Objectivity.

Using Higher Performance Databases In Advanced Intelligent Network Applications

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

Introduction: Providing advanced communication services is a major source of potential revenue for telecommunication companies (telcos). Advanced Intelligent Network (AIN) technology eliminates two important obstacles to providing these services. First, because AIN is an open standard, telcos do not get locked into long-term commitments to particular equipment vendors. Having a broad choice of equipment and supporting services increases competition and therefore decreases costs. Second, the AIN architecture decouples service processing from call switching. This flexibility allows telcos and third parties to quickly develop and deploy new services to take advantage of emerging market opportunities.

Advanced communication services are information intensive. To process calls that make use of these services, the various network nodes must have real-time access to data about each service and each subscriber. Network nodes must have access to this data 24 hours a day, 7 days a week, requiring high data integrity and data replication for high availability. Moreover, to compete for subscribers, service providers must have the ability to easily create and modify databases. While vendors of AIN equipment and services have a great degree of flexibility in building AIN implementations, certain ones have found that ODBMSs meet the above needs very well. Examining AIN requirements reveals that this choice stems from the natural match between the networking domain and the object-oriented paradigm, as well as the flexibility of ODBMSs themselves.

AIN Background: To understand AIN database requirements, it helps to first understand the AIN architecture and AIN services. Unfortunately, AIN's flexibility makes determining a complete set of database requirements applicable to all AIN implementations impossible. Because the AIN standard specifies only standard interfaces and a general topology, a service provider can customize the implementation at many levels, from network configuration to service processing. However, by examining the AIN architecture standard and an example AIN service, we can derive some general database requirements and identify potentially crucial issues.

AIN Architecture: The AIN architecture specifies a number of key elements in the network topology. Many of these elements require an associated database. While vendors do have flexibility in determining the capabilities of

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

the various elements, they must fulfill some basic responsibilities:

- **Service Switching Point (SSP)**. The SSP is a telecommunica tions switch that identifies calls requiring advanced service processing and forwards them to an appropriate SCP (see below). The SSP has a rudimentary database that allows it to properly identify calls, but does not generally contain data relating to service processing.
- **Service Control Point (SCP)**. The SCP handles calls requiring advanced service processing. To provide high performance, each SCP is usually configured to process calls for a single service. Normally, they are deployed in mated pairs that duplicate the same capabilities to ensure high availability and allow for load balancing. Each SCP has several processors. These processors may access a common database of service and subscriber or each processor may have its own database, depending on the implementation. When the SCP receives a request from an SSP, it may simply query its database for the necessary information and then process the call. For instance, some customers want calls to their 800 number routed to different physical locations depending on area code of the caller. In this case the, SCP would simply look up which physical location should receive the call. In other cases, the SCP may require further information from the user. To handle these calls, the SCP will route the call to an IP (see below) that can interface with the customer. For credit card calls, the SCP would route the call to an IP that could accept touch tone input. Once the IP had received the customer's credit number, the SCP would verify its validity and complete the call.
- Signaling Transfer Point (STP). The STP is simply an intelligent router that routes requests from SSPs to appropriate SCPs.
- Intelligent Peripheral (IP). The intelligent peripheral is a device that accepts input from customers. Examples of IPs are those that process touch tones and voice.

