



Ross School of Business at the University of Michigan

**Independent Study Project Report**

**TERM** : Fall 1996

**COURSE** : CIS 750

**PROFESSOR** : David Blair

**STUDENT** : Suresh Jayaraman

**TITLE** : Object Relational Database Management Systems and Applications  
in Document Retrieval



*THE UNIVERSITY OF MICHIGAN*  
*SCHOOL OF BUSINESS*

**Object Relational Data Base Management Systems  
and Applications in Document Retrieval**

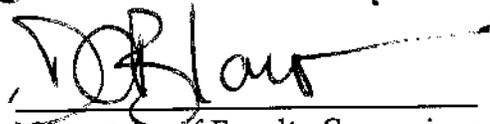
**by**

**Suresh Jayaraman**

A research paper submitted in fulfillment of the requirements for three credits,  
GRADUATE INDEPENDENT RESEARCH PROJECT Fall Term 1996  
Professor David Blair, Faculty Supervisor.

## Faculty Comments

This is an excellent discussion of Object-oriented Data Base Technology and its influence on Relational Data Base Technology. Of particular note is the author's discussion of how well ODBMS's handle complex data type — a problem of growing importance. He also makes some important point on how DBMS technology can handle the growing market of text management.



Signature of Faculty Supervisor

Assoc. Prof. CIS.

Title

Grade = Ex

# Object Relational Data Base Management Systems and Applications in Document Retrieval

## Table of Contents

1. Executive Summary.....	4
2. Evolution of DBMS architectures.....	5
2.1. Traditional File Management Systems.....	5
2.2. Classical Data Base Management Systems.....	7
3. Object-Oriented Technology.....	10
4. The Evolution of ODBMS.....	12
4.1. The ODBMS Market.....	13
4.2. Limitations of ODBMS.....	14
4.3. Approaches to ORDBMS.....	15
4.3.1. Object-to-Relational Mapping.....	15
4.3.2. Hybrid Databases (Universal Servers).....	16
5. A Framework for Classifying DBMS.....	17
6. The Business Case for ORDBMS.....	24
7. Business Benefits of ORDBMS.....	26
8. ORDBMS for Text and Document Retrieval.....	27
8.1. Text Retrieval with Informix ORDBMS.....	27
8.2. Sample Text Searches.....	29
9. Picking the Right Universal Server.....	32
10. Conclusions.....	33
11. References.....	34

## 1. Executive Summary

So elegant is the relational database management system (RDBMS) model that it has survived for 20 years, focusing on simple types of data: integers, scientific floating point, character strings, date/time and money. However, the business world is not nearly as simplistic as it once was. The World Wide Web is driving consumer demand for richer style of interaction. Gone are the days of command-line interfaces. Users want color, sound, animation, intuitive navigation. This has pushed the RDBMS model beyond its 20-year-old design capabilities. The need for data management solutions accessing complex data - in data warehouses, web pages and documents or competitive-advantage applications - is exploding. And if RDBMSs have one drawback, it is that they do not handle highly complex information well.

During the past decade, object-oriented technology has found its way into programming languages, user interfaces, databases, operating systems, expert systems, etc. Products labeled as object-oriented database systems have been on the market for several years and claim to handle complex information much better than traditional RDBMS. Due to vast amount of investment in RDBMS technology and the difficulty in adopting the object paradigm, object databases have lagged relational databases in acceptance considerably. The worldwide market for object databases is 100 times smaller than that for relational databases.

) The RDBMS vendors cannot, however, ignore the tremendous demand for managing complex information. The challenge facing RDBMSs is a key feature known as extensibility. The extension of the existing relational products or models is referred to as the Object-Relational (OR) or extended-relational (ER) DBMS. Looking to guard their flanks, they are incorporating pieces of object technology into their own engines in the form of Universal Servers. The Universal Server will be one of the most significant advances in the RDBMS technology over the next decade<sup>7</sup>.

This study explores this revolution in the database management software industry and the related benefits to Information Systems (IS) . It also discusses various business applications that will benefit from the new technology, including document/text management. This report should help an IS manager understand and evaluate the ORDBMS technology and products and their suitability for IS solutions.

