17

Using Oracle Object References

This chapter describes the standard Java Database Connectivity (JDBC) that let you access and manipulate object references.

This section discusses the following topics:

17.1 Oracle Extensions for Object References

Oracle supports the use of references to database objects. Oracle JDBC provides support for object references as:

- Columns in a `SELECT` clause
- IN or OUT bind variables
- Attributes in an Oracle object
- Elements in a collection type object

In SQL, an object reference (`REF`) is strongly typed. For example, a reference to an `EMPLOYEE` object would be defined as an `EMPLOYEE REF`, not just a `REF`.

When you select an object reference, be aware that you are retrieving only a pointer to an object, not the object itself. You have the choice of materializing the reference as a `java.sql.Ref` instance for portability, or materializing it as an instance of a custom Java class that you have created in advance, which is strongly typed. Custom Java classes used for object references are referred to as **custom reference classes** and must implement the `oracle.jdbc.OracleData` interface.

You can retrieve a `REF` instance through a result set or callable statement object, and pass an updated `REF` instance back to the database through a prepared statement or callable statement object. The `REF` class includes functionality to get and set underlying object attribute values, and get the SQL base type name of the underlying object.

Custom reference classes include this same functionality, as well as having the advantage of being strongly typed. This can help you find coding errors during compilation that might not otherwise be discovered until run time.

> **Note:**
>
> - If you are using the `oracle.jdbc.OracleData` interface for custom object classes, then you will presumably use `OracleData` for corresponding custom reference classes as well. However, if you are using the standard `java.sql.SQLData` interface for custom object classes, then you can only use weak Java types for references. The `SQLData` interface is for mapping SQL object types only.
> - You can create and retrieve `REF` objects in your JDBC application only by running SQL statements. There is no JDBC-specific functionality for creating and retrieving `REF` objects.
> - You cannot have a reference to an array, even though arrays, like objects, are structured types.

17.2 Retrieving and Passing an Object Reference

This section discusses JDBC functionality for retrieving and passing object references. It covers the following topics:

- Retrieving an Object Reference from a Result Set
- Retrieving an Object Reference from a Callable Statement
- Passing an Object Reference to a Prepared Statement

17.2.1 Retrieving an Object Reference from a Result Set

To demonstrate how to retrieve object references, the following example first defines an Oracle object type `ADDRESS`, which is then referenced in the `PEOPLE` table:

```
create type ADDRESS as object
   (street_name     VARCHAR2(30),
    house_no        NUMBER);

create table PEOPLE
    (col1 VARCHAR2(30),
     col2 NUMBER,
     col3 REF ADDRESS);
```

The `ADDRESS` object type has two attributes: a street name and a house number. The `PEOPLE` table has three columns: a column for character data, a column for numeric data, and a column containing a reference to an `ADDRESS` object.

To retrieve an object reference, follow these general steps:

1. Use a standard SQL `SELECT` statement to retrieve the reference from a database table `REF` column.
2. Use `getRef` to get the address reference from the result set into an `OracleRef` instance.
3. Let `Address` be the Java custom class corresponding to the SQL object type `ADDRESS`.
4. Add the correspondence between the Java class `Address` and the SQL type `ADDRESS` to your type map.

5. Use the `getObject` method to retrieve the contents of the `Address` reference. Cast the output to `Address`.

The `PEOPLE` database table is defined earlier in this section. The code for the preceding steps, except the step of adding `Address` to the type map, is as follows:

```
ResultSet rs = stmt.executeQuery("SELECT col3 FROM PEOPLE");
while (rs.next())
{
   OracleRef ref = rs.getRef(1);
   Address a = (Address)ref.getObject();
}
```

> **Note:**
>
> In the preceding code, `stmt` is a previously defined statement object.

17.2.2 Retrieving an Object Reference from a Callable Statement

To retrieve an object reference as an `OUT` parameter in PL/SQL blocks, you must register the bind type for your `OUT` parameter.

1. Cast your callable statement to `OracleCallableStatement`, as follows:

   ```
   OracleCallableStatement ocs =
      (OracleCallableStatement)conn.prepareCall("{? = call func()}");
   ```

2. Register the `OUT` parameter with the following form of the `registerOutParameter` method:

   ```
   ocs.registerOutParameter (int param_index, int sql_type, String sql_type_name);
   ```

   param_index is the parameter index and *sql_type* is the SQL type code. The *sql_type_name* is the name of the structured object type that this reference is used for. For example, if the `OUT` parameter is a reference to an `ADDRESS` object, then `ADDRESS` is the *sql_type_name* that should be passed in.

3. Run the call, as follows:

   ```
   ocs.execute();
   ```

17.2.3 Passing an Object Reference to a Prepared Statement

Pass an object reference to a prepared statement in the same way as you would pass any other SQL type. Use either the `setObject` method or the `setREF` method of a prepared statement object.

Use a prepared statement to update an address reference based on `ROWID`, as follows:

```
PreparedStatement pstmt =
   conn.prepareStatement ("update PEOPLE set ADDR_REF = ? where ROWID = ?");
pstmt.setRef (1, addr_ref);
    pstmt.setRowId (2, rowid);
```

17.3 Accessing and Updating Object Values Through an Object Reference

You can use the `Ref` object `setObject` method to update the value of an object in the database through an object reference. To do this, you must first retrieve the reference to the database object and create a Java object that corresponds to the database object.

For example, you can use the code in the "Retrieving and Passing an Object Reference" section to retrieve the reference to a database `ADDRESS` object, as shown in the following code snippet:

```
ResultSet rs = stmt.executeQuery("SELECT col3 FROM PEOPLE");
if (rs.next())
{
   Ref ref = rs.getRef(1);
   Address a = (Address)ref.getObject();
}
```

Then, you can create a Java `Address` object that corresponds to the database `ADDRESS` object. Use the `setObject` method of the `Ref` interface to set the value of the database object, as follows:

```
Address addr = new Address(...);
ref.setObject(addr);
```

Here, the `setValue` method updates the database `ADDRESS` object immediately.

Related Topics

- Retrieving and Passing an Object Reference