

Build Superior Java Applications with Oracle TopLink

*An Oracle White Paper
September 2005*

Build Superior Java Applications with Oracle TopLink

Executive Overview	3
The Challenge	3
The Solution	4
Why not just use JDBC?	4
Isn't Base EJB Container Managed Persistence (CMP) "Good enough"?	5
Build or Buy?	5
What a Persistence Architecture Must Provide	5
Caching	5
Querying	5
Locking	6
Deferred Reading	6
Sequencing	6
Transaction Support and Integration	6
The Importance of Flexibility	6
How does TopLink achieve such a high degree of flexibility?	7
Information Technology Challenges Organizations face Today	7
Oracle Application Server 10g and its Benefits	8
TopLink and Grid Computing	9
Conclusion	9

Build Superior Java Applications with Oracle TopLink

EXECUTIVE OVERVIEW

Building Java applications that use relational databases is perhaps the single most underestimated challenge in enterprise development today. More projects are delayed, under featured and difficult to maintain because of this underestimation. The problem lies with the use of fundamentally different technologies. Java and relational databases typically have different skill sets, different staff and ownership and different tools, modeling and design principles. Oracle TopLink provides a proven, powerful “out-of-the-box” solution to address this diversity between Java and relational databases. TopLink supports grid computing by enabling flexible application development and providing crucial persistence functionality.

THE CHALLENGE

Object and component technology have become the solution of choice for building enterprise applications. However, most organizations have a great deal invested in relational databases and have vital corporate data stored there. Relational databases are mature and their capacity and performance are predictable and reliable.

The object world and the relational world do not match up. One world consists of tables, rows, columns, and foreign keys; the other world contains object references, business rules, complex relationships, and inheritance. This is often referred to as the object/relational “impedance mismatch”.

JDBC provides a standard for connecting Java applications and relational databases but this is merely an interface for executing SQL and receiving database rows as results. Developers need a layer of infrastructure to manage translating the database rows to and from Java objects and components. In addition to this, developers need to be able to integrate transactions, queries, caching, locking, sequencing and other key database concepts into their applications.

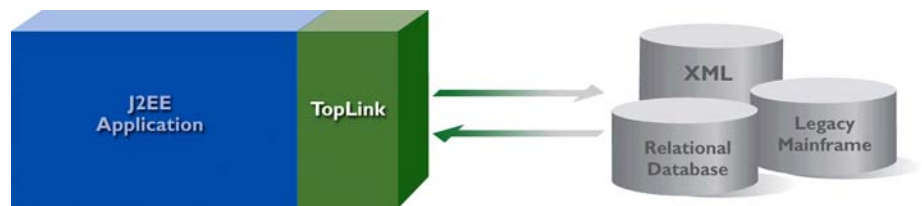
A persistence architecture for building Java and databases needs to not only boost developer productivity, but also application performance. Productivity can be aided with tools that help configure and manage how data is mapped and by providing infrastructure to address all the issues developers will encounter. Performance can be addressed with features that minimize database hits and

network traffic while always leveraging optimizations provided by JDBC and the databases.

Performance and productivity must not come at the cost of flexibility. Java developers need to be able to come up with their ideal architectures and a data architect must have the flexibility to design the relational schema efficiently as possible. Therefore, the persistence architecture needs to be highly flexible and unobtrusive.

THE SOLUTION

TopLink answers the challenge by linking the object and relational data worlds, allowing applications to transparently store and retrieve Java objects using a relational database or other storage systems.



TopLink allows Java applications to access data stored in relational databases as objects. Application developers are able to work with relational databases much as they would any Java application regardless of persistence.

With TopLink, developers focus on the application and object model rather than the infrastructure of the database. TopLink can map Java objects to an existing (legacy) database or can suggest a new database schema from an object model (or vice versa). It provides a rich set of features to read, write, delete, and manage objects efficiently.

The mature, robust persistence framework provided by TopLink significantly reduces the risk of using a relational database in a Java application.

Why not just use JDBC?

JDBC defines a low-level API for accessing databases. It does not work at the level of Java objects. Developers using JDBC must write methods that contain SQL statements and convert between database rows and business objects. TopLink does not replace JDBC, but makes efficient use of it and isolates developers from its details. Application developers work with objects using TopLink, rather than rows and SQL using JDBC calls.

