THOUGHT

The Dynamic O/R Mapping™ Company

CocoBase® Enterprise O/R

Technical Paper

QUERYING CONCEPTS

Dynamic Universal Querying™ with CocoBase®

For EJB and Java Applications on the J2EE, J2SE, and J2ME Platforms.

- Use with EJB-QL, a standards based, object-oriented syntax.
- Use with CBQuery Builder API, a SQL like API for advanced queries.

www.thoughtinc.com
US HEADQUARTERS / 657 Mission St. Suite 202 San Francisco, CA. 94105 / Tel. 415-836-9199 / Fax. 415-836-9191
Latin American Offices / Rua Professor Luiz Sanches Bezerra da Trindade, 69, Sala 301 Centro - Florianopolis - SC CEP Brazil 88015-160 / Tel. 55 48 2247783

INTRODUCTION

CocoBase® Enterprise O/R, Version 4.5 includes important new innovations for the Java database developer. The 4.5 version of CocoBase® provides powerful *Dynamic Universal Querying*™ for the J2EE, J2SE and J2ME platforms. Developer's can now benefit from an excellent and easy-to-use solution for dynamically querying application data in Java.

The query layer is built on the "Dynamic Object to Relational Mapping" architecture of CocoBase®. This extends for the developer all the benefits of CocoBase®, making for a querying system that decouples the application object from a particular database structure or particular SQL syntax, which allows data objects to be easily generated, maintained and reused. This furthers the ability of CocoBase® to cut up to 85% of the cost of database access development.

The CocoBase® querying layer can be used with EJB-QL, a standards-based, object oriented syntax as defined in the EJB 2.0 specification. It can also be used at an API level by with the CBQuery Builder which looks and works almost exactly like SQL. The developer has the choice to use either declarative-type querying with EJB-QL or procedural-type querying using the CBQuery Builder API to implement *Dynamic Universal Querying*™ in their Java applications.

The highlight is combining the simplicity of EJB-QL with the dynamic nature of the CocoBase® O/R Mapping layer. It results in unlimited querying for the user without the time-consuming work of pre-coded and pre-compiled SQL statements or custom finders. With the multitude of Object Query Languages that generally only work with one component architecture, THOUGHT Inc.® felt it was important to choose a single query syntax that was powerful enough to issue meaningful queries and simple enough that most developers could easily learn to use it. EJB-QL fits these requirements very well and the CocoBase® implementation uses the syntax for any type of Java persistence and not just Enterprise Java Beans (EJB).

