Whitepaper

TOPLink for Java, Version 2.5

The Challenge:

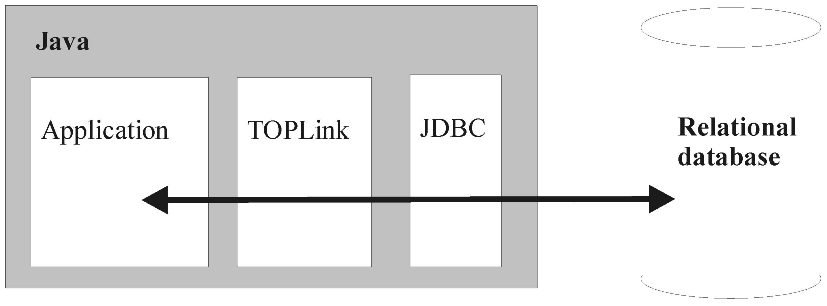
Object technology has become the solution of choice for building enterprise applications. However, many organizations have a great deal invested in relational databases and have much of their vital corporate data stored there. Relational databases are mature and their capacity and performance are predictable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

Objects need to access data stored in relational databases. However, the object world and the relational world do not completely 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” and is a major challenge for organizations adopting object technology.

The Solution:

TOPLink answers the challenge by providing a bridge between the two worlds, allowing applications to transparently store and retrieve Java objects using a relational database.



The TOPLink framework: Java applications, TOPLink, JDBC, and the relational database.

Features

TOPLink's feature set has evolved from feedback received by thousands of developers in more than thirty different countries. The TOPLink client base reflects a wide cross-section of industries and diverse requirements. As a result, TOPLink has a solid design that provides a great deal of flexibility and performance.

TOPLink's features include:

- ✓ minimal changes to the domain model
  - NO methods to explicitly save and restore
  - NO SQL in domain classes
  - no need to sub-class from a persistent super-class
- ✓ support of complex relationships: 1-1, 1-many, many-many
- ✓ Enterprise JavaBean support
- ✓ mapping flexibility with 15 different mapping types, including object/relational mappings with Oracle 8i
- ✓ TOPLink Builder -- map objects visually
- ✓ sophisticated object-level querying
- ✓ full inheritance support
- ✓ mapping objects to multiple tables
- ✓ store multiple objects in a row (aggregate)
- ✓ "just-in-time" database reads
- ✓ stored procedures, custom SQL
- ✓ optimistic and pessimistic locking
- ✓ object-level transactions; no need to explicitly write objects
- ✓ application server integration and support

The TOPLink technology has been in use since 1992 and was first released as a product in 1994. It provides a robust, feature-rich framework that can be re-used on multiple projects. TOPLink is in production world-wide in industries such as insurance, banking, transportation logistics, manufacturing, aerospace, automotive, system integration, pharmaceutical and health care.

The solution: TOPLink for Java, Version 2.5

TOPLink for Java is a member of the TOPLink family of advanced object-to-relational persistence products. 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 an object database.

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 generate a new database schema from an object model. 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. With TOPLink, your organization can:

- ✓ adopt object technology without sacrificing the investment in relational technology
- ✓ create business solutions using Java objects and business rules, without having to focus on the infrastructure of the database; no SQL programming is required
- ✓ generate a new database schema from an object model
- ✓ save 30-40% of development costs -- by buying rather than building a persistence framework
- ✓ save future development costs by re-using the persistence framework and business objects across applications
- ✓ reduce time-to-market by leveraging mature technology

TOPLink has a solid design that provides a great deal of flexibility and performance. The experience gained from use on real projects is reflected in its rich set of features.

But I'm using JDBC already . . .

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 provides a layer on top of it. Application developers work with objects using TOPLink, rather than rows and SQL using JDBC calls.

Persistence framework: buy or build?

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 home-grown 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.