## 2. Evolution of DBMS architectures<sup>18</sup>

A database system is essentially nothing more than a computerized record-keeping system. The database itself can be regarded as a kind of electronic filing cabinet; in other words, it is a repository for a collection of computerized data files allowing the users to retrieve and maintain information in various forms.

DBMS have evolved from a simple file based approach to object oriented approach in the last three decades. A key notion of DBMS is to provide data structures that model real world relationships. The data structures that provide the relationships fall into the following models:

1. Traditional File Management Systems
2. Classical Data Base Management Systems
  - Hierarchical
  - Network (CODASYL)
  - Relational
3. Object-Oriented
4. Extended/Object-Relational (in the future)

### 2.1. Traditional File Management Systems<sup>9</sup>

This is the simplest structure for a DBMS system. This system uses a sequential or a random access file and allows the reading or writing of one record at a time. Random (direct or indexed) access increases performance over sequential processing by providing access to only those records that are required to accomplish a specific objective.

To illustrate how these files might work, Figure 1 shows an employee file implemented under sequential, direct access, and indexed file structures. Note the three-digit employee id (emp\_id) field in each example.

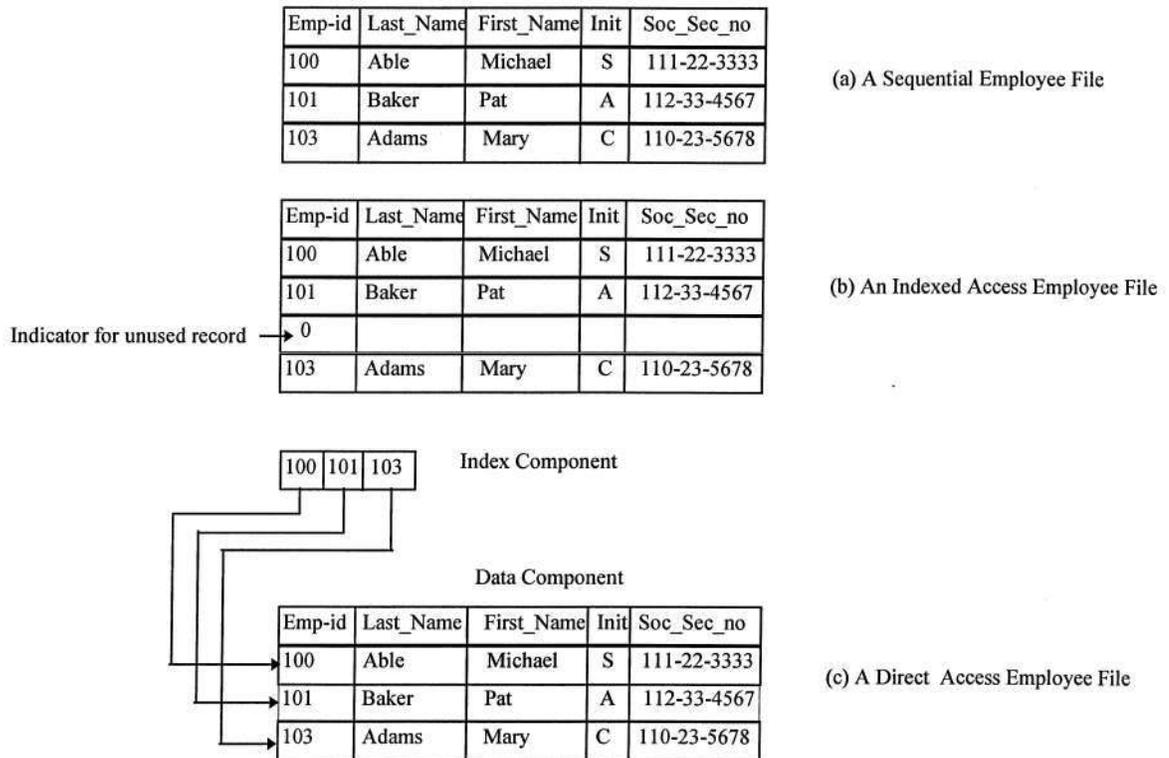
In studying the example, note also the effect of not having an employee with an id of "102". In Figure 1(a), the file is sorted by emp\_id, and the employee record for 102 is simply missing. Figure 1(b), however, is based on a direct access scheme that equates the emp\_id to the record number to be accessed. Therefore, record number 102 exists in the file, but is unused at this time as indicated by having a "zero" value stored in the emp\_id field.