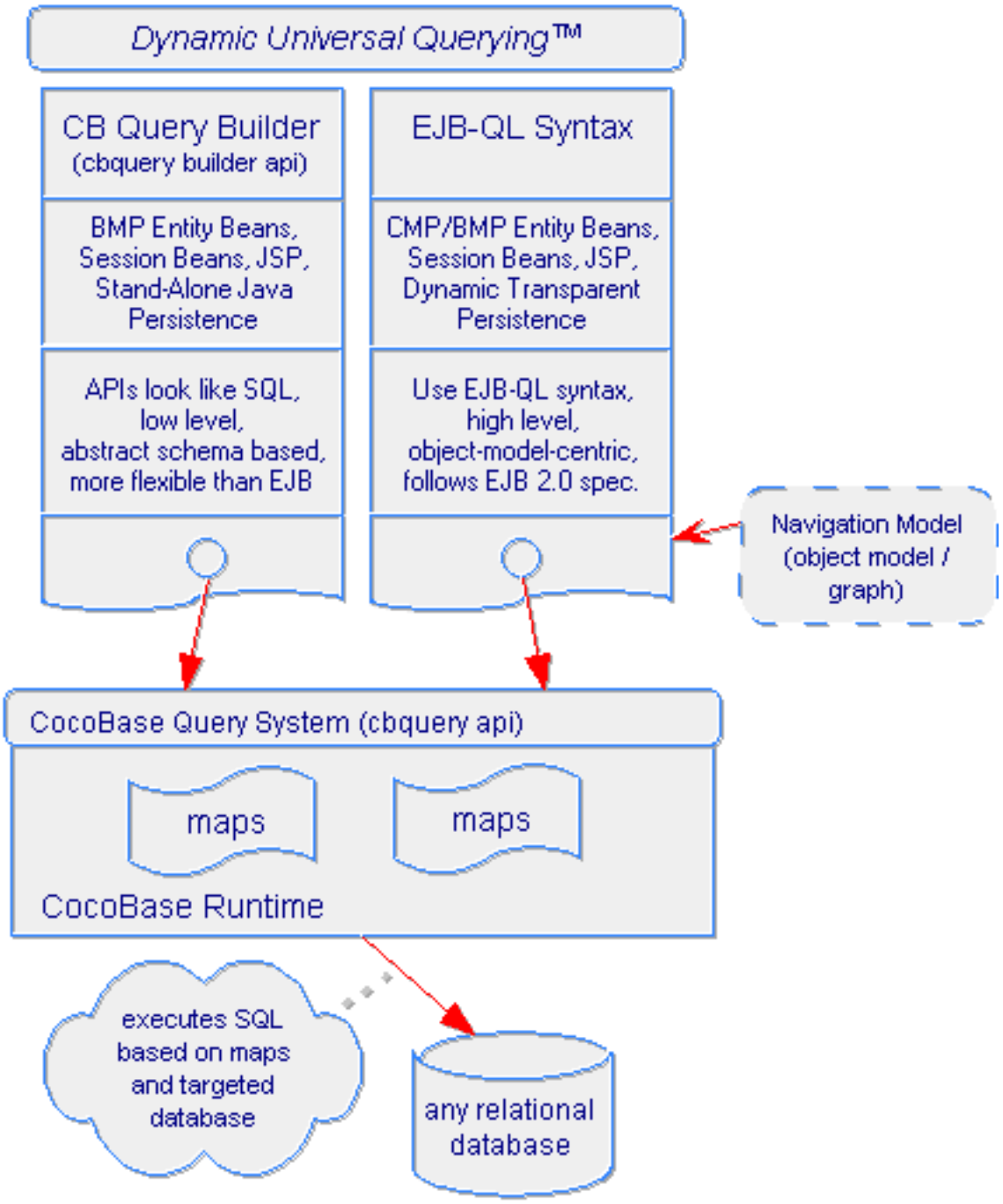
Highlighted Benefits:

- Powerful and Easy to Use Querying system for issuing Meaningful Queries.
- One Querying system for use with CMP & BMP Entity Beans, Session Beans, JSPs, Dynamic Transparent Persistence and Stand-Alone Java persistence.
- Provides Unlimited Querying (based on the mapped data) to Application Users without all the work of pre-coded / pre-compiled SQL statements or custom finders.
- Query system is architected to provide powerful queries with a rapid response time.
- No changes to the Query are needed when changes are made to the database.
- Further increases security of data in the database by limiting access to only the data represented in the map.

Two Ways To Express a Query:

- Declarative-type, standards-based Querying using the EJB-QL Syntax as specified in EJB 2.0.
- Procedural-API type Querying using the CocoBase® CBQuery Builder API, looks almost exactly like SQL, and is based on an "Abstract Schema (i.e. maps)."

Diagram of CocoBase® Querying system:



DYNAMIC FLEXIBLE NATURE OF THE COCOBASE® QUERYING SYSTEM

Understanding the CocoBase® Dynamic O/R Mapping™ Architecture

CocoBase® removes database specific code and predefined SQL from the application source code and instead, relies on user defined database maps to generate application specific SQL at runtime. CocoBase® maps provide a template for dynamically generating the structure of the SQL, and query values are programatically specified and inserted at runtime. This architecture decouples the application object from a particular database structure or particular SQL syntax and allows data objects to be easily generated, maintained and reused. It also centralizes database and object model maintenance since maps stored in database repositories can easily be edited or replaced, and any changes are instantly reflected throughout the enterprise.

Typically, O/R tools are often implemented by statically coding SQL directly into an application, or by tightly coupling the O/R layer to a single class hierarchy, which can be very fragile, or may result in limited reusability and difficulty in maintenance. In contrast, the CocoBase® dynamic mapping layer allows data maps to be shared, changed on the fly, and evolved quickly to meet the needs of the enterprise. The CocoBase® O/R mapping layer also provides many performance and scaling optimizations that can easily be configured to best fit specific application needs without requiring application regeneration and redeployment.