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

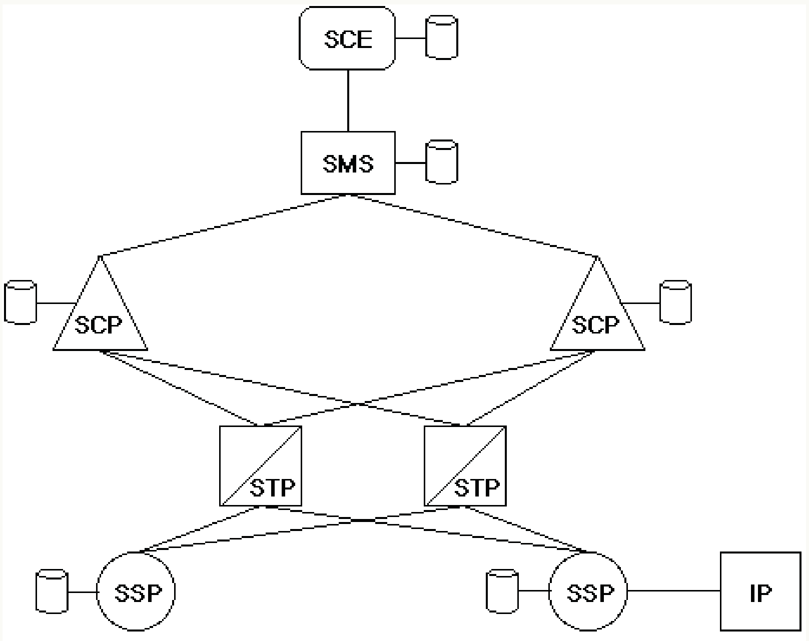

Figure 1 – AIN Architecture

- **Service Management System (SMS)**. The SMS is a non-real-time system responsible for managing the operation of SCPs. Each SMS can manage several different SCPs offering different services. For example, an SMS may control one SCP that handles a credit card service and another SCP that handles a virtual private network service. The SMS is responsible for configuring SCPs, provisioning new services, and updating SCP databases. The SMS stores all the data for each SCP in its master database. All admin istrative and customer updates to the SCP must pass through this master database. While the SMS and SCP databases contain the same information, they may use different formats. The SMS database requires maximum flexibility to allow for efficiently updating of a service and does not have a real-time performance requirement. On the other hand, the SCP database requires maximum performance. To meet the requirements of both data bases, some vendors may use differently structured databases for each. The SMS can still transfer differently formatted data to an

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

SCP because they communicate through a standardized interface independent of logical format.

- Service Creation Environment (SCE). The SCE is a non-real-time system for creating, testing, and deploying new services. The SCE has a sophisticated suite of tools for creating service processing logic and the different databases. The SCE has a sophisticated database of its own that contains common building blocks for services. By using these building blocks, developers can create new services in a fraction of the time it would take to create them from scratch. Since service providers can derive significant lever age from an SCE that facilitates faster service development, the implementation of this AIN element differs significantly from vendor to vendor.

Figure 1 provides a graphical view of the AIN architecture and depicts which elements have associated databases.

AIN Services: We saw in the previous section, that implementers of AIN systems have some flexibility in their design of AIN network elements. They have even greater flexibility with AIN services. From a database perspective, creating a new service requires creating a generic model of the data, or schema, that the SCP database will use for calls that use this new service. In most cases, the service developers can choose from several different data models. One model may be more flexible and allow for quicker updates. Another may be easier to develop and have better response time. One of the important advantages of the object-oriented paradigm is that it mirrors the way humans process information and provides the same flexibility. Figure 2 presents the schema for a credit card AIN service. In the figure, boxes indicate classes, which define the variables and procedures associated with specific objects. Lines indicate the possibility of a two-way link, called an association between objects belonging to the connected classes. The numbers in brackets indicate the minimum and maximum number of connections between objects of those classes.

This service has a fairly straight forward data model. The fundamental class in the service is the Basic Subscriber. For each instance of this class, the database stores various identification and billing information. An instance of

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

this class can in turn have connections to instances of three other classes: Auto Call List, PIN List, and Qualified Number List. These classes correspond to three different features of the service. An instance of Basic Subscriber will only have a relationship to instances of these other classes if the subscriber has purchased the corresponding feature. The auto call feature allows the subscriber to save time by specifying a short c ode that will automatically dial a frequently called number. The subscriber can specify up to 16 such codes. The PIN feature is simply basic credit card calling service. The subscriber can specify up to 32 different PINs. The qualified number feature protects the subscriber against unauthorized use by only allowing calls to certain numbers. A subscriber can specify up to 32 qualified numbers.

From the description of this application, we see two important things from a database perspective. First, we need f ast searching among objects and fast navigational from object to object. When we process a call, we have to search all PIN objects for a match to the sequence of numbers input by the customer. Then , we have to find the PIN List connected to that PIN object and the Basic Subscriber connected to that PIN List. Finally, we have to find the Qualified Number List connected to that Basic Subscriber's and then search among all Qualified Numbers connected to that list to make sure one of them matches the number dialed. We have to complete this same sequence of searches and navigations for all calls before any of the callers get frustrated and hang up. The second interesting thing about this schema is that it requires manipulating variable length data. The number of Auto Calls, PINs, and Qualified Numbers can vary. We don't want to allocate disk space to storing the maximum number if we don't have to.

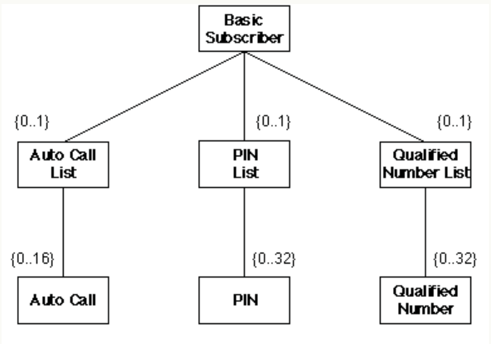

Figure 2 – Application Schema

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

These requirements generalize to all AIN services. AIN services are about making connections and taking different actions based on particular connections. A priori, this means moving from piece of data to another - navigational access. Furthermore, AIN services are about customization. People want to store information specific to their calling needs, information that may grow over time - variable length data.

AIN Database Requirements: In the previous section, we took an initial look at AIN database requirements. Now we'll make a more detailed investigation of four different categories of requirements and discuss how ODBMSs in general and Objectivity/DB in particular meet these requirements.

