topics in the WebSphere Adapter for JDBC User Guide. You can find the information center for WebSphere Adapter for JDBC at this link:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.wsadapters602.jca_jdbc.doc/doc/stjdb_welcome.html

WebSphere Adapter for JDBC, version 6.0.2 fix pack 2, includes the following functional enhancements:

- Connection properties in addition to user name and password

  Use the new DriverConnectionProperties property in the activation specification (for inbound operations) and managed connection factory (for outbound operations) to specify additional connection properties. See “Specifying additional JDBC driver connection properties for inbound and outbound connections” for more information about the property.

- Support for data sources that cache prepared statements

  Improve the performance of inbound and outbound operations using caching of prepared statements if a data source is used to connect to the database. For more information, see “Prepared statement caching using data sources.”

- Improved display of stored procedure names in the enterprise service discovery wizard

  Easily locate a stored procedure in the enterprise service discovery wizard because the stored procedure name is displayed in a more readable way. For more information, see “Improved display of stored procedure names.”

- Support for globalized data types in Oracle and MS SQL databases

  You no longer need to use the business object editor to manually add attributes for columns with globalized data types in Oracle and MS SQL. The data types NCHAR, NVARCHAR, NTEXT, TEXT, RAW, MONEY, and SMALLMONEY are now supported. See “Support for globalized data types in Oracle and MS SQL.”

- Improved support for business object hierarchies

  - Use the enterprise service discovery wizard instead of the business object editor to build a hierarchy of child business objects with the foreign keys set appropriately in parent and child business objects. See "Building hierarchical business objects."

  - Group unrelated business objects in a business object hierarchy by using the enterprise service discovery wizard to create wrapper business objects. See "Wrapper business objects."

- Support for NULL values in Update, Delete, Retrieve, and RetrieveAll operations

  The adapter can update, delete, or retrieve a record from a database table when the column value is NULL. To learn how the outbound operations were changed for this support, see "Updated documentation for outbound operations."

- Additional quick start tutorials

  Learn about the adapter using additional quick start tutorials. See "Additional quick start scenarios" for more information.

Specifying additional JDBC driver connection properties for inbound and outbound connections

In the Adapter for JDBC, version 6.0.2 or earlier, database connection properties were limited to the user name and password. Use the new DriverConnectionProperties property to optionally set additional driver-specific properties on the connection.

Set the DriverConnectionProperties property in the activation specification for inbound operations and in the managed connection factory for outbound operations. Table 1 describes the format of the new property.

In the enterprise service discovery wizard, type the additional connection properties in the new **JDBC Driver Class Properties** field in the Configure Settings for Discovery Agent window.

Table 1. New managed connection factory and activation specification property

Property name	**Type**	**Description**	**NLS characters allowed**
DriverConnectionProperties	String	Additional properties for connecting to the database using the JDBC driver. This information is used in addition to the Username and Password properties. Specify the value as one or more *Name*:*Value* pairs, separated by the semicolon character (;). For example, the following value of this property specifies a login timeout interval, makes a read-only database connection, and sets the security mechanism: `loginTimeout:20;readOnly:true; securityMechanism:USER_ONLY_SECURITY`	Yes, but bidirectional characters are not supported

Prepared statement caching using data sources

New support for prepared statement caching can improve the performance of inbound and outbound operations.

You can enable caching of prepared statements with the Adapter for JDBC if you use a data source to connect to the database. A prepared statement object in Java™ is a query containing an SQL statement that is compiled only once, but can be run multiple times. Caching can reduce the time required to perform an operation or a batch of operations. You can use a prepared statement for both inbound and outbound processing.

To gain this advantage, define a data source using the administrative console of the server and enable caching on the data source. Then use the DataSourceJNDIName property to name the data source you defined on the server. This property is described in Table 2.

For an example of how to enable prepared statement caching, refer to the quick start tutorial titled “Using a data source with a prepared statement cache.” For more information about the samples, see “Additional quick start tutorials.”