Understanding *Dynamic Universal Querying*™

CocoBase's® *Dynamic Universal Querying*™ runtime-based solution lets developers construct useful & meaningful 'object queries' based on an abstract object model (that is represented in a set of maps), instead of hard-coded to the physical table structure as is typically done. Because a CocoBase® map binds an object structure to a relational structure in real-time, different relational table structures can be used with the same object structure as long as the map expresses how this should occur. This means that a single object model can be used on a variety of different optimized physical layouts without recompile or redevelopment of application code. Simply create a map in CocoBase® and when the application connects to the database, the correct runtime mapping will transparently occur. Because the basis of the query is the map, developers have a lot of flexibility to fine tune exactly how the resulting SQL will be generated, even including specific code required by the database in the map. And developers can protect their investment by building queries at a more abstract level, instead of low level SQL or static EJB-QL.

ACCESSING COCOBASE® *DYNAMIC UNIVERSAL QUERYING*™ WITH EJB-QL

The CocoBase® *Dynamic Universal Querying*™ system can be accessed using the EJB-QL syntax (an object query language created for the J2EE Container Managed Persistence architecture). This new approach is profoundly different from how developers are accustomed to using EJB-QL. It is a decidedly more dynamic and runtime based architecture. The typical querying systems now available require that queries be pre-compiled prior to deployment and they become statically 'converted' into SQL. This process rigidly binds an object query to a specific relational database structure that severely limits any cross database portability. This also limits the ability of the database administrators (DBA) to evolve the relational schema as needed. Now developers will not have to know in advance,

for example at the creation of an Entity Bean, all of the query options that the user will need.

The CocoBase® *Dynamic Universal Querying*™ system lets the client developer or even end user issue an EJB-QL Query at runtime! The same Entity Bean can be re-used extensively by simply retargeting to another relational database, which may have an entirely different naming convention and table layout, where one is normalized and the other de-normalized. With CocoBase® the same bean would work unchanged and would not require redeployment. This would be accomplished by simply retargeting the JDBC driver. With most built-in CMP solutions, this functionality is not even an option. The beans would have to be re-written for each use to reflect the different table structures of the database which is a costly and time consuming prospect.

Using the EJB-QL syntax with the CocoBase® *Dynamic Universal Querying*™ system is not limited to just querying against CMP Entity Beans. It can be used with BMP Entity Beans, Session Beans, JSPs, Servlets, and stand-alone Java persistence. The CocoBase® *Dynamic Transparent Persistence*™ architecture has also incorporated this new querying system.

CocoBase® includes an EJB-QL Console tool that allows developers to design and experiment with their queries (see below for screen shot). It is integrated with the CocoBase® graphical user-interface "CocoAdmin" and allows input of free-form EJB-QL with a real-time response. It is also available as a command prompt level as well. These allow developers to directly perform Object Oriented queries on the CocoBase® repository and verify the results against the database. This is a powerful tool that developers can use to quickly and effectively check their queries.