Finally, Figure 1(c) shows these same data stored in an indexed file structure. Note how the system maintains a key for each record in an index component of the file, which is linked to the corresponding data component. Here, as in a sequential file, only records for active employees appear.

Such forms of file structures have several important things in common. First, all records within a given file "look alike"; that is, the same fields exist at the same location or offset within each record. In addition, the same fields exist at the same location or offset within each record.

**Figure 1**

**Employee File Implementations with Simple File Structures**



As a result even minor changes in record layout require changes to every application program that accesses the file. In addition, relationships across files must be embedded within application logic. The advantages and disadvantages of file management systems are summarized in Figure 2.

**Figure 2**

**Traditional File Management Systems**

**Advantages**

- Easy to create and simple to use
- Require minimal overhead to access and use

**Disadvantages**

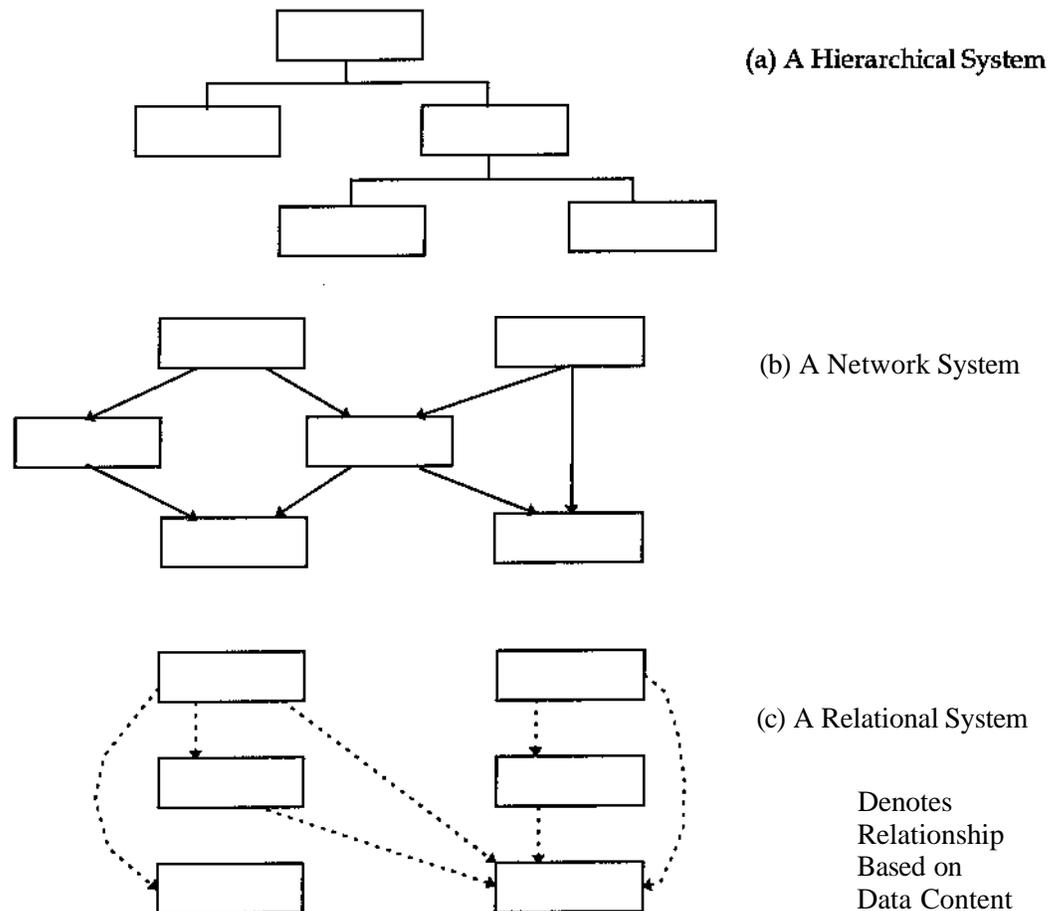
- Changes in structure or content require simultaneous changes to all application programs
- Relationships across files must be embedded within application logic
- Encourage the proliferation of redundant data