TOPLink and persistence

To achieve persistence, TOPLink does NOT force the class to sub-class off a persistent super-class. TOPLink also does not force a developer to implement a particular persistence interface or have database logic within the domain classes.

It is important for both scalability and maintainability that the persistence layer isolate business programmers from the database. Application developers should not write and maintain SQL statements, but instead focus on the business rules. TOPLink provides such isolation, allowing an architecture where the user interface and business objects no longer depend on where and how the data is stored.

TOPLink, object technology, and relational databases

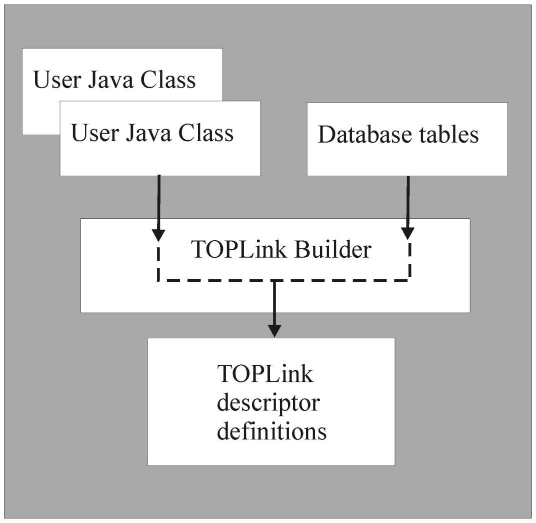
TOPLink's design goal is to be non-intrusive on the Java classes. Application developers work at the level of Java. When objects are read in, not only are the instance variables filled in with data but references to other objects are automatically maintained. The referenced objects are traversed by navigating 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, only to the TOPLink mappings. Java business classes can be re-used with a completely different database schema.

How does TOPLink do this?

TOPLink creates a set of meta-data “descriptors”, or mappings, that define 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.

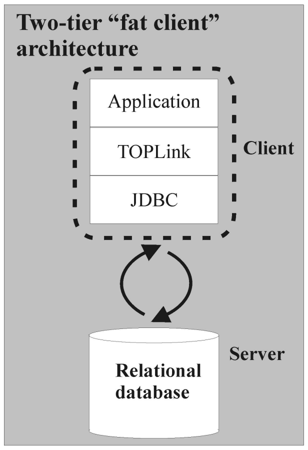
No SQL programming is required. The meta-data descriptors (mappings) are independent of both language and database.

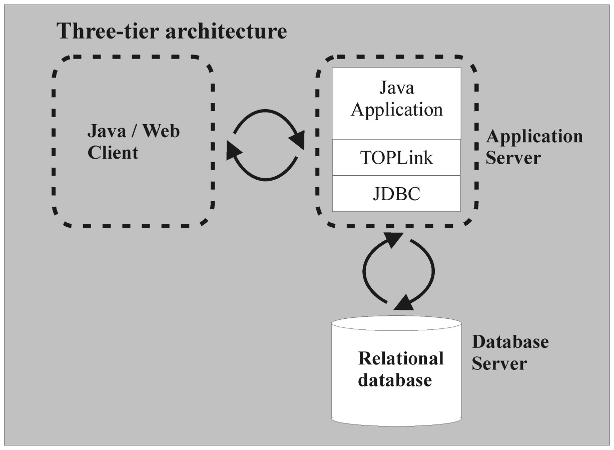


The TOPLink project architecture.

Two-tier, three-tier, and n-tier architecture

TOPLink can be deployed in a two-tier, “fat client” architecture or on an application server in a three-tier or *n*-tier architecture. TOPLink has no restrictions on the location of the objects and can be used with CORBA, RMI, HTTP, or other architectures.





TOPLink's features

TOPLink offers a comprehensive feature set that has evolved based on feedback from thousands of developers in more than 30 countries. Our client base reflects a wide cross-section of industries and diverse requirements.

