A. CANDIDATE OVERVIEWS

A.1 The ODMG Standard

Object Databases involve more than relational databases because the schema defines not only the data structure and types of the objects in the database, but also the methods associated with the objects. The ODMG standard specifies requirements for the following areas of the ODBMS.

- **The database engine**: The DB engine provides all of the database functions and supports most of the object database model. The engine supports indexes and data placement, security, concurrency, recovery, client/server, and distribution.
- **The Object Definition Language**: The ODL replaces the Data Definition Language of a traditional RDB. The ODL is a DDL for object types, and defines the characteristics of types, including their properties and operations. The primary objective of the ODL is to facilitate portability of database schemas across conforming ODBMSs.
- **The Object Query Language**: The OQL allows access to the objects in the database, extracts information from the database and invokes methods on objects.
- **Programming language**: The PL is the language external to the database which is used to write methods associated with objects and to write applications that use and manage these objects. For these languages a *binding* specifies how the language can be used to write methods and applications on a database schema.

Choosing a ODBMS which is "largely" compliant with these standards should make our life easier and would make it possible to change products in the future. Some of the features contained in the ODMG-93 standard have evolved in the period since the draft and thus, some of the non-compliance among vendors is necessary to produce a product more consistent with current technologies.

B. COMPARISON BETWEEN OBJECTIVITY/DB, OBJECTSTORE, O_2 AND UNISQL

B.1.1 Overall database system tools

- **O_2** - The software environment includes O_2 - Look, a GUI builder for objects, O_2Graph, a graph manipulation package, and O_2Tools, a graphical development environment. It also provides many other tools including a report writer, upper and lower CASE tools and Web integration tools. It includes language bindings for C++ and Smalltalk. It provides a completely ODMG compliant OQL, but no ODL.
- **Objectivity** - The suit of software tools include Administrative tools, C++ and Smalltalk bindings, SQL++, ODBC driver and Fault Tolerant Operation and Data Replication options. It also has a completely ODMG compliant ODL and partially compliant OQL (which is SQL++).
- **ObjectStore** - ObjectStore has a collection of about 30 different utilities for copying, exporting, listing, and otherwise manipulating databases. Some of the more notable are osverify, osbrowser, performance expert. It has a C++ language binding. It provides neither an ODL nor OQL.
- **UniSQL -**Tools available Interfaces to the UniSQL server include ODMG compliant C++ and ODMG-based Smalltalk, C API, Microsoft's Visual Basic, and Visual C++, ParcPlace Systems Visual Works Smalltalk, Interactive and Embedded SQL\X, and Protosoft's Paradigm Plus ob

B.1.2 Database engine architecture

All of the architectures consist basically of the following major components:1. The schema manager, 2. the Object manager, and 3. the storage manager. Each product has its own way of combining and implementing these basic features.

- **O_2 -** The client/server architecture of the O_2 engine is based on a page server, as the server deals only with pages and does not understand the semantics of objects. The main advantage of a page server architecture is that it places most of the complexity of the system on the workstation, where the majority of the available CPU cycles are concentrated, leaving the server to perform the tasks that it only can perform. Since entire pages are transferred between the workstation and the server, the overhead on the server is minimized. Because the server does not know about objects, the O_2 engine executes queries and methods on the client.

  The client/server split is performed at the O_2 store level, which deals with pages. The split is made in the following way:

  - **Server process**: 1. A layer provides the storage and retrieval of disk pages plus disk resource allocation, 2. Page-cache, 3. Concurrency control and recovery services. The server is accessed via RPC (socket) interface and the process is multithreaded, allowing physical IO to be done in parallel with client requests.

    - **Workstation process**: 1. Page-cache, 2. All the higher levels. In addition, a local lock manager caches locks acquired by the workstation.

- **Objectivity** - Objectivity/DB claims to have a completely distributed architecture permitting distribution of database services among machines, including object caching, updates, transversals, conversion operations for interoperability. IO services are located on the machines where the data resides. Only concurrency control is centralized. Objectivity provides three types of server processes.

    - **Lock servers**: Because a separate server is used to manage lock requests, IO requests can be processed independently of lock requests. Even in single machine environments, where the machine host manages both locking and application IO requests, the machine can process lock requests concurrently with IO operations, and multiple disk IO requests can be made in parallel.
    - **Local data servers** : Local caching greatly improves performance by reducing the number of times objects need to be read.
    - **Local and remote database servers**: Page servers perform page read and write operations with one or more pages containing a cluster of objects. Support for dedicated I/O servers, on machines which are specialized for I/O operations, are also allowed.