In addition, the file descriptions are explicitly declared in each application program. In each case, the program must issue I/O commands directly to the appropriate system access method for all read/write requests, using the required format for the command desired.

## 2.2. Classical Data Base Management Systems<sup>9</sup>

The three classical types of data base management systems are hierarchical, network, and relational. As illustrated in Figure 3(a), a *hierarchical system* has the general shape or appearance of an organizational chart. A node on the chart, representing a particular record type, is subordinate to only one record at the next highest level just as, on an organizational chart, an employee reports to only one boss. This kind of structure is often referred to as an "inverted tree", with the top-most record referred to as the "root".

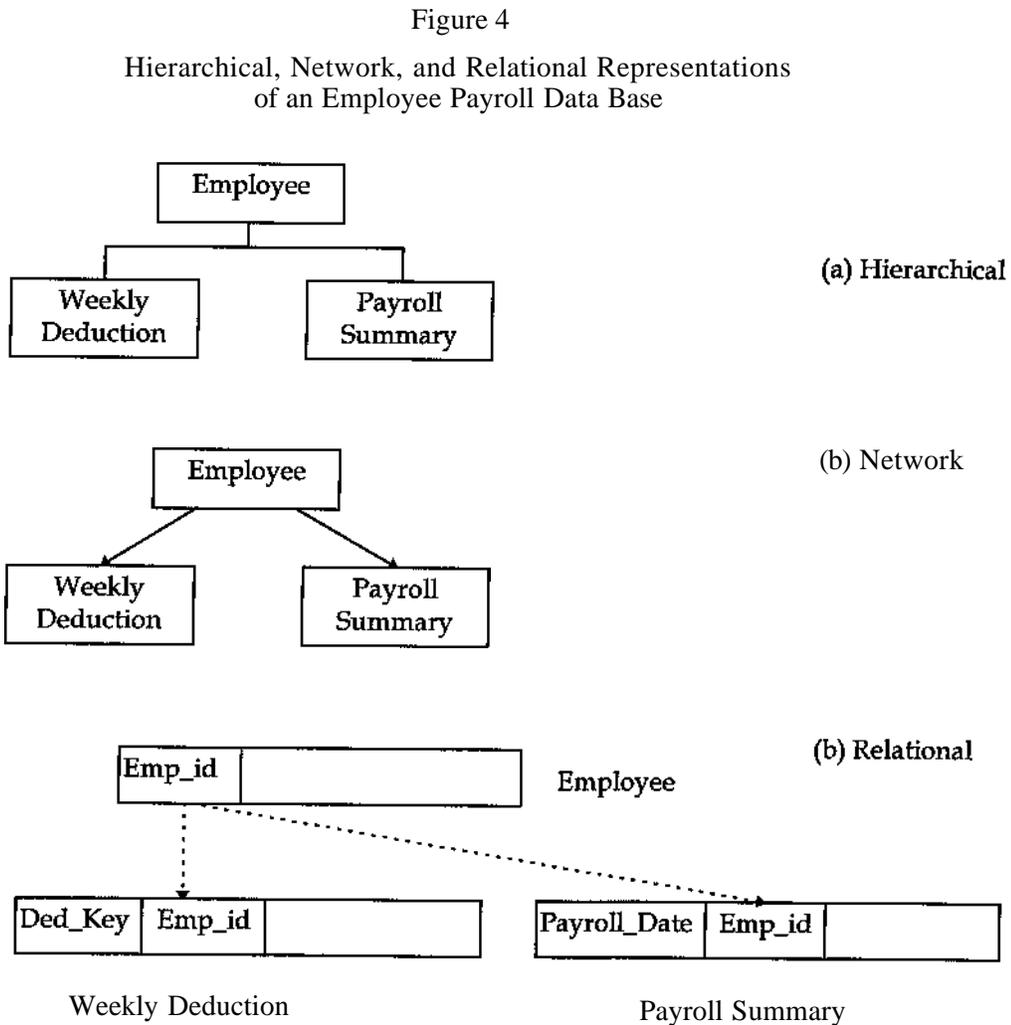
Figure 3  
The Three Types of Classical Data Base Systems