Mappings

TOPLink for Java provides a rich mapping hierarchy to handle the wide variety of data types and references that an object model may contain. The mappings include:

- ✓ direct-to-field (strings, numbers, dates and so on)
- ✓ 1-1: object relationships
- ✓ variable 1-1 mappings: heterogeneous references
- ✓ 1-many: object relationships
- ✓ many-to-many: object relationships
- ✓ aggregate: a relationship to an object stored in the same row
- ✓ transformation: runs arbitrary code to transform values
- ✓ object type: simple translations such as “male” -> ‘M’
- ✓ type conversion: converting from database types to Java types, such as String -> Date
- ✓ serialized object: stores a serialized object in a BLOB
- ✓ direct collection: stores a collection of "basic" Java types

Object/relational mappings – Oracle 8i

TOPLink includes mappings to handle object/relational types defined in JDBC 2.0 and implemented in databases such as Oracle 8i:

- ✓ Array: handles Varrays
- ✓ Structure Mapping: object types
- ✓ Reference Mapping: refs

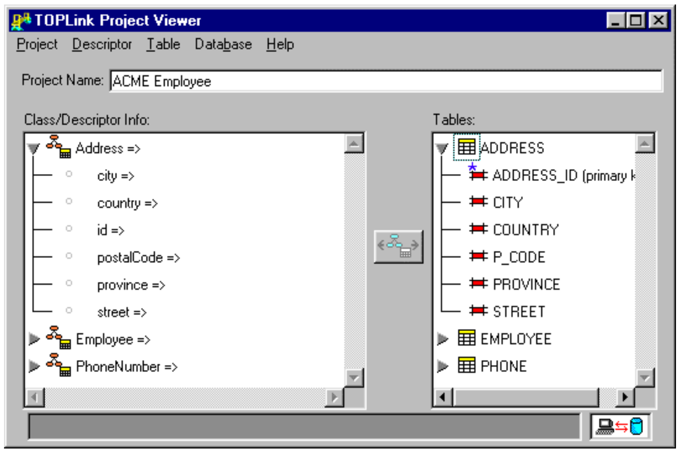
Visual TOPLink Builder

The TOPLink Builder is a development-time tool that assists developers in defining how objects map to the database. Java business classes are imported into the Builder and the relational database meta-data can also be read in through a JDBC connection. The object's attributes can be mapped to the corresponding database fields and the object's references are mapped through foreign keys.

The Builder's "neediness" capability can identify problems such as inconsistent mappings or missing information.

For an existing database schema, the Builder can generate mappings based on the Java object model. It uses type information, naming conventions and user-supplied information to determine the mappings. Ambiguous mappings can be resolved visually by the developer.

For a new database, a schema can be generated from the object model. The Builder can create the tables on the database or output a DDL script that can be edited by a database administrator.



Inheritance

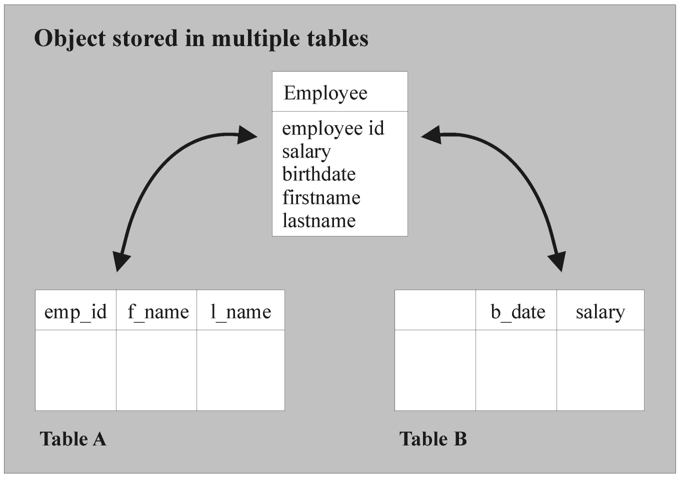
TOPLink fully supports inheritance. A subclass can reside in the same table as its superclass or be extended in a different table. TOPLink can support queries and relationships based on abstract classes. When reading and writing, TOPLink maintains the subclass information and retrieves and stores instances of the appropriate subclass.