Table 2. DataSourceJNDIName property in activation specification and managed connection factory

Property	**Type**	**Description**	**Globalized**	**Default value**
DataSourceJNDIName	String	Name used by the adapter to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password in the data source when establishing the connection. If not, just the DataSourceJNDIName property is used. To improve the performance of inbound or outbound operations, specify the name of a data source that is enabled for prepared statement caching.	Yes	None

Wrapper business objects

Support has been added for wrapper business objects. A wrapper business object is a top-level business object that contains multiple table business objects, but does not correspond to any database table or view. Use a wrapper business object to create a container for unrelated business objects.

Overview of wrapper business objects

As a container for unrelated business objects, the wrapper business object can contain multiple (n) cardinality entities. A multiple (n) cardinality entity has at least one unique attribute marked as a primary key. When a wrapper business object is generated, its primary key attributes are the primary keys of the child business objects. The primary keys of the child business objects are marked as foreign keys and reference the primary key of the wrapper business object.

A wrapper business object is identified by the application-specific information property of Wrapper set to `true` in the top-level business object. It does not have TableName application-specific information. The attributes of a wrapper business object are all the key attributes of the child business objects. The attributes of the wrapper business object do not have any ColumnName application-specific information, because the wrapper does not map to objects in the target database. Figure 1 shows a sample wrapper business object that contains table business objects for the Customer and Order tables. The figure illustrates how the attributes and keys for a wrapper business object relate to the child business objects.

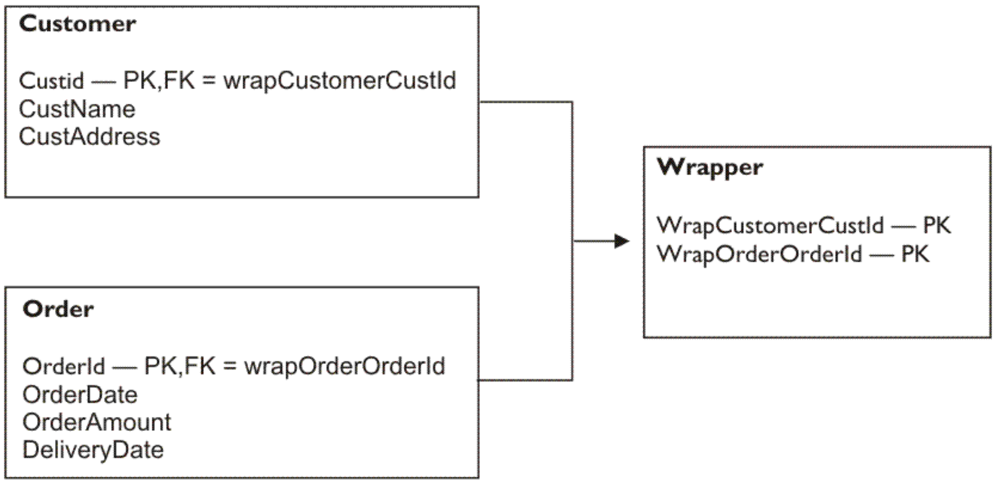

Figure 1. A wrapper business object that contains two table business objects

How a wrapper business object is processed

For information about how outbound operations are processed for wrapper business objects, see “Updated documentation for outbound operations.”

Creating a wrapper business object

Create a wrapper business object using the enterprise service discovery wizard. In the wizard, you create the top-level object and then select multiple table objects for it to contain. For detailed instructions for creating a wrapper business object and adding children to it, see the "Generating wrapper business objects" sample. For more information about the samples, see “Additional quick start tutorials.”

Business object-level application-specific information

Table 3 contains the application-specific information for the Wrapper business object.

Table 3. Business object-level application-specific information for wrapper business objects

Application-specific information	Type	Description	Bidirectional transformation supported
Wrapper	Boolean	Indicates whether the business objects is a wrapper business object When this property is `true`, no other business object level application-specific information is needed.	No

Improved display of stored procedure names

For stored procedures defined in PL/SQL packages, the application-specific information for stored procedures was generated correctly, but the names were hard to locate in the enterprise service discovery wizard. The names are now displayed in a more readable way.

In the Adapter for JDBC, version 6.0.2 before this fix pack, the enterprise service discovery wizard displayed the stored procedure names in the format *PackageName*.*SPName*. This made it hard to locate the stored procedure name in the list of stored procedures. The stored procedures are now displayed in the format *SPName*(*PackageName*).