As shown in Figure 3(b), a network structure is somewhat similar, but has one major difference. Subordinate record types, depicted by arrows on the network diagram, may participate in as many subordinate relationships as desired. Therefore, a much more complex diagram may be used to represent the structure of the data base. Networks provide more flexibility than a simple hierarchical system in the data relationship that may be maintained.

Finally, a relational data base consists of a collection of simple files or tables, each of which has no structural or physical connection such as those typically used in hierarchical or network systems, as shown in Figure 3(c). The various records possess the interrelationships as depicted by a network-like diagram, but these relationships are based on the data content of the records involved, not by pointer chains or other types of structural connection techniques.

Figure 4 illustrates how a data base containing employee, weekly deduction, and payroll summary records might look under each of these types of data base management systems.



In each case, ignoring internal limits of data set size for the sake of discussion, there can be as many occurrences of any record type as needed. In each case, as records are or added or deleted, the DBMS maintains any relationships between and across these records.

The relative advantages and disadvantages of each system are shown in Table 1.

Table 1  
DBMS Tradeoff Matrix<sup>1</sup>

<i>[DBMS</i>	<i>Example</i>	<i>Advantages</i>	<i>Disadvantages</i>
Hierarchical	IBM's IMS	<ul style="list-style-type: none"> <li>• Well established technology</li> <li>• Fast response times</li> </ul>	<ul style="list-style-type: none"> <li>• Not all data structures are hierarchical</li> <li>• Cannot model "many-to-many" relations</li> <li>• Hard to change database design</li> </ul>
Network	Computer Associates' IDMS	<ul style="list-style-type: none"> <li>• Can model more complex structures than hierarchical (ex. "many-to-many" relations)</li> <li>• Can answer more complex queries</li> </ul>	<ul style="list-style-type: none"> <li>• Access programs can be very complex</li> <li>• All query types must be known before design</li> </ul>
Relational	DB2 Informix Oracle Sybase	<ul style="list-style-type: none"> <li>• Relational theory provides more formal basis for design</li> <li>• No practical limit for data structure complexity</li> <li>• Data structures are dynamic rather than static</li> <li>• Supports <i>ad hoc</i> queries (Implicit relations)</li> <li>• Build on known user ability (to read tables)</li> <li>• Design choice for new Information Systems</li> <li>• New versions take advantage of parallel hardware architectures</li> </ul>	<ul style="list-style-type: none"> <li>• Generally slower than hierarchical or network systems for predictable data structures</li> <li>• Not suitable for managing complex data (text, video, time-series etc.)</li> </ul>

### 3. Object-Oriented Technology<sup>3,6</sup>

Relational databases store data in tables that have to be joined together to answer complex queries. Object oriented database management systems (ODBMS) link complicated data structures as more easily accessible objects. Object-based software also can handle multimedia data such as video and audio. Relational databases typically reduce everything to numbers and characters. To understand the object-oriented approach let us look at a few key concepts.

#### Objects

*Objects combine procedures and data*

The basic unit organization in the object-oriented approach is the object. An *object* is a software "packet" containing a collection of related data elements and a set of procedures, called *methods*, for operating on these elements. The data within an object can be accessed only by the object's methods, which handle all the routine tasks of reporting current values, storing new values, or performing calculations. This arrangement, called *encapsulation*, protects data from corruption by other objects and hides low-level implementation details from the rest of the system.