    The Servers can be put together in single or multi-tier schemes which (they claim) deliver consistent performance for database applications as the number of users increases and/or the number of databases grow. The distributed architecture eliminates processing bottlenecks that affect databases which are centralized. The structure can be as simple as a single tier consisting of both the server and application, or as complex as a distributed mixed tier design consisting of multiple remote servers, some with local servers and a central Lock Server to manage concurrency .

- **ObjectStore** - ObjectStore applications require two auxiliary processes for their execution, a Server and a Cache Manager.

    - **Server process**: A server handles access to the ObjectStore databases, including storage and retrieval of persistent data. A single application can use several databases, including those on different file systems, handled by different servers.
    - **Cache Manager**: A Cache Manager is started automatically when an ObjectStore application starts. It runs on the machine running the application, and participates in the management of this application's client cache. If additional applications are started on the same machine, the same Cache Manager handles the caches of these applications as well.

- **UniSQL** - A client-server object-relational database system which provides "reliable" support for "large" and complex data types and transparent integration of data from a variety of multi-vendor databases. The UniSQL server consists of two parts:

- **UniSQL/X DBMS**: UniSQL Server's unified object-relational database engine which provides data management for multi-user applications. It runs under a single operating system process, and uses multithreading to achieve concurrency.
- **UniSQL/M Multidatabase**: provides transparent access to existing relational, object-oriented, and pre-relational databases and file systems.

B.2 Object Reference Architecture

- **O_2 -** An O_2Store system is a set of volumes from which files are allocated. There is no limit to the number of volumes and a volume may reside anywhere on a network, may span more than one operating system files, and may expand dynamically. A sequential file is a sequence of records and records may vary in length and may be inserted and deleted at arbitrary locations within a file. On disk, records storing objects or values refer to each other by physical identifiers; that is, identifiers that reflect the disk locations. The O_2Store record identifiers are used directly and are structured as follows: Volume identifier - 2 bytes, Page identifier within the volume - 4 bytes, and Slot number - 2 bytes which indirectly addresses a record within a page.
- **Objectivity** - Objectivity guarantees integrity of objects by using object references instead of direct pointers to access objects. This architectural feature follows ODMG-93 guidelines, and protects Objectivity/DB from serious object corruption that can be caused by an errant application. Any Objectivity/DB transaction can transparently and simultaneously access all data in a database through 64-bit object references. For 64-bit architectures, this allows you to access up to approximately 2^{63} (or 10^{19}) bytes (10 Million Terabytes) of data and up to 2^{58} (or 10^{17}) objects. For 32 bit machines, due to operating system file size constraints, you can access up to 100 terabytes of data and up to 10^{13} objects.
- **ObjectStore** - ObjectStore achieves a combination of transparency and efficiency with a unique memory mapping architecture called Virtual Memory Mapping Architecture (VMMA). All data is stored in an ObjectStore database in its native C++ format. All pointers in the database take the form of regular virtual memory pointers. The value of a pointer in a given segment is the segment's pseudoaddress for the object the pointer refers to. A pseudoaddress is the identifier a segment uses for an object pointed to by that segment. Each segment has an associated persistent table, the persistent relocation map (PRM), that allows determination of an object's on-disk location (i.e. the database name, segment number, and offset), given its pseudoaddress. Each application's client must have a transient relocation table (TRM) which allows determination of the real address an object occupies in the client cache.
- **UniSQL** - They support a "hybrid" data model which allows conventional relational structures as well as object data. An Object Identifier (OID) is used to identify each instance of a class. The physical storage location can be immediately determined from the OID and navigation from one object to its component parts is performed with pointer chasing. Precise structure and details concerning database scalability were not specified.

B.3 Conclusion

Based on the information supplied by O_2, Objectivity, and Object Design some preliminary evaluation thoughts can be compiled.

The O_2 product seems to be of good overall design and the number of tools which the company provides is impressive. It provides a standard client/server database engine architecture and the Object Reference architecture seems to scale up to large databases, though this is not clear. Objectivity provides fewer tools up front, but its engine architecture seems to be the most flexible of the three considered. One concern is that the use of a separate lock server process might represent a bottleneck, though this needs to be researched further. The Object addressing architecture seems to be sound and resilient to object loss due to database corruption. The use, by ObjectStore, of an independent cache server on each node might provide additional throughput if multiple applications are accessing the same data however, considering the size of most of our databases, this may not be likely. The ObjectStore object pointer architecture is highly tuned for C++, and seems more vulnerable to corruption than other architectures due to the need for relocation maps although, it no doubt is a very efficient technique for accessing in-cache data. Since most of our data will be persistent, this may not be as important.