For example, before this fix pack, if package A included a stored procedure named DSP, and package B included a stored procedure named CSP, the wizard displayed the stored procedures in the following order:
B.CSP
A.DSP

Now, these stored procedures are displayed in the following order:
CSP(B)
DSP(A)

Support for globalized data types in Oracle and MS SQL

Support for globalized data types in Oracle and MS SQL eliminates the need to use the business object editor to manually add attributes for columns with these types. The data types NCHAR, NVARCHAR, NTEXT, TEXT, RAW, MONEY, and SMALLMONEY are now supported.

The types returned by the JDBC metadata are mapped to the business object attribute types as listed in Table 4. The adapter supports only the JDBC types listed in the table. Any columns with types that are not listed are not added to the business object. An informational message is produced explaining the problem, for example, `The column named *xxxx* in the table named *yyyy* is not of a supported type and will not be added to the business object.`

Table 4. JDBC metadata column and business object attribute type

JDBC metadata column type	**Business object attribute type**
BIT	BOOLEAN
CHAR LONGVARCHAR VARCHAR	STRING
INTEGER NUMERIC SMALLINT TINYINT BIGINT	INT
TIME TIMESTAMP DATE	STRING
DECIMAL	STRING
DOUBLE FLOAT	DOUBLE
REAL	FLOAT
BLOB	HEXBINARY
CLOB	STRING
BINARY VARBINARY	HEXBINARY

LONGBINARY	
NCHAR NVARCHAR NTEXT	STRING
TEXT	STRING
RAW	STRING
MONEY SMALLMONEY	STRING

Building hierarchical business objects

You can now build a hierarchy of business objects with the enterprise service discovery tool, instead of using the business object editor.

Use the enterprise service discovery tool instead of the business object editor to build a hierarchy of child business objects that sets foreign keys appropriately in parent and child business objects. To review a sample describing how to do this, refer to the quick start tutorial titled "Using the enterprise service discovery wizard to define a business object hierarchy." For more information about the tutorial, see “Additional quick start tutorials.”

Additional quick start tutorials

New samples, or quick start tutorials, show how to use the Adapter for JDBC to accomplish common goals, such as creating a records in a table, retrieving records from the database, creating a hierarchy of business objects, using stored procedures, and so on. Each tutorial provides detailed instructions for accomplishing the goal of the tutorial. These samples are offered in addition to the quick start tutorial provided in the *Adapter for JDBC Version 6.0.2 User Guide*, which shows how to use the adapter to create a database record using parent and child business objects, and to associate a stored procedure with the child business object.

Download the new tutorials from the IBM technote "Quick start tutorials for WebSphere Adapter for JDBC" at following location:

http://www.ibm.com/support/docview.wss?rs=0&q1=JDBC+adapter+6.0.2+QSS&uid=swg27009124&loc=en_US&cs=utf-8&cc=us&lang=en

Tutorials are provided for the following goals:

- **Creating records in tables in a parent-child relationship using an Oracle database**

  This tutorial demonstrates how to create a record in tables in a parent-child relationship. The tutorial also demonstrates the use of a stored procedure attached to a business object. A stored procedure is associated with the child business object, using the application-specific information of the CreateSP operation. The adapter calls the stored procedure to create the record in the child table instead of generating the insert SQL statement.

- **Creating records in tables in a parent-child relationship using an MS SQL Server database**

  This tutorial demonstrates how to create a record in tables in a parent-child relationship. The tutorial also demonstrates the use of a stored procedure attached to a business object. A stored procedure is associated with the child business object, using the application-specific information of the CreateSP operation. The adapter calls the stored procedure to create the record in the child table instead of generating the insert SQL statement.

- **Creating a business object for a stored procedure and running the stored procedure using the Execute operation with an Oracle database**

This tutorial demonstrates how to create business objects for a stored procedure and then run the stored procedure using the Execute operation. The tutorial also demonstrates the support for Array and Struct data types.

- **Creating a business object for a stored procedure and running the stored procedure using the Execute operation with an MS SQL Server database**

  This tutorial demonstrates how to create business objects for a stored procedure and then run the stored procedure using the Execute operation. The tutorial also demonstrates the support for result sets returned by the stored procedure.