EJB-QL Examples

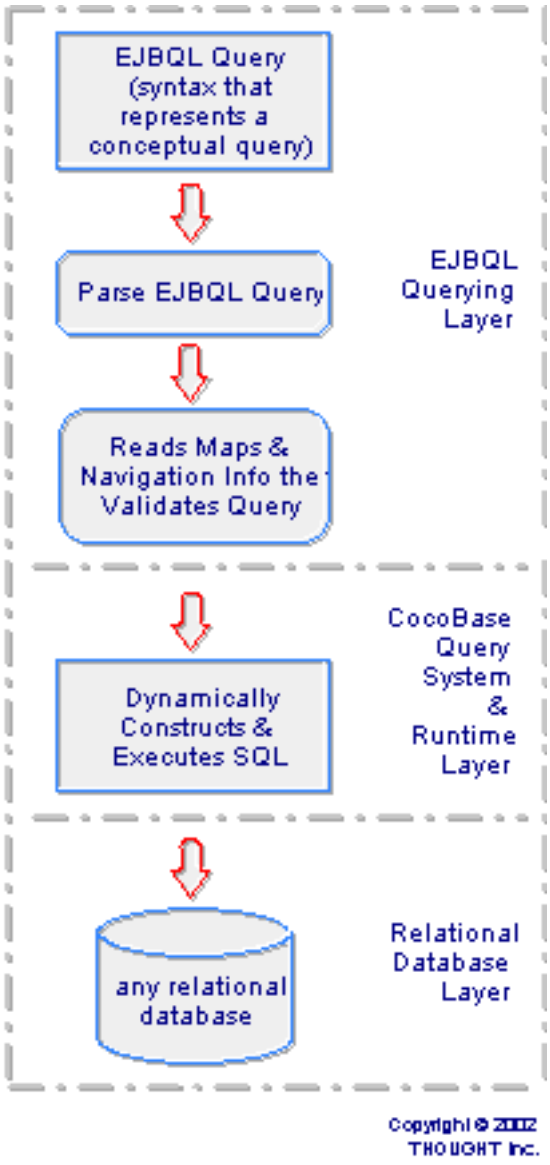
Here is an example of using CocoBase's® *Dynamic Universal Querying*™ with a Finder (notice that you do not need to put in a finder for every possible iteration of the query which saves a lot of time):

```
Vector params = new Vector();
params.add(getId());
Enumeration customerEnum = customerHome.findByQLQuery(
"SELECT OBJECT(c) from Customer c where c.id = ?1 ", params);
// step through enumeration
```

Here is an example with the EJB-QL syntax actually used directly in the code:

```
EJBQLQuery ejbQuery = newEJBQLQuery(cocoDataSource,nav.getModel());
ejbQuery.compile("SELECT OBJECT(c) from Customer c where c.id = ?1 ");
ejbQuery.setExtent(com.mypkg.Customer.class);
ejbQuery.setParameter(1,myCustomer.getId());
pkv = ejbQuery.execute();
```

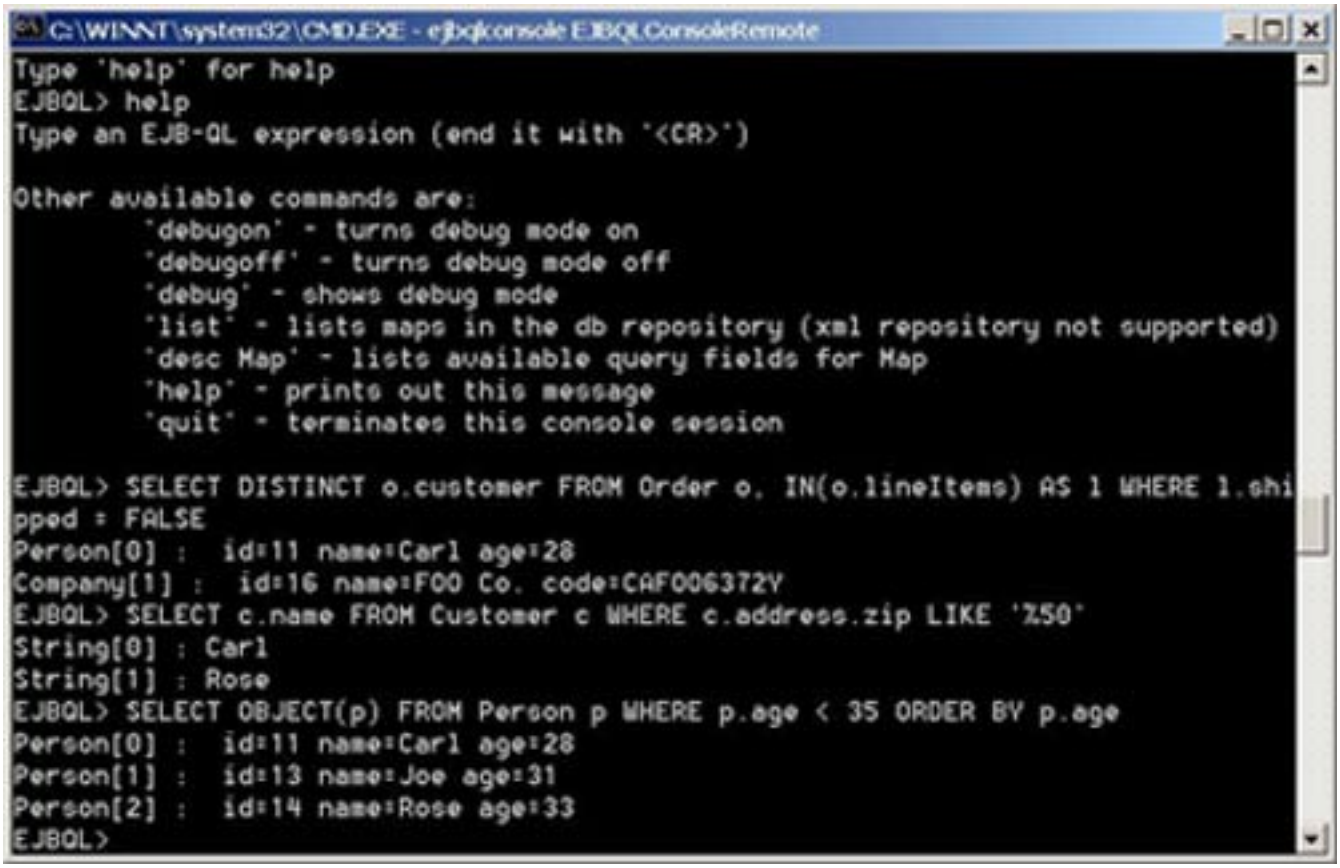
Diagram of CocoBase® *Dynamic Universal Querying*™ Using EJB-QL:



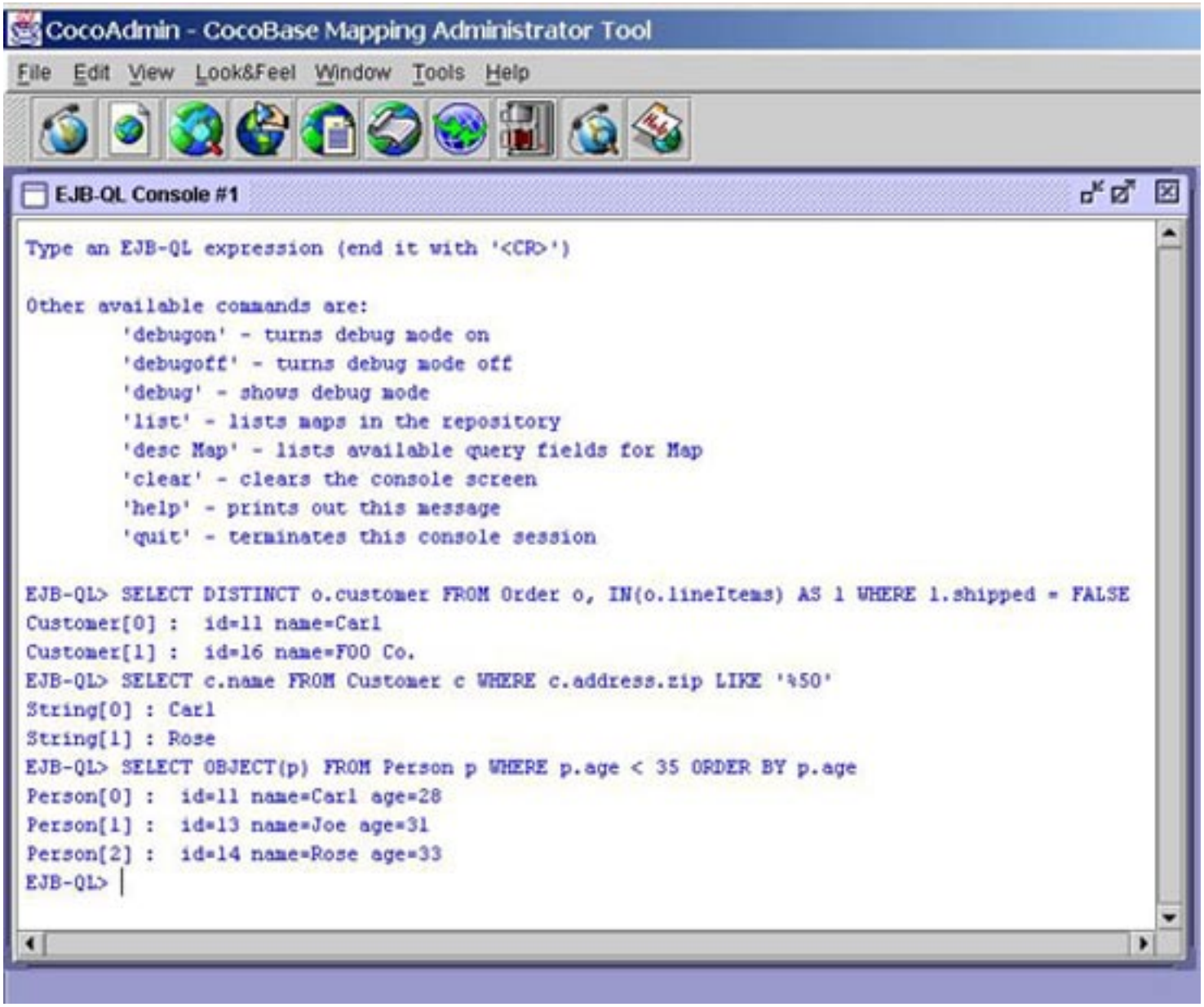
CocoBase® EJB-QL Console Tool For Real-Time Verification of Queries

