

Bridging The Object-Relational Gulf

Howard Ong

Principal Consultant

Aurora Consulting Pty Ltd

Abstract

With the ever-increasing popularity of J2EE applications comes the need to bridge the gulf between the new object-oriented world of Java and the old relational world of databases. Of the many Object-Relational Mapping Tools available today, Oracle Toplink offers superb functionality and versatility. This paper introduces the concept of an Object-Relational Mapping tool and provides an overview of Oracle Toplink functionalities. For a higher level of realism, the paper uses a demo application to explain some Toplink tips and traps.

About The Author

A frequent presenter at local and regional conferences, Howard has been a proficient user of database technology since 1991. In particular, Howard possesses in-depth experience in the planning and development of Business Intelligence and E-Business Applications. Harboring a keen interest in the application of advanced database in these areas, Howard and his team have helped many organisations deploy BI and E-Business solutions using Oracle, Microsoft and IBM technology. Since 1997, Aurora Consulting has been delivering Data Warehouses and E-Business solutions to a wide variety of organisations including large government departments and ASX-listed companies.

For more information on Howard or Aurora Consulting, visit <http://www.aurora-consult.com.au> or email info@aurora-consult.com.au.

Introduction

While object-oriented technology becomes increasingly popular as the solution of choice for building enterprise applications, the relational database remains firmly entrenched as the preferred data storage solution. Hence, a gulf has developed – between object-oriented world of the tools and the relational technology world of the database.

In a traditional Java application, the developer would build the necessary logic to transform relational data into objects and vice-versa. This leads to applications that are complex to build and sometimes slow to run. Moreover, such custom-built object-relational transformation is also less likely to be reusable.



The solution to this is to have a third-party tool takes over the Object-Relational Mapping process – hence the need for Object-Relational Mapping Tools.



Object-Relational Mapping

The purpose of an Object-Relational Mapping Tool is to provide an abstraction layer between the database rows and the Java objects. In this manner, the developer only has to deal with a persistent layer of objects, independent of the underlying relational structure. Hidden from the developer, the Object-Relational Mapping Tool handles all the relational transactions, queries, locking, caching in the background. Thus, the use of an Object-Relational Mapping Tool has 2 advantages :

- Developer productivity gain
- Improved application performance

Toplink Basics

Oracle Toplink consists of 2 components :

- Toplink Mapping Workbench, a GUI for specifying Object-Relational Mapping rules.
- Toplink Foundation Library, essentially a collection for JAR files for inclusion in the application.

Mapping Workbench

For this demonstration, we use the following relational tables that most readers should be familiar :

```
SQL> desc emp
Name          Null?         Type
-----
EMPNO         NOT NULL     NUMBER(4)
ENAME         VCHAR(10)
DEPTNO        NUMBER(2)
```

```
SQL> desc dept
Name          Null?         Type
-----
DEPTNO        NOT NULL     NUMBER(2)
DNAME         VCHAR(14)
```

The main steps involved in creating and mapping Java classes to these tables are :

- Create and compile the descriptor objects : Employee and Department.
- Create a new project in Mapping Workbench.
- Import table definitions Emp and Dept. Create a reference (FK) in Emp to Dept.
- Import the classes Employee and Department.
- Map each of the classes to their respective tables.
- Map each attribute and specify the getter and setter methods. Most attributes are mapped directly to table columns.
- Specify Employee.dept as a one-to-one mapping with Indirection. Specify the Indirection getter and setter methods. Specify the table reference (FK).
- Generate the deployment XML.

An extract of a deployment XML generated by Toplink is displayed below :

```
<descriptor>
  <java-class>au.wa.aurorademo.toplinkdemo.Employee</java-class>
  <tables>
    <table>EMP</table>
  </tables>
  <primary-key-fields>
    <field>EMP.EMPNO</field>
  </primary-key-fields>
  ...
  <mappings>
    <database-mapping>
      <attribute-name>empNo</attribute-name>
      <read-only>false</read-only>
      <get-method-name>getEmpNo</get-method-name>
      <set-method-name>setEmpNo</set-method-name>
      <field-name>EMP.EMPNO</field-name>
      <type>oracle.toplink.mappings.DirectToFieldMapping</type>
    </database-mapping>
    ...
  </mappings>
  <type>oracle.toplink.publicinterface.Descriptor</type>
</descriptor>
```

A Special Note On Indirection :

Foreign key relationships between tables are usually translated to an object being associated with another. In this example, a Department object is encapsulated within the Employee object to indicate which department the employee belongs to. Indirection allows the associated Department object to be created only when required, thereby promoting efficiency.