Interfaces

TOPLink descriptors can be defined for an Interface. This permits queries and mappings to use Interfaces rather than concrete classes. Multiple implementers and implementees are supported. Variable one-to-one mappings allow a reference to be specified as an Interface to support heterogeneous types to be mapped.

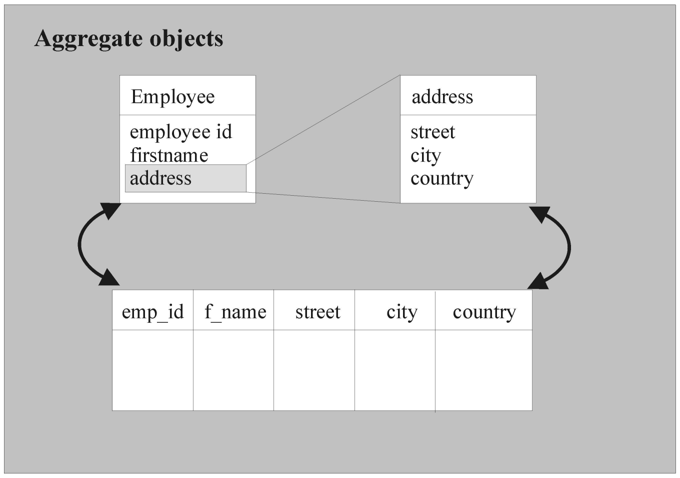
Multiple tables

There are several cases where an object's data may be stored in more than one table. A subclass may add data in its own table. An object built from a legacy database may have data spread across multiple tables. TOPLink handles these situations by allowing an object to map to an arbitrary collection of tables.



Aggregation

A table may contain data for more than one object. TOPLink uses an aggregation relationship to optimize the reading of objects stored in the same row.



Reading objects – query and expression objects

In non-object applications, accessing a relational database includes writing SQL. When building Java applications accessing a relational database, queries are best expressed using Java-like syntax.

TOPLink allows developers to build sophisticated queries at the object-level, rather than using SQL or JDBC. Developers have the ability to substitute custom SQL for any operation, but entire complex applications can be easily written without a single line of SQL in them.

Basic query

For simple queries, TOPLink provides an API to easily retrieve objects. The following line of code returns all the employee objects in the database.

```
employees = toplinkSession.readAllObjects(Employee.class);
```

To issue a more specific query, TOPLink has a hierarchy of expression classes. These objects may be built automatically by TOPLink or manipulated manually for maximum flexibility.

An expression builder is used to build queries. Expressions themselves are first-class objects. They use a Java-like syntax and can be manipulated and composed more easily than strings. The following basic expression is used to retrieve the employees who make more than $250,000 per year.

```
ExpressionBuilder emp = new ExpressionBuilder();
Expression richEmployeesExpr = emp.get(“salary”).greaterThan(250000);
richEmployees = session.readAllObjects(Employee.class,richEmployeesExpr);
```

Complex queries - joins

More complex queries that traverse an object's references can be used. Querying across 1-1 relationships between objects corresponds to doing joins on the database.

The following expression finds all the employees whose last name is 'Smith' and who live in New York. This query joins data across two tables, although on the TOPLink expression this is transparent as it only appears as an object reference.

```
Expression allSmiths = emp.get("lastName").equal("Smith");
Expression inNewYork = emp.get("address").get("city").equal("New York");
employees = session.readAllObjects(Employee.class,allSmiths.and(inNewYork));
```

Complex queries across object relations - anyOf operator

TOPLink supports object level queries across 1-to-many and many-to-many relationships.