CocoBase® includes an EJB-QL Console tool that allows developers to design and experiment with their queries (see below for screen shot). It is integrated with the CocoBase® graphical user-interface "CocoAdmin" and allows input of free-form EJB-QL with a real-time response. It is also available as a command prompt level as well. These allow developers to directly perform Object-Oriented queries on the CocoBase® repository and verify the results against the database. This is a powerful tool for developers to quickly and effectively check their queries.

Screen shot of the CocoBase® EJB-QL Console at the Command Prompt Level:



Screen shot of the CocoBase® EJB-QL Console in CocoAdmin GUI Administrator:



ACCESSING COCOBASE® *DYNAMIC UNIVERSAL QUERYING*™ WITH CBQUERY BUILDER

For developers who do not choose to use EJB-QL, the CocoBase® *Dynamic Universal Querying*™ system is also available at an API level. The CB Query Builder provides method calls in the form of APIs to program any needed "advanced" queries. This would be used when the standard CocoBase® Query system "Query by Example" does not meet the needs of the developer.

The CBQuery Builder APIs are designed to look almost exactly like SQL. In fact, if you know how to use SQL then using the CBQuery Builder API is almost trivial to learn. The real difference from using the APIs versus SQL, is that the API is focused at the CocoBase® map level and not directly at the database level of tables and rows. This provides the developer a high degree of flexibility to execute a query in a desired manner with fine-grained control over the results.

This is Procedural-API type querying that is based on an "Abstract Schema" and is represented in the CocoBase® Map that defines where the data is held in the database and how it is accessed. This is excellent for low-level querying that is focused on the specifics of creating custom queries that span relationships between maps. Compared to using EJB-QL, it does not require parsing and eliminates the need to compile the query.

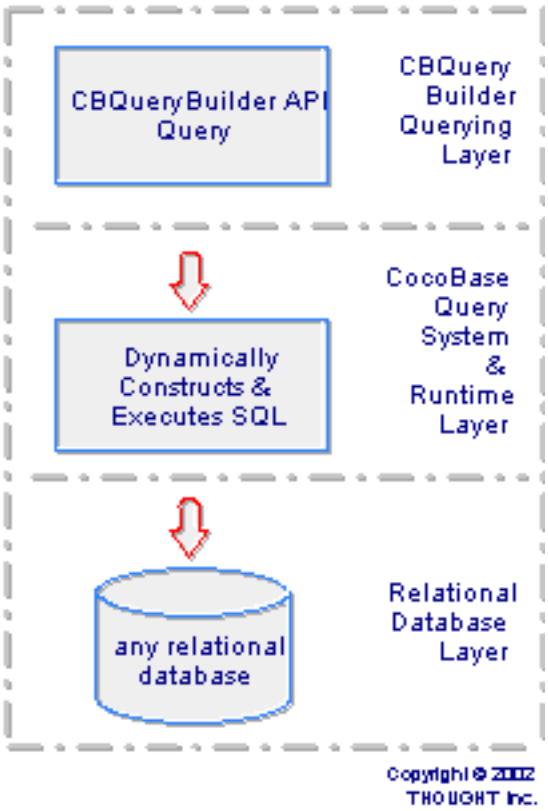
The CBQuery Builder can be used in any Java environment or architecture. It is simply a Java API to query against the database, just like JDBC. For example, it can be used with BMP Entity beans, Session beans, JSPs/Servlets as an alternate way to implement finder/selector methods or even with custom session beans. Examples of when the CBQuery Builder API could be a benefit are reporting, high through-put queries such as large result sets, impromptu queries where object model management is not important, and where the limitations of querying under the J2EE EJB specification is not convenient. Where as EJB-QL is constrained by the physical navigation and object model, using the CBQuery Builder is much more free-form. It doesn't require an object or navigation model that matches the query syntax for increased flexibility.

