THE EVOLUTION OF A DISTRIBUTED, FEDERATED OBJECT QUERY ENGINE

Leon Guzenda (leon@objectivity.com)
Chief Technology Officer, Objectivity, Inc.
www.objectivity.com

Abstract

This White Paper describes the evolution of the general purpose Objectivity/DB object database from a homogeneous, navigational object access tool to a distributed, federated (heterogeneous) query engine.

Introduction

Object Databases have advanced significantly over the past twenty years. Early products, including Objectivity/DB, were released with relatively crude query capabilities. This was mainly because the applications using them at that time primarily required navigational access. Modern applications need a wide variety of query mechanisms, ranging from navigational, through textual and geospatial to ontology driven algorithms. Objectivity/DB has been used to build some of the World's largest (Petabyte scale) databases and mining them requires advanced query capabilities. This White Paper describes the new Objectivity Parallel Query Engine and its applicability in a high performance, scalable, federated query environment.

An Overview of the Architecture of Objectivity/DB

Storage Hierarchy

Objectivity/DB is a distributed object database management system. The logical storage hierarchy is shown in Diagram 1. Instances of objects of multiple classes can be grouped within containers, which are themselves grouped within databases in a federated database.

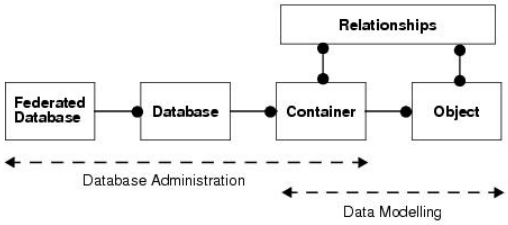

Diagram 1 – Logical Storage Hierarchy

A simple example of a federation would be one for a newspaper and magazine publishing company. Each database might contain all issues of a particular publication. Each issue would be in its own container, which would store all of the Section, Article, Paragraph, Sentence, Word, Image, StyleSheet and other objects for that issue.

The federated database contains one or more schema, a catalog of databases and other system information. Databases can be distributed around a network and can be replicated to multiple locations. Databases can be stored in a single operating system file or multiple files, each holding one or more containers.

Process Architecture

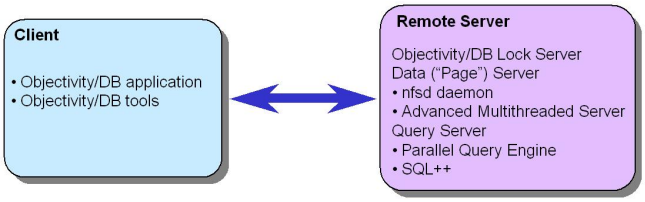

Diagram 2 – Process Architecture

Objectivity/DB distributes its processing tasks among clients and multiple kinds of server. Diagram 2 illustrates the major software components. The Objectivity/DB kernel library is linked with the client.

If the client is a single process, there are no other database users and all data is local then there is no real need for any other kind of process. The kernel will access local databases and handle transaction semantics. In practice, there are always multiple, concurrent clients and in most cases there are remote databases. Lock Server processes handle concurrency issues and Data ("Page") server processes access remote databases.

Clients can handle basic query processing (other than the granting of locks and reading remote disks) themselves, or they can request that the kernel break the query into smaller parallel tasks that are sent out to Query Servers close to the objects that need to be accessed.

Federated Queries

The Parallel Query Engine

Until recently, all databases in an Objectivity/DB federation had to be formatted and controlled by the ODBMS. This was not the original intent, as the architecture caters for heterogeneous databases, i.e. ones controlled by other DBMSs. The introduction of the Objectivity/DB Release 9.2 Parallel Query Engine in early 2006 paved the way for a powerful federated query mechanism.

Replaceable Components

Objectivity/DB is not distributed under an open source licensing scheme. However, it was originally conceived as a toolkit, with the ability to enhance, replace or omit various components. In practice, this turned out to be a difficult scenario to quality assure and support with the tools available in the early '90s, so the number of replaceable components has been limited to a few hooks in the Objectivity Open File System [OOFS] layer, which is a back-end for the Advanced Multi-threaded Server [AMS]. OOFS can be modified to access novel storage devices or file systems. It also has Generalized Security Architecture hooks, making it possible for sites to control access to their databases with existing mechanisms, rather than any imposed by Objectivity/DB.

There is a test suite for AMS and OOFS, so if a user modifies OOFS and can run the test suite satisfactorily then Objectivity Customer Support will provide full support for any problems (other than diagnosing issues in the modified or replaced code, of course) as if it were the standard Objectivity product. This offers the users the benefits of traditional maintenance and support licenses with many of the major benefits of open source licensing.

Parallel Query Engine Components

Diagram 3 shows the major components of the Objectivity Parallel Query Engine. The colored (or darkened) components are user replaceable.

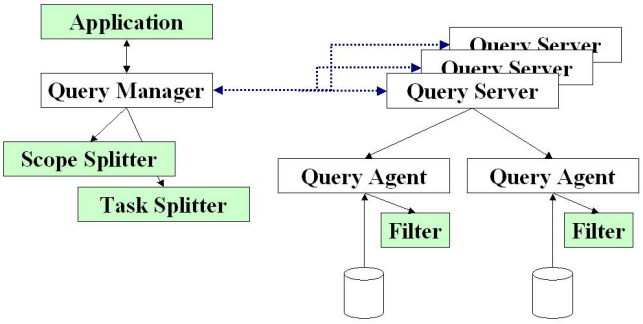

Diagram 3 – Parallel Query Engine

The client initiates a parallel scan iterator that specifies the scope (federation, database, container etc.) of a query and the predicates to be applied to object instances within that scope.

Unlike traditional database query optimizers, the application designers can substitute their own task and scope "splitters". Imagine a federation that has databases for each country in the World and that each database has containers for all major cities in that country. Each container has images and other information of interest to the users. The client might issue a query to find images of zoos in Europe. The range splitter would first look up a reference list of countries in Europe. This would return a list of databases or container identifiers.

The Parallel Query Engine would then send the query predicate and an individual database or container identifier over to the appropriate Query Server. The Query Server executes the query and returns qualifying objects. The Parallel Query Engine marshals the outgoing requests and incoming queues of objects (or object identifiers). The iterator returns a qualified object to the client application every time that it steps.

The Query Server also has a replaceable component, called the Filter. Suppose that the previously described query also specified that only objects representing zoos that are open on Wednesdays should be returned. The Filter can be any Objectivity/DB application, or it can access any other data source. It can do the additional qualification, eliminating unwanted objects.

The target database or container need not be controlled by Objectivity/DB. Any kind of external data source can be accessed. The only requirement is that the Filter return an object (or an object identifier) in a format that is intelligible to the client side Parallel Query Engine.

Summary

The mechanism described here turns Objectivity/DB into a powerful, federated object query engine that is user modifiable. Future releases will allow the replacement of Query Servers, supply frequently requested gateways and provide generalized Query Servers that can be called from non-Objectivity/DB clients, using Service Oriented Architecture and grid protocols.

References

[1] Witold Litwin, Abdelaziz Abdellatif, *Multidatabase Interoperability*, Computer, v19 n.12, pp.10-18, Dec. 1986

[2] Objectivity, Inc. *Technical Overview, Release 9*, January 2006.
http://www.objectivity.com/Misc/docs/oodb_techOverview.pdf

[3] *Iterator* - http://en.wikipedia.org/wiki/Iterator