The following query returns all of the employees who work on one or more projects that have a budget greater than $10,000,000 and live in the Cayman Islands.

```
Expression bigProjects = emp.anyOf(“projects).get(“budget”).greaterThan(10000000);
Expression offshore = emp.get(“address”).get("country").equal("Cayman Islands");
session.readAllObjects(Employee.class, bigProjects.and(offshore));
```

Query options

With any query, advanced customization features are available to set options such as outer joins, ordering, custom SQL, batch reading, cursored streams, parameter binding, statement caching, and many others.

Read performance – “just in time” reading

When a system handles arbitrarily complex relationships there must be a way to limit the depth to which relationships are followed when reading. Otherwise, when TOPLink reads an object, all of the related objects would also be read, then all of their related objects and so on. This would have a drastic effect on read performance.

TOPLink can delay reading the related objects until they are used. This “Just-in-Time” reading greatly improves read performance without affecting the object model or the application development.

If an object is never referenced, the resources required to read in the object are never used. This is completely transparent to the application using the object. The object and all of its references appear to be in memory at all times, when in fact TOPLink “faults” in the required data only when necessary.

In the following example an employee is read in. The address object is not actually read in until the `getAddress()` method is sent to the employee.

```
Employee employee = session.readObject(Employee.class);
/* Address object is not physically read in until this next line is executed */
Address address = employee.getAddress();
```

Read performance – object caching

The most expensive operation that occurs in a Java application using a relational database is a call to the database server. Object caching can dramatically improve performance on database intensive applications. Once an object has been read into memory it can be saved in the cache. This eliminates the need for a database call if the object is requested again. There are several different types of caches and they can be set on a class-by-class basis.

- ✓ weak reference caching (available in Java 2 / JDK 1.2)
- ✓ soft reference caching (available in Java 2 / JDK 1.2)
- ✓ full caching
- ✓ LRU caching
- ✓ no caching

Identity

In Java, every object has a unique identity (note that equality is different than identity). Relational databases support *external identity* by using and enforcing unique primary keys. To handle complex relationships and circular references correctly, TOPLink ensures that each unique ID (primary key) on the database corresponds to exactly one object in memory.

With a unit of work, an object's identity is maintained within the context at a particular unit of work.

Stored procedures and custom SQL

By default, TOPLink generates dynamic SQL. Instead of using the TOPLink generated SQL, custom SQL or stored procedures may be specified. (Note that no SQL programming is *required*.)

Projects may make extensive use of stored procedures for performance and security reasons. TOPLink allows stored procedures to be defined at the class level for querying, inserting, updating and deleting. This can be set at a per-class level or used on a per-query basis.

Read optimization - cursored streams

A query may return hundreds or even thousands of objects. Reading so many rows and building objects from them can severely affect performance.

TOPLink addresses this problem with cursored streams, which act like objects. Messages such as `read()` and `next(int)` return the appropriate objects. The `CursoredStream` reads and builds the objects in increments, so even though TOPLink behaves as if it already has all of the resulting objects, it does not read in the objects until instructed to do so. The resources used for instantiating the objects are amortized over the lifetime of working with the result set.

The faster initial response, with occasional brief delays later, leads to better perceived performance than a long initial pause. In some situations the user may only be interested in the first few objects and not need the entire collection.

With JDBC 2.0, scrollable cursors add the ability to move a database cursor forwards, backwards and to a precise position.

Read optimization - batch reading and joining

Batch reading allows a group of objects to be read in with a single database call. This allows more efficient reading and minimizes the calls required to the database. Batch read options can be set on a per-query basis or on a mapping.

Read optimization - partial object reading

If an object has many attributes that do not all need to be read in, partial object reading can be used. This allows individual attributes to be specified on a per-query basis and results in only the specified attributes being read in. If the entire object is needed later on, the object is refreshed and the remaining attributes are read.

Write optimization - batch writing

With batch writing, a group of update or insert statements can be batched together and sent with a single call to the database. Significant performance improvements can result from using this feature. Batch writing is supported through the JDBC 2.0 API in Java 2 and in JDK 1.1.x on most databases.