- **Creating a business object for a stored procedure and running the stored procedure using the Execute operation with a DB2® database**

  This tutorial demonstrates how to create business objects for a stored procedure and then run the stored procedure using the Execute operation. The tutorial also demonstrates the support for result sets returned by the stored procedure.

- **Creating a business object for a user-defined query and using the RetrieveAll operation to retrieve the records returned by the query with a DB2 database**

  This tutorial demonstrates how to create a business object for a user-defined query that does a join on two tables. It also describes how to use the RetrieveAll operation to retrieve the records returned by the user-defined query.

- **Creating a business object for a user-defined query and using the RetrieveAll operation to retrieve the records returned by the query with an Oracle database**

  This tutorial demonstrates how to create a business object for a user-defined query that does a join on two tables. It also describes how to use the RetrieveAll operation to retrieve the records returned by the user-defined query.

- **Creating a business object for a user-defined query and using the RetrieveAll operation to retrieve the records returned by the query with an MS SQL Server database**

  This tutorial demonstrates how to create a business object for a user-defined query that does a join on two tables. It also describes how to use the RetrieveAll operation to retrieve the records returned by the user-defined query.

- **Using an XA data source to perform XA transactions using the Adapter for JDBC with a DB2 database**

This tutorial demonstrates how to use an XA data source to perform XA transactions using the Adapter for JDBC with a DB2 database.

- **Using an XA data source to perform XA transactions using the Adapter for JDBC with an Oracle database**

  This tutorial demonstrates how to use an XA data source to perform XA transactions using the Adapter for JDBC with an Oracle database.

- **Using a data source with a prepared statement cache**

  This tutorial demonstrates how to use a data source with a prepared statement cache using the Adapter for JDBC with an Oracle database.

- **Define a business object hierarchy using the enterprise service discovery wizard**

  This tutorial demonstrates how to build a hierarchy of multiple cardinality business objects using the enterprise service discovery wizard instead of the business object designer. It also describes how to set the Ownership, Keep Relationship, and Required application-specific information for the child business object. The tutorial uses the Adapter for JDBC with an Oracle database.

- **Generating wrapper business objects**

  This tutorial demonstrates how to create a wrapper business object that contains child business objects for two unrelated database tables. The tutorial uses the Adapter for JDBC with an Oracle database.

Updated documentation for outbound operations

Outbound operations Create, Delete, Retrieve, RetrieveAll, and Update now support wrapper objects and NULL values. The following sections contain complete reference information about these operations:

- Create operation
- Delete operation
- Retrieve operation
- RetrieveAll operation
- Update operation

Create operation

When given a hierarchical business object, the Create operation recursively traverses the business object, creating rows corresponding to each table.

To process the Create operation, the adapter performs the following actions:

1. If a top-level business object is a wrapper object, it is ignored. No processing is done for wrapper objects.

2. The Create operation recursively inserts each single-cardinality child business object contained with ownership into the database. In other words, the adapter creates the child and all child business objects that the child and its children contain.

   If the business object definition specifies that an attribute represents a child business object with single cardinality and that attribute is empty, the adapter ignores the attribute. However, if the business object definition requires that the attribute represent a child, and it does not, the adapter returns an error and stops processing.

3. The Create operation retrieves and checks for the existence of each single-cardinality child business object contained without ownership. If the retrieval is unsuccessful, indicating that the child does not exist in the database, the adapter returns an error and stops processing. If the Retrieve operation is successful, the adapter recursively updates the child business object.

Note: For this approach to work correctly when the child business object exists in the application database, primary key attributes in child business objects must be cross-referenced correctly on Create operations. If the child business object does not exist in the application database, the primary key attributes must not be set.

4. The adapter inserts the top-level business object in the database by performing the following actions:
   a. Sets each of its foreign key values to the primary key values of the corresponding child business object represented with single cardinality. Because values in child business objects can be set by database sequences or counters or by the database itself during the creation of the child, this step ensures that the foreign key values in the parent are correct before the adapter inserts the parent in the database.
   b. Generates a new, unique ID value for each attribute that is set automatically by the database. The name of the database sequence or counter is stored in the attribute's application-specific information. If an attribute has an associated database sequence or counter, the value generated by the adapter overwrites any value passed in by the application server.
   c. Inserts the top-level business object into the database.