Although TopLink isolates developers from JDBC, it provides the features and flexibility required to leverage and manage any special performance enhancements provided by your JDBC vendor.

Isn't Base EJB Container Managed Persistence (CMP) "Good enough"?

Relying exclusively on CMP does not allow you to build persistence solutions for commonly used Java architectures such as regular Java objects in Servlets and JSPs and Session Beans accessing regular Java business objects. Most application servers force you to make *every* persistent component an Entity Bean whereas TopLink allows using either CMP Entity Beans, Java objects, BMP Entity Beans or a mixture. Architects and developers often find the mapping and querying support with CMP to be too restrictive, and the memory and performance overhead too high. By allowing a choice in architectures, and by providing industry leading CMP support, TopLink allows maximum flexibility in application design and performance.

For more information on this topic, please visit:

http://otn.oracle.com/products/ias/toplink/addvalueover_ejb2.html

Build or Buy?

Building a custom persistence framework can easily consume 30-40% of a project's resources. This problem is much more challenging than it first appears and often requires the efforts of the most experienced members of a project team. The resulting homegrown framework requires support and maintenance and may not be re-usable on other projects. Persistence is often on the critical path of a project and a mature, commercial product such as TopLink provides substantial benefits. With TopLink, a project's resources can focus on building the application, not on infrastructure.

WHAT A PERSISTENCE ARCHITECTURE MUST PROVIDE

When considering solutions to help building Java applications with relational databases, it is easy to focus on "object-relational mapping" and overlook the other infrastructure that is required. "Impedance mismatch" is more than dealing with the composition and decomposition of objects and relational data. Among countless others, the following features are also a key to a good persistence architecture:

Caching

There must be flexible caching options so that frequently used data can be shared and reused efficiently. This leads to improved application performance and memory management. A good caching solution can be tailored to the application's needs but allowing control over the volume and durability of cached objects on a class-by-class basis.

Querying

Where appropriate, a developer should be free to build queries using SQL. If there are legacy queries or security concerns, then a persistence architecture should allow stored procedures to be used. In a lot of cases, it would be preferred to allow

queries to be written in Java, within the context of the business model. This allows Java developers who may not be SQL experts to leverage their skills and improves maintainability by further isolating database details from the application and vice versa.

Locking

There must be solid support for locking whenever multiple applications or threads share access to data. It is essential that any Java application manage locking in such a way that it complies with any legacy applications that may be sharing the same data. Developers should have control over the degree of flexibility including optimistic, pessimistic and “last one wins” strategies.

Deferred Reading

Relational databases routinely store terabytes of data whereas most Java applications are designed to run in just a few hundred megabytes of memory. It is therefore important that the developer can easily manage the object graph being used by the application to minimize reading unnecessary data. Managing this should be transparent to the developer yet highly configurable where necessary.

Sequencing

Relational databases require data to have a unique field to represent each row. Objects do not have this constraint. A persistence architecture should support various strategies for managing sequencing including native (database) managed sequencing, application domain assigned sequencing or perhaps the persistence architecture should automatically manage sequencing on its own.

Transaction Support and Integration

Application servers manage Java based transactions when using entity beans for persistence. When using Java objects, a Java based transaction framework is needed. Regardless of if entity beans or Java objects are being used, a transaction framework should minimize database interactions and order SQL to respect database constraints. Java based transactions should be easily integrated with external transaction managers such as XA and JTS.

THE IMPORTANCE OF FLEXIBILITY

Java architects want their applications to be portable across J2EE Servers and relational databases and want their developers to be able to use solutions within any development process or tool set. Data managers typically have to design their models around the enterprise, not any particular application or technology. It is therefore essential that a persistence architecture not only facilitates portability, but also be usable within any development process or tool set and allow for a high degree of flexibility when mapping the Java and relational models.

TopLink's design goal is to be highly flexible with the Java model and relational schema. Application developers work at the level of Java. When objects are read in, not only are the instance variables filled in with data but also references to other objects are automatically maintained. The referenced objects are traversed by navigating the object model using normal Java methods, rather than making additional explicit database calls or managing foreign keys.

TopLink supports arbitrarily complex models and automatically maintains references between objects. Changing the database schema does not normally require changes to the business objects (and vice versa), only to the TopLink mappings – which are easily managed with the TopLink Mapping Workbench. Java business classes can be re-used with a completely different database, or database schema.