Write optimization - parameterized SQL

TOPLink supports full parameterized SQL statements, which allows statements to be prepared and cached. This can also be used for reading, but has the biggest impact when used on writes.

Object locking

Both pessimistic and optimistic locking strategies are available to address concurrency issues.

Pessimistic locking physically prevents another user from changing the data.

Optimistic locking raises an exception if another user has changed the data on the database since it was read in. A flexible optimistic locking policy can be customized to address a variety of implementations.

Unit of work - object level transactions

TOPLink supports object-level transactions called “unit of work”. A unit of work tracks and manages changes to a group of changes at the object level. It handles all of the updates automatically and writes the objects that have changed when the unit of work is committed. Only the attributes of the objects that have changed are written to the database, resulting in minimal update statements being generated. The user also does not have to keep track of the specific changes to the objects as it is automatically tracked for them. If the unit of work is released, the objects are left at their previous state.

Units of work can be nested to several levels. Only when the outermost unit of work is committed does TOPLink attempt to write the changes to the database. Parallel units of work are also supported. Object identity is maintained within each unit of work.

Each unit of work has its own copy of the objects being edited. Changes are not seen by other clients or units of work until they are successfully committed to the database. Upon commit, the changes to the objects are merged back into the shared object cache.

An actual database transaction is not started until the unit of work is committed. The duration of database transactions is kept to a minimum. This permits shared database connections to be used more efficiently and minimizes the potential for database deadlock situations.

Database transactions

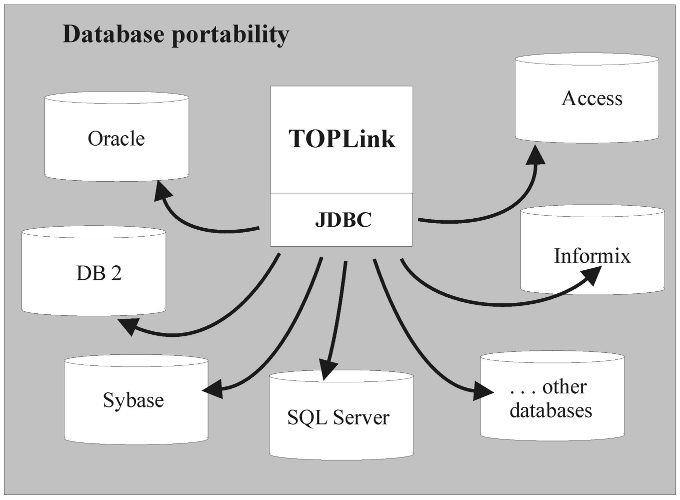
TOPLink automatically manages database transactions when using units of work. Database transactions can also be specified and controlled explicitly by calling the simple transaction protocol supplied.

```
beginTransaction();
rollbackTransction();
commitTransaction();
```

The default behavior is to not start a database transaction until a TOPLink unit of work is committed. This can be over-ridden by explicitly starting a database transaction.

Database portability

TOPLink is built on JDBC and is portable across any JDBC-compliant database, including Oracle, DB2, Sybase, SQL Server, Informix and Access. Any differences between database vendors are handled by components within TOPLink.



Mainframe and other data sources

TOPLink's flexible framework allows access to other data sources such as legacy data on a mainframe. A customizable extension framework is available to address these types of specific needs. Please contact The Object People (info@objectpeople.com) for more details.

Application server features

Deploying in a three-tier architecture requires some additional features:

- An object cache that can be shared by all clients.