*Objects provide high level data structures*

Once defined, new types of objects can be used as basic data types within a program in the same way as the built-in data types that handle numbers, dates, and other elementary kinds of information. This ability to create new, high-level data structures on demand and use them in subsequent programming is called *data abstraction*. Data abstraction is central to the object-oriented approach because it allows programmers to think in terms of the problems they are solving rather than the data types of the language.

#### Messages

*Objects communicate through messages*

Objects communicate with one another through messages. A *message* is simply the name of a receiving object together with the name of one of its methods. A message is a request for the receiving object to carry out the indicated methods and return the result of that action.

*The same message can be handled differently by different objects* Any number of objects can include the same method, and each can implement it according to its own unique needs. That allows any given message to be sent to many different objects without worrying about how the message will be handled or even knowing what kind of object will receive it. The ability to hide implementation details behind a common message interface is known as polymorphism. Polymorphism makes the object-oriented approach very flexible because it allows new kinds of objects to be added to a completed system without rewriting existing procedures.

## Classes

*Classes describe the properties of objects*

Object-oriented programming supports the repeated use of common object types through the use of classes. A class is a general prototype which describes the characteristics of similar objects. The objects belonging to a particular class are said to be instances of that class.

*This is an efficient arrangement*

Classes allow objects to be defined in a very efficient manner. The methods and variables for a class are defined only once, in the class definition, without repeating them in every instance of that class. The instances contain only the actual values of the variables.

*Classes can inherit characteristics*

Although it is possible to define classes independently of each other, classes are usually defined as special cases, or subclasses, of each other. Through a process called inheritance, all the subclasses for a given class can make use of the methods and variables of that class. Inheritance increases the efficiency of the class mechanism even further: behavior that's characteristic of larger groups of objects is programmed only once, in the definition of the higher-level class, and the subclasses merely add to or modify that behavior as required for their special cases.

*Class hierarchies allow classes to be defined efficiently*

Subclasses may be nested to any degree, and inheritance will accumulate down through all the levels. The resulting treelike structure is known as class hierarchy. Some languages allow a class to inherit properties from more than one superclass, a feature known as multiple inheritance. This feature complicates matters by creating multiple overlapping hierarchies, but it permits much more flexible relationships.

## 4. The Evolution of ODBMS <sup>2</sup>

Object databases began arriving in the late 1980s to compete against relational databases. Object databases could store complex data types. In 1989, a group of representatives from ODBMS vendors produced "The Object-Oriented Database System Manifesto"<sup>11</sup>. This paper attempted to define exactly what constituted (or should constitute) an object database. The paper was quickly followed by a response from a different set of database thinkers, led by Michael Stonebraker, entitled "The Third-Generation Database Manifesto"<sup>12</sup>. This paper formed much of the intellectual basis for the ERDBMS/ORDBMS efforts. The manifestos capture academic positions that were taken in the mid-to-late 1980s by researchers seeking to progress from the relational model.

Table 2

**Comparison of Object and Object/Extended-Relational Manifestos**

<b>OO Database System Manifesto</b>	<b>1 ORDBMS/ERDBMS Manifesto</b>	<b>1</b>
Support for complex objects	Support for complex objects (array, sequence, record, set, union as types)	
Unique object identity	Object identity only if there is no unique key	
Support for encapsulation	Support for encapsulation	
Support for class structure	Support for class structure	
Support for inheritance	Support for inheritance	
Support for polymorphism	No specific reference	
Computational completeness	Support for SQL	
Support for extensibility	Support for extensibility	