How does TopLink achieve such a high degree of flexibility?

TopLink creates a set of meta-data “descriptors”, or mappings, that defines how objects are to be stored in a particular database schema. TopLink uses these mappings at run-time to dynamically generate the required SQL statements. The descriptors can be changed without having to re-compile the classes they represent.

No SQL programming is required. The meta-data descriptors (mappings) are independent of both language and database.

The SQL generated can be completely controlled and managed by the developer or database manager.

Information Technology Challenges Organizations face Today

The primary challenge facing organizations today is the very high cost of their information technology infrastructure. This very high cost arises from three related causes:

- *Excess Computing Capacity* that is poorly utilized due to the need to build capacity for peaks, and the inability to use the spare capacity efficiently
- *Expensive Capacity Growth* due to the inability to add capacity quickly, when needed, and in low cost, modular units to avoid further compounding the problem of excess capacity
- *High Cost of Management* due to the complexity of systems; the specialized tools, procedures, and skills required; and the large amounts of human intervention needed to manage systems.

Grid Computing & Oracle's Grid Computing Offering

Grid computing is a new software architecture designed to effectively pool together large amounts of low cost modular storage and servers to create a virtual computing resource across which work can be transparently distributed to use capacity very efficiently, at low cost, and with very high availability. The *resources* in a grid can include storage, servers, databases, application servers, and applications. By pooling resources together, Grid Computing can offer dependable, consistent,

pervasive, and inexpensive access to these resources regardless of their location and when needed, thereby fulfilling the need for computing capacity on-demand.

While Grid Computing has primarily been used by the scientific community to solve very specialized problems, the rapid evolution of cost-effective networked storage; high speed, high density blade servers; high speed network Interconnects; and low cost operating systems together with the evolving capabilities of systems software (databases and Application Servers) to exploit these advances have now made it possible for Enterprises to leverage the benefits of Grid Computing.

Oracle offers a comprehensive solution to manage information and run Enterprise Applications on Grids using Oracle Database 10g and Oracle Application Server 10g. Both Oracle Database 10g and Oracle Application Server 10g can be managed in a Grid Computing environment using Oracle Grid Control. Together these products address the challenges faced by I/T organizations today:

- *Radically Reduce or Eliminate Excess Computing Capacity* by automatically load balancing workloads to use spare capacity efficiently eliminating “islands of computation”
- *Modular, Inexpensive Capacity Growth* by adding capacity on-demand in low cost modular units
- *Radically Lower Cost of Management* by centralizing administration of the resources in a Grid and automating provisioning and administration tasks across these resources

Oracle Application Server 10g and its Benefits

Oracle Application Server 10g, the next generation of Oracle’s Integrated Software Infrastructure for Enterprise Applications has been designed to enable Grid Computing. It has been designed to effectively pool together large numbers of low cost servers to create a virtual computing resource across which Enterprise Applications can be transparently distributed to use capacity very efficiently, at low cost, and with very high availability. Any existing application that runs on Oracle Application Server can transparently take advantage of Grid Computing without any changes. Service-Oriented Applications will find additional benefits when deployed in a Grid. Oracle Application Server 10g provides a number of Grid Computing features, most importantly:

- *Radically Reduce or Eliminate Excess Computing Capacity* through Policy-Based Resource Management; Metrics-based Workload Management; and a variety of advanced back up, disaster recovery, and clustered fail-over solutions to provide maximum availability in a Grid.
- *Modular, Inexpensive Capacity Growth* through Automated Installation, Configuration, and Software Provisioning (including both software cloning and patch management) across hundreds of nodes in a Grid.
- *Radically Lower Cost of Management* and eliminating human errors in management through Centralized Systems Monitoring, Unified Application Server Cluster Management (including Cluster Monitoring,

Cluster Optimization, and Cluster-wide Application Deployment), and centralized Identity Management across a Grid.

TopLink and Grid Computing

TopLink supports the grid infrastructure by enabling the flexibility required to develop service-oriented architectures, and by providing the persistence architecture required by applications.

CONCLUSION

Building Java applications that use relational data is a significant challenge for organizations. TopLink assists architects and developers by delivering a proven persistence architecture at the onset of the development process.

TopLink significantly reduces the time-to-market and development effort required for building enterprise Java applications.