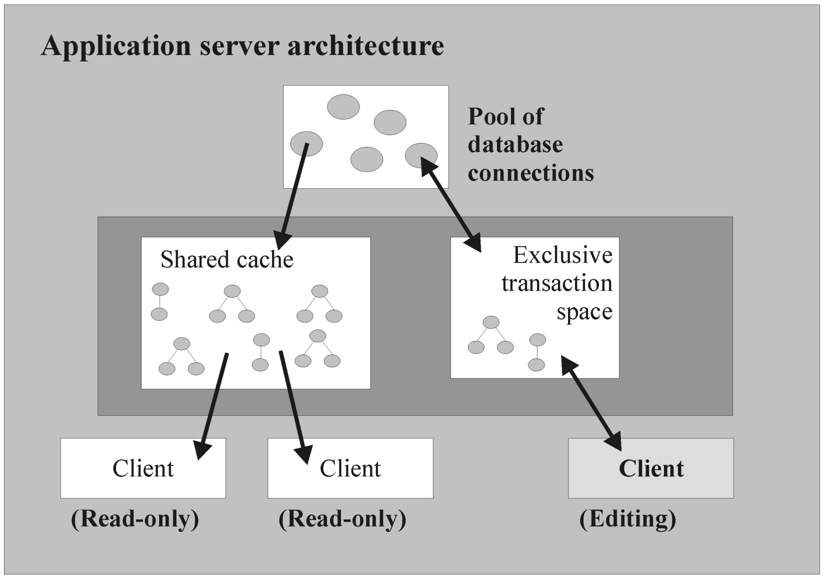
  This can greatly improve performance for objects that are frequently read but seldom modified.

- An exclusive transaction space.

  When a client begins a transaction or unit of work, an exclusive transaction space is managed for that client. Changes to the objects are not seen by other clients until the transaction is committed.

- A configurable pool of database connections.

  Not every client requires a dedicated database connection allocated for it on the server. The TOPLink server manages a configurable pool of database connections. With this pool, TOPLink makes much more efficient use of computing resources.


Application server architecture
Pool of database connections
Shared cache
Exclusive transaction space
Client
Client
Client
(Read-only)
(Read-only)
(Editing)

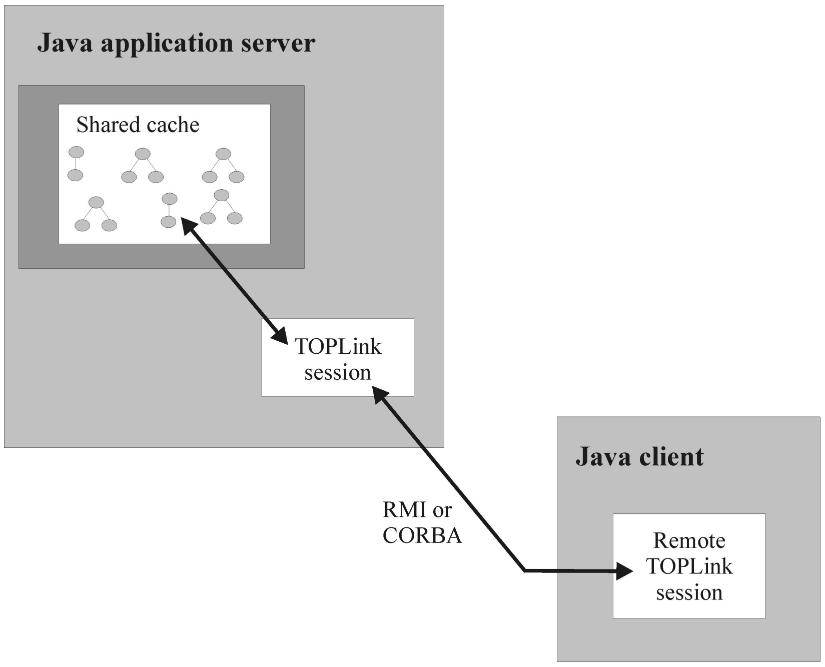
Client-side TOPLink functionality

For three-tier or *n*-tier architectures where the client and server are written in Java, remote TOPLink sessions are available to use on the client. The remote session is a fully functional client-side TOPLink session that communicates with the server-side session through RMI or CORBA. It handles full object replication and synchronization from the server to the client.