CBQuery Builder Example

Example of Using CBQuery Builder API to Execute a Query;

```
Vector result = qb.select("Customer.NAME","Address.STREET","Address.ZIP")
                  .as("NAME","STRET","ZIP")
                  .from("Customer").innerJoin("Address").on("ADDRESS_ID","ID")
                  .execute();
```

Diagram of *Dynamic Universal Querying*™ using the CBQuery Builder:



CHOOSING WHETHER TO USE EJB-QL OR CBQUERY BUILDER

Choosing to use EJB-QL or CBQuery Builder depends mostly on the goals of the developer, more than the actual architecture of the application. Please note that both alternatives are valid for use with any Java architecture.

Using EJB-QL can be considered a "best practice" use of CocoBase® for the following cases;

- Used at a higher-level programatically.
- Is a declarative-type language.
- Object-Oriented approach (i.e. data navigation is through path expressions, no joins are needed, collection and object value comparison expressions are allowed, etc.).
- Standards-based syntax that is part of the J2EE EJB specification.

However, use of the EJB-QL with the CocoBase® *Dynamic Universal Querying*™ system requires;

- Developers need to know or learn EJB-QL which is fairly simple.
- Requires use of a navigation model.
- Follows the Java naming convention that CocoBase® uses to do Java name conversion.
- Parsing and compilation step required prior to execution of query.
- Queries need to conform to EJB-QL query syntax guidelines.

On the other hand, the CBQuery Builder can be considered a "best practice" use of CocoBase® when;

- No object / navigation model is available or used.
- Developers are more comfortable with SQL-like algebra (projections, joins, cartesian product) and don't want to know about the EJB-QL object-oriented approach.
- When more flexibility is needed than is provided by EJB-QL (i.e. the query cannot be expressed in EJB-QL due to language constraints).

CLOSING

THOUGHT Inc.® innovated the world of Object to Relational Mapping with CocoBase® by introducing the concept of *Dynamic Object to Relational Mapping*™. The new 4.5 release of CocoBase® expands on this legacy of innovation by providing truly *Dynamic Universal Querying*™ (runtime-based) that is accessed with both EJB-QL, a standards-based Object Querying syntax from the J2EE EJB specification, and the CBQuery Builder API by using the CBQuery Builder. The dynamic, runtime-based approach of CocoBase® offers extensive deployment and re-use options that are generally less expensive to develop and maintain. In the current economic environment, a tool feature such as this that increases both efficiency and effectiveness becomes critical to the success of the enterprise developer.

About CocoBase® Enterprise O/R

CocoBase® Enterprise O/R, Optimized for J2EE, J2SE and J2ME Customer Success, solves the Object to Relational impedance mismatch. It virtually eliminates the need to hand-code database access for EJB and Java Applications. This can directly decrease up to 85% of the cost of database access development for enterprise customers faced with deploying fine-grained / coarse grained, simple to complex relationships in company applications.

About THOUGHT Inc.®

THOUGHT Inc.®, the Dynamic O/R Mapping™ Company, architects of CocoBase® Enterprise O/R, was formed in 1993, and subsequently revolutionized object to relational mapping technology with landmark solutions and industry leadership. More information on THOUGHT Inc.® can be obtained online at WWW.THOUGHTINC.COM or by calling, (415) 836-9199.

LEGAL NOTICES