5. The adapter processes each of the multiple-cardinality child business objects as follows:
   a. Sets the foreign key values in each child to reference the value in the corresponding primary key attributes in the parent. Because the parent's primary key values might have been generated during the creation of the parent, this ensures that the foreign-key values in each child are correct before the adapter inserts the child into the database.
   b. Inserts each of the multiple-cardinality child business objects into the database.

Delete operation

The Delete operation is performed by pruning the incoming business object and then retrieving the complete business object from the database. The Delete operation is then applied recursively on each business object in the hierarchy.

The Delete operation supports physical and logical deletes, depending on the StatusColumnName value in the application-specific information of the business object. If the StatusColumnName value is defined, the adapter performs a logical delete operation. If the StatusColumnName value is not defined, the adapter performs a physical delete operation.

Physical deletes

For physical deletes the adapter takes the following actions:

- It recursively deletes all multiple-cardinality child business objects.
- It deletes the top-level business object.

  If the top-level business object is a wrapper object, it is ignored. No delete is done for wrapper business objects.
- It recursively deletes all single-cardinality child business objects contained with ownership.

Logical deletes

For logical deletes the adapter takes the following actions:

- It issues an update that sets the status attribute of the business object to the value specified by the business object-level application-specific information. The adapter ensures that only one database row is updated as a result, and it returns an error if this is not the case.
- It recursively logically deletes all single-cardinality children contained with ownership and all multiple-cardinality children. The adapter does not delete single-cardinality children contained without ownership.

NULL data and the Delete operation

The adapter can delete a record from a database table when the column value is null. For example, a Customer business object might have these columns: custid, ccode, fname, and Iname, where custid and ccode form composite keys. Composite keys are primary keys that refer to more than one attribute and are used to define the uniqueness of the business object. You can delete a Customer record for which ccode is null. The adapter generates a delete query for the Delete operation as:

```
delete from customer where custid=? and ccode is null
```

Retrieve operation

To process the Retrieve operation, the adapter performs the following actions:

1. Removes all child business objects from the top-level business object received. In other words, it makes a copy of the top-level business object without any children.

2. Retrieves the top-level business object from the database.
    - o If the retrieval returns one row, the adapter continues processing.
    - o If the retrieval returns no rows, indicating that the top-level business object does not exist in the database, the adapter returns the error `RecordNotFoundException`.
    - o If the retrieval returns more than one row, the adapter returns an error.

3. Recursively retrieves all multiple-cardinality child business objects.

   Note: The adapter does not enforce uniqueness when populating an array of business objects. It is the database's responsibility to ensure uniqueness. If the database returns duplicate child business objects, the adapter returns duplicate children.

4. Recursively retrieves each of the single-cardinality children, regardless whether the child business object is contained with or without ownership.

   Note: All single-cardinality child business objects are processed based on their occurrence in the business object and before the parent business object is processed. Child object ownership and non-ownership do not determine the processing sequence, but they do determine the type of processing.

NULL data and the Retrieve operation

The adapter can retrieve a record from a database table when the column value is null. For example, a Customer business object might have these columns: custid, ccode, fname, and Iname, where custid and ccode form composite keys. Composite keys are primary keys that refer to more than one attribute and are used to define the uniqueness of the business object. You can retrieve a Customer record for which ccode is null. The adapter generates a select query for the Retrieve operation as:

```
select custid, ccode, fname, Iname from customer where
custid=? and ccode is null
```

RetrieveAll operation

The adapter uses the RetrieveAll operation to retrieve an array of business objects from the database. The way the adapter retrieves an array differs depending on whether the RetrieveAll operation is for database table business objects or for user-specified SQL business objects.

For database table business objects

All of the key and non-key attributes populated in the incoming business object determine the selection criteria for the retrieval. The adapter may retrieve multiple rows for the top-level business object from the database, depending on the attributes selected. If no attributes are populated in the incoming business object, all the rows are retrieved from the respective table in the database.

The name of a generated business object matches the name of the table in the database. For example, the Customer table in the database will be represented as a business object named "Customer".