The remote TOPLink session supports the full functionality of a normal TOPLink session including:

- client-side caching and object identity
- client-side units of work with nested and parallel support
- remote proxies for on-demand loading of object relationships from the server
- complex querying support on the client

The client-side functionality allows for three-tier applications to be easily ported to a three-tier architecture using RMI or CORBA. For applications that are running Java on the client and server and want to transparently replicate business objects on the client, the TOPLink remote sessions make this trivial to build.



Java 2 (JDK 1.2) and JDK 1.1 compatible

Either Java 2 or JDK 1.1.x can be used with TOPLink for Java, Version 2.5. The Java 2 version supports new features such as:

- ✓ weak and soft reference caches
- ✓ access to private and protected variables through reflection
- ✓ JDBC 2.0 support
- ✓ collection classes

Java Transaction Service (JTS)

Distributed database transactions are supported with TOPLink's integration with JTS. TOPLink participates in a distributed transaction as a synchronized object. The transaction controller provides a distributed transaction aware database connection for TOPLink to use for a particular transaction context.

Enterprise JavaBeans (EJB)

TOPLink for Java can be used with Enterprise JavaBeans (both session beans and entity beans).

With JTS integration supported, the full functionality of TOPLink can be used with session beans on any compliant EJB server.

TOPLink can also be used with entity beans using bean-managed persistence, although not all of the functionality of TOPLink is applicable in a bean-managed environment.

For TOPLink's full object/relational mapping functionality to be used in an entity bean container-managed persistence environment, the TOPLink EJB server-specific product line must be used. Server-specific products are necessary as the integration is dependent on the server/container API, which is not part of the EJB specification and is vendor-specific.

TOPLink for WebLogic is the first integrated TOPLink EJB product and provides value-added features that go beyond what is outlined in the EJB specification. Other EJB servers will be supported in the future. For details on the TOPLink for WebLogic product, please refer to the TOPLink for WebLogic white paper.

TOPLink support and documentation

TOPLink for Java is accompanied by extensive documentation. The documentation set includes a user/reference manual, installation guide, tutorial guide and a troubleshooting manual.

TOPLink technical support provides responsive, thorough service. The support team is staffed by TOPLink developers and consultants who are intimately familiar with the product.

TOPLink services

TOPLink provides a powerful framework for developing applications, but persistence is still a challenging problem. Mission-critical Java applications have rigorous requirements and demands.

The Object People can provide experienced TOPLink consultants world-wide for on-site assistance. The TOPLink staff has significant experience working on a wide variety of enterprise-level projects.

Proof-of-concept consulting programs are available globally. A TOPLink expert works directly with the development team for a 3 - 5 day period to rapidly introduce TOPLink and build an initial prototype. This is an excellent way to expedite the learning of TOPLink's advanced feature set and capabilities.

The Object People also provides a 3-day course, “TOPLink Fundamentals”, which provides a rapid training curriculum for developers using TOPLink. Advanced TOPLink workshops are available that can be customized to address a particular area (CORBA, EJB, performance tuning, and so on). This course is available at The Object People offices or on-site at the customer location.

Summary

The object-to-relational impedance mismatch is a significant challenge for organizations adopting object technology. TOPLink for Java assists developers in delivering enterprise-level Java applications with relational database access.

The mature, robust persistence framework provided by TOPLink significantly reduces the time-to-market and development effort required for building enterprise Java applications.

Additional information

The Object People has been providing object-oriented services and products to Fortune 500 companies worldwide since 1989. Our TOPLink family of products includes TOPLink for Java, TOPLink for Smalltalk and TOPLink for WebLogic. We are very successful in assisting our clients deliver their mission-critical applications. The world headquarters are in Ottawa, Canada with regional offices in Raleigh, North Carolina, Germany, and England.

For more information on the TOPLink family of products or The Object People's Education and Consulting services please go to our website: http://www.objectpeople.com, or e-mail info@objectpeople.com. You may also contact our sales office nearest to you.

Notes