Reliability and Integrity: As mentioned above, AIN applications need to operate in a 24x7 environment, necessitating very high reliability. Mean time between failures and mean time to repair therefore become very important when selecting a database platform. The evaluation of different products should include a careful investigation into the observed value of these statistics at various customer sites. Objectivity/DB is so reliable that when a potential customer asked an existing customer how long it took Objectivity/DB to recover from a failure, the customer replied, "I don't know, it's never failed."

In addition to reliability, AIN applications require both data integrity and referential integrity. Data integrity measures a database's resistance to data corruption. Objectivity/DB and some other ODBMSs provides high data integrity through the use of safe object identifiers (OIDs) or object references. OIDs provide a level of indirection between the application and an object's location on data pages. In contrast, some products allow the use of direct pointers to data. Such pointers become invalid after a transaction commits and reusing them can lead to data corruption. Data corruption in AIN applications could prevent the completion of some calls at the very least and possibly the total failure of an SCP.

Objectivity/DB provides referential integrity through bi-directional associations. The database automatically deletes an association when either of the objects in the association is deleted. In our earlier AIN service example, if a

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

subscriber discontinued service, we would delete the corresponding Basic Subscriber object. Objectivity/DB would automatically delete the associated PIN List and its associated PIN objects. By propagating delete operations, Objectivity/DB prevents PIN objects with dangling references to Basic Subscribers from remaining in the database. If we did not propagate the delete operation in this case, a former subscriber that tried to use their old PIN would cause an application crash as the database tried to locate the nonexistent Basic Subscriber object on the other end of the connection.

Availability: Closely related to reliability and data integrity is the availability of the database. AIN applications require 24x7 availability, and cannot tolerate the database being unavailable. Objectivity/DB offers unmatched availability through its support of schema evolution, fault tolerant configuration, and data replication.

Objectivity/DB schema evolution provides a mechanism for changing the physical layout of objects in an application while the database continues to operate. For example, suppose that the example above was altered to handle voice or data communications. The new service would allow each Basic Subscriber to indicate whether each Qualified Number was for normal use or high baud rate data transmission. This is likely to involve a change to the Auto Call object, calling for the addition of a data member to indicate voice or data transmission. Objectivity/DB's schema evolution supports making such changes to the definition of persistent data in the application and in the existing database, without ever taking the database off-line.

Availability is also a requirement in the face of underlying system or network errors. Objectivity provides the ability to divide a federation of databases into autonomous partitions, each of which will continue to operate despite the failure of servers or networks in other partitions. Further, replicas of a database may be created in each partition, providing not only better availability to the replicated data, but improved read performance as well. If a network problem interrupts service to one of the database replicas, another replica will transparently provide the requested data to the application.

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

Data Model: The greatest advantage offered by ODBMSs for AIN applications comes when we build the data model. In our example service, we saw that its data model required many associations among objects and the use of variable length data. Remember, this service was fairly simple. A more complicated service would make even greater use of these features. A relational database could not meet these needs and provide the requisite real-time performance because associations in relational databases, known as joins, are relatively expensive. When an application requires navigating many joins, performance degrades quickly.

As discussed above, bi-directional associations provide applications with referential integrity. They also provide applications a very fast means of moving among related objects. Two bi-directional associations, one from a Basic Subscriber to a PIN List, and one from the PIN List to a PIN, allow the application to go directly from a PIN to a Basic Subscriber or vice-versa. All AIN services where the subscriber is the caller will require this same navigation. To authorize use of the service, we need to collect a PIN. To take the appropriate action for the caller, we need to find the corresponding Basic Subscriber and its associated service information.

Object databases allow applications to handle variable length data by providing variable length array data structures. In our example, a PIN List contains a variable length array of PINs. If a subscriber only defines three PINs, the database only uses disk space for the three PINs instead of allocating space for all 32 possible PINs. This feature keeps databases smaller, decreasing disk space needs and increasing performance. Objectivity goes a step further by allowing objects to have multiple variable length arrays. Some products limit objects to just one. This limitation can force developers to either split an object into two pieces or write extra routines to combine two logical data structures into one variable length array.

Another important benefit of the object-oriented paradigm comes from its facilitation of reuse. When we talked about an SCE, we discussed how it provided building blocks for services. These building blocks are actually classes. For instance, we showed above how every service where the subscriber is the caller needs to have the same relationships among Basic Subscriber, PIN List, and PIN. An ODBMS allows us to build these

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

classes once and reuse them in every appropriate service. Having many reusable classes greatly decreases the time required to deploy services.