UniSQL is somewhat of a mystery in that it professes to bring together the best of both the OODBMS and RDBMS worlds. They admit that they are less mature than some of their competitors and have no long transaction support, versioning, or "full" clustering support. It runs under a single multithreaded process, which prevents it from scaling to large number of simultaneous users.

B.4 Bibliography

1. O_2 Database System Technical Overview, May 1996.
2. Objectivity Technical Overview, Version 4, April 9, 1996.
3. ObjectivityStore C++ API User Guide, Release 4.0.1, May 1996.
4. " Candidate ObjectStore Delivers”, Advanced Systems , April 1995.
5. The Object Database Standard: ODMG-93, Release 1.2 , 1996.
6. "UniSQL's Object-Relational Data Management Technology", The Bowen Group Inc. product profile, April 1995.
7. "Next Generation Object Database Systems", Excerpted from Patricia Seybold Group Reports, March 17, 1995.

C. CANDIDATE COST ESTIMATES

This is a summary on the prices for a standard configuration of an Object Oriented Database. The configuration is 5 servers with 100 concurrent users, running on both UNIX and NT servers. It was assumed that there were 3 developers on the project. Prices for WWW products were included where applicable and discounts if they where offered.

UniSQL

Unix			
Description	**Count**	**per unit**	**Total**
UniSQL/X	100	910	########
C++ Interface	100	402.5	########
Total List Cost			########
Volume Discount		40%	########
Net Software Cost			########
Annual Maintenance			########

NT			
Description	**Count**	**per unit**	**Total**
UniSQL/X	100	728	########
C++ Interface	100	322	########
Total List Cost			########
Volume Discount		40%	########
Net Software Cost			########
Annual Maintenance			########

Note: UniSQL does not have a web product at this time. They do have one in beta testing right now. This is for only one server. Prices that are better in line with our needs.

Objectivity

Unix			
Description	**Count**	**per unit**	**Total**
Objectivity/DB for SGI	4	3600	########
Objectivity/DB for SGI/FTO Option	1	4680	$ 4,680.00
Objectivity/DB C++ Developers Inte	3	3000	$ 9,000.00
Objectivity/DB C++ DDL for SGI	3	4000	########
Objectivity/DB User Access for SGI	100	1320	########
Total List Cost			########
Volume Discount		40%	########
Net Software Cost			########
Annual Maintenance			########

NT			
Description	**Count**	**per unit**	**Total**
Objectivity/DB for NT	4	3000	########
Objectivity/DB for NT/FTO Option	1	3900	$ 3,900.00
Objectivity/DB C++ Developers Inte	3	3000	$ 9,000.00
Objectivity/DB C++ DDL for NT	3	4000	########
Objectivity/DB User Access for NT	100	1320	########
Total List Cost			########
Volume Discount		40%	########
Net Software Cost			########
Annual Maintenance			########

Notes: Objectivity has a Fault Tolerant Option (FTO) for their server. We might want one of these for the on-line system.

O_2

Description	**Count**	**Cost**	**Total**
Server		N/A	
O2 Database Tool Packa	1	#######	$12,000.00
Developer Connections	3	#######	$12,000.00
Runtime Connections	97	#######	#########
O2Web	1	#######	$ 5,000.00
Total List Cost			#########
Volume Discount		0%	$ -
Net Software Cost			#########
Annual Maintenance		15%	$33,450.00

Note: The NT & UNIX prices are the same. Once a developer license is purchased you can have as many servers as you need.

Object Store

UNIX			
Description	**Users**	**Cost**	**Total**
Development Server	5	#######	########
Development Client	5	#######	########
Developer Connections	3	#######	########
Objectstore Performance Ex	2	#######	########
ObjectStore Inspector 5 pack	1	#######	$ 5,825.00
Runtime Client	100	#######	########
Web Product Runtime	1	#######	########
Web Product Developement	1	#######	########
Total List Cost			########
Volume Discount		0%	$ -
Net Software Cost			########
Annual Maintenance			########

Description	Users	Cost	Total
Development Server	5	#######	########
Development Client	5	#######	########
Developer Connections	3	#######	########
Objectstore Performance Ex	2	#######	########
ObjectStore Inspector 5 pack	1	#######	$ 4,475.00
Runtime Clients	100	$ 750.00	########
Web Product Runtime	1	#######	########
Web Product Developement	2	#######	########
Total List Cost			########
Volume Discount: 50%		50%	########
Net Software Cost			########
Annual Maintenance			########

Notes: The web tools are NT for both UNIX and NT. The server is an Internet server which will serve anyone that needs it.