To retrieve an array of business objects, the adapter performs the following steps:

1. For each of the rows retrieved from the database, it constructs a top-level business graph and creates a container of business graphs using all of the retrieved rows. The name of the container business object is the business object name with the string "Container" appended to it.
2. Retrieves each of the business graphs in the container using the Retrieve operation, to create the hierarchy shown in Figure 2.

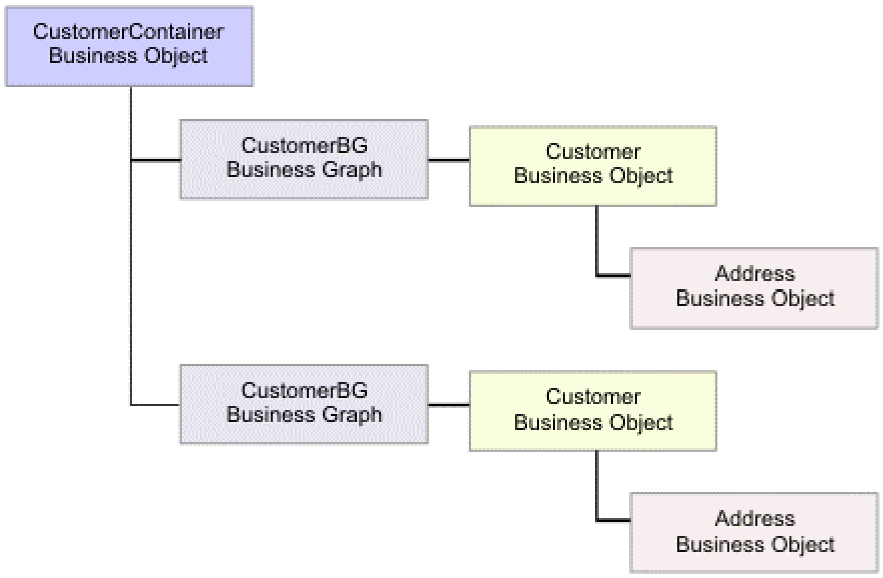

Figure 2. Structure of the business object returned in a RetrieveAll operation

The following errors can result from a RetrieveAll operation:

- RecordNotFoundException—One or more of the populated business objects in the input object does not exist in the enterprise information system.

- MatchesExceededLimitException—The number of hits in the EIS exceeds the value of the ResultSetLimit property that is defined in the interaction specification. The MatchCount property contains the actual number of matches that the adapter found in the EIS, so that you can either increase the limit, or refine the search appropriately.

  Note: If the ResultSetLimit property is set to a very large number, problems may occur related to a lack of sufficient memory, depending upon the size and number of business objects returned.

- EISSystemException—One or more unrecoverable errors are reported by the EIS.

For user-specified SQL business objects

Business objects that are created for user-specified SQL statements also support the RetrieveAll operation. When the enterprise service discovery wizard generates the query business object for the user-specified SQL statement, the adapter runs the SELECT SQL statement and returns data to the database in the structure shown in Figure 3.

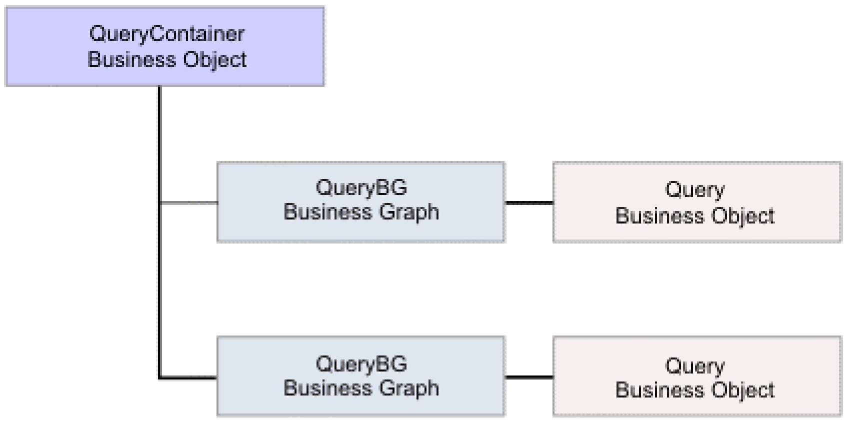

Figure 3. User-specified SQL business object

To process the query business object generated by the enterprise service discovery wizard for the user-specified SELECT statement, the adapter performs the following actions:

1. Obtains the SELECT SQL statement from the query business object.
2. Checks whether a dynamic where clause is specified in the query business object.
   - If there is a dynamic where clause, the adapter replaces the default where clause in the SELECT statement with the dynamic one.
   - If there is no dynamic where clause, the adapter replaces parameters in the SELECT statement with the corresponding values specified in the query business object.
3. Runs the SELECT statement.

