Objectivity Case History

Customer Information

Customer: TekNow

Status: (X) SOLD ( ) Prospect ( )Integrator
Industry: Telephone Billing and Monitoring
Application Domain: Network management
Status: In Development
Platform: NT
Compiler: Version 5.0 VC++
Other Tools:

Buying Criteria

This company was looking for innovation, they needed a fast scalable network storage application for handling complex billing and market usage patterns for the cellular phone market. The application is also to store and count specific traffic and usage information for competitive pricing and positioning. The application in general takes advantage of Objectivity distributed architecture to provide a single logical view to a phone or device anywhere in the network.

The data model directly models the real world layout of complex telephone networks and switches. This includes terminals and exchanges private and public. The purpose of this program is to look up subscribers and then determine what billing codes and any other services like voice, email, fax or other digital communication options. Every object contains an adapter so basically everything is treated similar to the way plug-ins are loaded within a browser. Everything in this system can be unplugged and plugged in somewhere else if necessary. This feature is supported by the persistent capable classes and the C++ language.

The NT platform demanded several interesting engineering demands. The Dynamic Data language (.ddl) needed to be integrated as a Dynamic Link library (dll). To get the persistent definitions working caused a struggle with ooddlx preprocessor. Several files were used in addition to the Objectivity required files. This made the number of files per persistent class about five. With a projected class load of close to a hundred classes this implementation leads to some moans and groans from the client. The task of managing changes to the persistent classes also came up. Schema migration while in production was a major concern. Objectivity schema migration fits perfectly for this requirement.

The use of a persistent linked list and persistent rb-tree were both essential to this project. The linked list helped to manage the list of plug-ins and other assorted tasks. Where the large persistent rb-tree was used for the federation wide lookup for subscribers, and terminals. Both of these tasks needed to be developed for this project. Objectivity does not [Ed: did not at that time] supply federation indexing or a templated linked list for use in such applications. Data access patterns (FD indexing, event exposure, TMI) could use improvement.

Administration and the storage hierarchy also created a small struggle. The federation wide custom indexing scheme will handle the majority of the subscriber lookups. Other indexing and map strategies were used to access subclasses.

The customer was really impressed with the architecture and the ability to distribute databases. The one logical view from any client also was important. The vision and willingness for this company to embrace this new technology is one to follow. The flexibility in this particular business area will definitely be interesting to watch. Objectivity architecture will really shine in this application.

Why Objectivity

The dynamic performance characteristics, and the distributed storage capability. One logical view from any client was important. The computer as a commodity, scaling across several machines for efficiency and load balancing were also very important to the core business.

Contact Information

Objectivity Rep: John Jarrell
Customer Contact: Michael Morris
Phone:
Email: memorris@prime.net