Transaction: While the data models of all AIN applications are relatively similar, transaction characteristics vary significantly among applications. For example, two of the most important transaction characteristics are the number of simultaneous readers and the number of simultaneous updaters that access the database. These characteristics can vary greatly depending on implementation choices. One Objectivity customer building an AIN application tried two different ways of configuring an SCP database. The first used eight server processes accessing eight separate databases with no overlapping data. In this model, there would be only one reader and one updater for each database. The second approach used four server processes accessing a single database. In this model, there would be many readers and updaters. Service providers may want to make this choice on a service by service basis to optimize for the unique calling pattern associated with each service. To meet these varying needs, Objectivity/DB allows developers to tailor the concurrency control characteristics of different applications.

Objectivity/DB allows flexible concurrency control through a locking mode called Multiple Reader, One Writer, or "MROW", that allows multiple readers and a single writer to access the same data at the same time. The readers view the object as it existed at the point of the last successfully committed transaction. The updater can update the object without interfering with existing readers. If the updater c hooses to commit the transaction, Objectivity/DB makes the new updates visible to everyone at the start of the next new transaction. This mode allows the maximum level of concurrent access to the objects in applications with many readers, significantly inc reasing performance. Service providers that chose a database architecture with a single database would find this feature extremely useful.

Objectivity/DB provides another flexible mechanism, called containers, that allows service developers to increase or decrease the level of locking granularity. Objectivity/DB locks entire containers of objects that are logically related. Containers may hold any number of objects and change size dynamically to accommodate new objects. Certain services may have a few

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

objects that get updated very frequently. A company that allocates 800 calls to different customer service centers may want to change this allocation dynamically. For these objects, the developer may want to have fine locking granularity to decrease concurrenc y conflicts. Putting each such object in its own container gives fine locking granularity. Unfortunately, this approach decreases locking efficiency because the server must process a lock for each object. Objectivity/DB allows developers to use increased locking granularity only when they need it. The vast majority of objects will be updated infrequently and will be updated in logical groups of objects. Putting many of these objects in each container yields high locking efficiency. Developers can choose the appropriate place on this spectrum on an object by object basis.

Performance: While flexible concurrency control is important to AIN application, performance is crucial. Objectivity/DB increases performance advantages through page clustering and efficient client cache. As mentioned previously, the data model for most AIN applications has many complex relationships among objects. A query to the database often requires the retrieval of multiple objects. The performance of the application is directly related to how fast the database can find objects and retrieve them from disk. Disk access is the single largest factor in determining how fast an application will run. Since the page is the smallest unit of disk access, whenever the disk needs to retrieve an object, it grabs the entire page on which the object resides. Consequently, the placement of objects on the disk greatly affects the performance of the application. Retrieving two objects on different pages requires two disk accesses. Retrieving a hundred objects from the same page requires only one. Objectivity/DB's page clustering allows a developer to achieve the latter outcome. When creating an object, the developer specifies another object that the new object should be near. In the case of our example service, we would want to put all Qualified Number object and their associated Qualified Number List objects on the same page. Therefore, when we retrieve a Qualified Number List from disk, we get automatically get the Qualified Number we're going to need at the same time. By repeating this process throughout the application, the developer greatly reduces the number of disk accesses.

Once the system fetches a page from disk, it caches it in memory. The

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171

location of the cache can greatly affect performance. Some object databases and most relational databases rely on a server cache. When the server retrieves a disk page from the disk, it copies the page into the cache of the server process and delivers the data to the application process an object or record at a time. The server and the application run as separate processes and exchange information using IPC calls. The database must go through the following steps to fulfill an application request for data:

1. Fetch page from disk into server process cache
2. Copy data into application memory
3. Application accesses the data, possibly modifying it
4. Copy data back to server in the case of data modification
5. Copy data back to disk

The database must repeat all 5 steps each time the application modifies the data and commits the transaction, since the server writes the page back to disk at that point..

Objectivity/DB improves on this performance by having the client fetch the page directly from disk into its cache, eliminating IPC calls and the intermediate step of copying the data to the server cache. Furthermore, since the objects on the page stored in application memory are clustered, the application will have related objects already in memory for the next transaction, further decreasing disk access.

Conclusion: AIN applications and Objectivity/DB are an excellent match, because AIN applications require continuous database availability, complex data models and peak performance. Objectivity/DB gives AIN developers the fault tolerant configurations and data replication they need to provide uninterrupted service that will continue to operate despite hardware and network faults.

Not only does Objectivity/DB support the code re-use inherent to object oriented development, but it also supports new service development after deployment through a sophisticated schema evolution mechanism. The flexibility of Objectivity/DB's object oriented data modelling gives developers a number of implementation options that allow service providers to deploy

Using Higher Performance Databases In Advanced Intelligent Network Applications

new services quickly to take advantage of rapidly changing market opportunities.

www.objectivity.com

Corporate Headquarters:
3099 N. First Street, Suite 200
San Jose, CA 95134 USA
Tel: (408) 992-7100
Fax: (408) 992-7171