To support Indirection, instead of defining a native Department object within Employee, a proxy object implementing the ValueHolderInterface interface is defined. The getter and setter methods are also modified to respectively return and accept the ValueHolderInterface. These are used by Toplink to set and retrieve the Department object. Finally, to present a simple and consistent API to the application developer, the author recommends the creation of another pair of getter and setter methods that respectively returns and accepts the Department object. Below is the code example :

```
private ValueHolderInterface dept;

public ValueHolderInterface
getDeptIndirect() {
    return dept;
}

public void
setDeptIndirect(ValueHolderInterface dept)
{
    this.dept = dept;
}

public Department getDept() {
    return (Department) dept.getValue();
}

public void setDept(Department dept) {
    this.dept.setValue(dept);
}
```

Getting It To Work

Below are the main steps for a typical Toplink query operation on the Employee :

- Create a Toplink Project object from the deployment XML file.
- Create a Toplink DatabaseSession object from the Project object. Get the DatabaseSession object to login.
- Build a query Expression.
- Execute the query and retrieve an empMaster object.

A typical Toplink insert or update operation on the Employee consists of the following main steps :

- Initiate a new transaction for the DatabaseSession.
- Create an UnitOfWork object from the DatabaseSession.
- Register empMaster with the UnitOfWork. In the case of an insert, empMaster is a newly created object. In the case of an update, empMaster is an existing object previously queried. This registration returns empUow, a clone of empMaster.
- Update empUow, the clone object.
- Commit the UnitOfWork.
- Refresh the empMaster with a fresh copy from the database.

To update an associated object, eg changing the Employee's Department, follow these steps :

- Query the new department using the **same** DatabaseSession. This is the deptMaster object.
- Register deptMaster with the **same** UnitOfWork. This returns the deptUow, a clone of deptMaster.
- Assign deptUow to empUow.

When performing an update, do take note of the following :

- Toplink requires that the same DatabaseSession used for query be used for the update. Otherwise, if a new DatabaseSession is used, Toplink will generate a Primary Key violation error.
- Note that all updates are to be performed on empUow, the clone object. The empMaster must **not** be changed as Toplink uses it as a base reference for implementing optimistic locking.
- Upon a successful update, get Toplink to refresh empMaster from the database. Otherwise, Toplink will keep using its cache value for that object. An alternative to refreshing empMaster is to turn off caching altogether. This, however, has performance implications.
- For a web application, the author recommends that Optimistic Locking be enabled. With Optimistic Locking enabled, be sure to check for OptimisticLockException after every update operation.

The steps required to delete the Employee object is very similar to update :

- Initiate a new transaction for the DatabaseSession.
- Create a UnitOfWork object from the DatabaseSession.
- Register a previously queried empMaster object with the UnitOfWork. This returns empUow, a clone of empMaster.

- Call the method UnitOfWork.deleteObject() on empUow, the clone object.
- Commit the UnitOfWork.

Other Object-Relational Mapping Tools

Toplink is just one of the many Object-Relational Mapping tools available in the market today. Notably, there are a number of high-quality tools in the open-source arena. Object/Relational Bridge and Hibernate are 2 prominent open-source tools that spring to mind. Appendix A is a list of other Object-Relational Mapping tools in the market. **The list is far from exhaustive.**

Conclusion

As evident from the preceding example, even though Toplink has some powerful features, the developer has to follow a number of rules when interacting with it.

It may also be evident to some readers that there is some functionality overlap between Toplink and BC4J. In future releases of these products, Oracle will be streamlining their respective functionality, with BC4J tending towards a front-end code-generator; and Toplink performing all backend interaction with the database.

Appendix A : Other Object-Relational Tools

(Please note that following list is not exhaustive.)

- Abra (<http://abra.sourceforge.net>)
- Castor (<http://castor.exolab.org>)
- Cayenne Framework (<http://objectstyle.org/cayenne>)
- CocoBase (<http://thoughtinc.com>)
- Databind (<http://databind.sourceforge.net>)
- DBGen (<http://dbgen.sourceforge.net>)
- EdgeXtend (<http://www.persistence.com/products/edgextend>)
- Hibernate (<http://hibernate.sourceforge.net>)
- Hywy (<http://www.hywy.com>)
- Java Skyline (<http://www.javaskyline.com/database.html#ormap>)
- Jaxor Framework (<http://jaxor.sourceforge.net>)
- JCorporate Toolkit (<http://www.jcorporate.com>)
- JDO Genie (<http://www.hemtech.co.za/jdo/index.html>)
- Kodo JDO (http://www.solarmetric.com/Software/Kodo_JDO)
- Libelis (<http://www.libelis.com>)
- Object Frontier (<http://www.objectfrontier.com>)
- Object/Relational Bridge (<http://jakarta.apache.org/ojb>)
- Simple ORM (<http://www.simpleorm.org>)
- Software Tree (<http://www.softwaretree.com>)
- SQL2Java (<http://sql2java.sourceforge.net/>, <http://www.bitmechanic.com/projects/s2j>)
- Torque (<http://jakarta.apache.org/turbine/torque>)