4. Obtains the result set that is returned and populates the query business object values with the data returned from the database, creating a container business object with the structure shown in Figure 3.
5. Retrieves the entire hierarchy (a *deep retrieve*) of each top-level query business object in the container, if any child business objects are defined for the query business objects.

   Note: Query business objects can be only top-level business objects. A query business object cannot have child query business objects.

NULL data and the RetrieveAll operation

The adapter can retrieve records from a database table when the column value is null. For example, a Customer business object might have these columns: custid, ccode, fname, and Iname, where ccode need not be a primary key. You can query all Customer records for which ccode is null. The adapter generates a select query for the RetrieveAll operation as:

```
select custid, ccode, fname, Iname from customer where
custid=? and ccode is null
```

Update operation

The Update operation is done by comparing the incoming business object with a business object that is retrieved from the database using the primary keys specified in the top-level, incoming business object.

When updating a hierarchical business object, the adapter performs the following steps:

1. It uses the primary key values of the source business object to retrieve the corresponding entity from the database. The retrieved business object is an accurate representation of the current state of the data in the database.

   If the retrieval fails, indicating that the top-level business object does not exist in the database, the adapter returns a `RecordNotFoundException` error, and the update fails.

   If the retrieval succeeds, the adapter compares the retrieved business object to the source business object to determine which child business objects require changes in the database. The adapter does not, however, compare values in the source business object's simple attributes to those in the retrieved business object. The adapter updates the values of all non-key simple attributes.

If all of the simple attributes in the top-level business object represent keys, the adapter cannot generate an update query for the top-level business object. In this case, the adapter logs a warning and continues.

2. It recursively updates all single-cardinality children of the top-level business object.

   If the business object definition requires that an attribute represent a child business object, the child must exist in both the source business object and the retrieved business object. If it does not, the update operation fails, and the adapter returns an error.

   The adapter handles single-cardinality children contained with ownership in one of the following ways:

   - If the child is present in both the source and the retrieved business objects, instead of updating the existing child in the database, the adapter deletes the existing child and creates the new child.
   - If the child is present in the source business object but not in the retrieved business object, the adapter recursively creates the child in the database.
   - If the child is present in the retrieved business object but not in the source business object, the adapter recursively deletes the child from the database.

   For single-cardinality children contained without ownership, the adapter attempts to retrieve every child from the database that is present in the source business object. If it successfully retrieves the child, the adapter populates the child business object but does not update it, because the adapter never modifies single-cardinality children contained without ownership.

3. It updates all simple attributes of the retrieved business object, except those whose corresponding attribute in the source business object is not specified.

   Because the business object being updated must be unique, the adapter verifies that only one row is processed as a result. It returns an error if more than one row is returned.

4. It processes each multiple-cardinality child of the retrieved business object in one of the following ways:
   - If the child exists in both the source and the retrieved business objects' arrays, the adapter recursively updates it in the database.
   - If the child exists in the source array but not in the array of the retrieved business object, the adapter recursively creates it in the database.
   - If the child exists in the array of the retrieved business object but not in the source array, the adapter recursively deletes it from the database unless the application-specific information for the attribute that represents the child in

the parent has the KeepRelationship property set to `true`. In this case, the adapter does not delete the child from the database.

NULL data and the Update operation

The adapter can update a record from a database table when the column value is null. For example, a Customer business object might have these columns: custid, ccode, fname, and Iname, where custid and ccode form composite keys. Composite keys are primary keys that refer to more than one attribute and are used to define the uniqueness of the business object. You can update a Customer record for which ccode is null. The adapter would generate an update query for the Update operation as:

```
update customer set fname=?, Iname=? where custid=? and ccode
is null
```