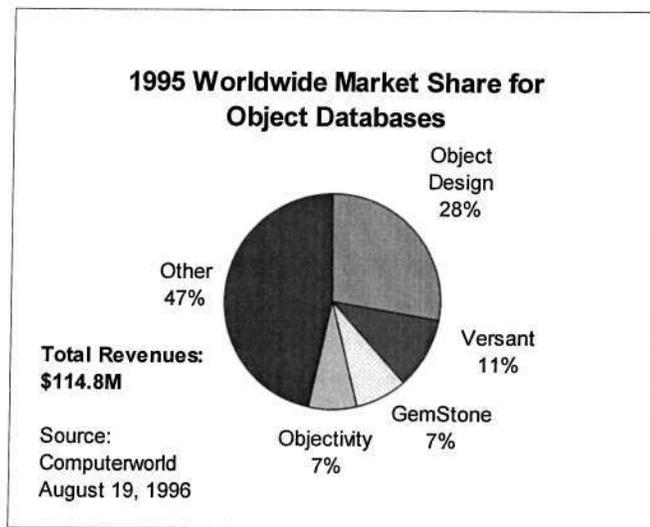
In their time, the two manifestos (and other papers) attracted attention and framed the debate about what a database should be fundamentally about. Table 2 compares some important qualities highlighted by the two manifestos. What is startling is not so much the differences, but the similarities. Support for complex objects, encapsulation, class structure, inheritance, and extensibility are not in dispute. Thus, the basic set of ideas presented by the ODBMS side is not in dispute - only how to achieve them in a database.

Other factors are perhaps more important in separating the two approaches. Most object databases have become persistence mechanisms for applications built with object-oriented languages. They have close ties to Smalltalk, C++, and now Sim Microsystems' Java. The ODBMS products support access to the objects and moving from one object to another by traversing pointers or other references. Therefore, object databases are good at storing part/subpart hierarchies. Objects will typically be stored inside container classes (such as sets, lists, and bags) that can, of course, be nested.

#### 4.1. The ODBMS Market <sup>6</sup>

The ODBMS market consists of a half a dozen players. The general perception is that ODBMS is suitable for niche applications and mainstream acceptance is far from reality. Figure 5 shows the 1995 worldwide market share for ODBMS.

Figure 5



Customer such as Sprint Corp., Lehmen Brothers and Time, Inc. are using ODBMS technology for specialized applications. Sprint Corp. is testing the object waters by off-loading a small piece of its commercial long-distance customer service application from a mainframe database to Versanti software. Wanting to off-load queries on stock trades from its main database servers, Lehmen Brothers is deploying small object databases on individual PCs as local caches for trading records. Time Inc/s New Media unit uses Object Design's ObjectStore database for personalized news service. Major ODBMS vendors and their target markets are shown in Table 3.

Table 3  
 ODBMS vendors and Target Markets

<i>Vendor</i>	<i>Products</i>	<i>Targeted Markets</i>
Object Design, Inc.	ObjectStore	Internet, computer-aided design, telecom, financial services
Versant Object Technology Corp.	Versant ODBMS	Telecom, health care, Internet, financial service
Gemstone Systems, Inc.	GemStone	Financial Services, utilities, insurance, manufacturing
Objectivity, Inc.	Objectivity/DB	Software developers, Internet
Ontos, Inc.	Ontos DB/Explorer	Low-end uses in manufacturing, aerospace, financial services

#### 4.2. Limitations of ODBMS <sup>13</sup>

Current ODBMS products lack the features that users of database systems have become accustomed to, and therefore, have come to expect. The missing features include a full non-procedural query language, automatic query optimization and processing, automatic concurrency control, authorization, dynamic schema changes, and parameterized performance tuning.

- **Most of the OODBs suffer from the lack of query facilities.** Nested subqueries, set queries (union, intersection, difference), aggregation functions and group by, and even joins of multiple classes, etc. These facilities are all fully supported by RDBMSs. Though these products allow users to create a flexible database schema and populate the database with many instances, they provide neither a powerful enough means of retrieving objects from the database nor a means to share objects with other users in a controlled way.
- **Some of the current OODBs require the user to explicitly set and release locks.** Relational Data Bases automatically set and release locks in processing query and update statements issued by users.
- **Most OODBs do not support authorization.** RDBMSs support authorization; that is, they allow users to either grant and revoke privileges to read or change the data they created to other users or change the definition of the relations they created to other users.
- ® **Most of the current OODBs allow new classes to be added to the database, but they do not allow any additional changes to the database schema, such as adding a new attribute or method to a class, adding or dropping a new superclass to a class, or dropping a class. Using the ALTER command,**

RDBMSs allow the user to dynamically change the database schema to add or drop a column in a relation or to drop a relation.

- **Most of the OODBs offer a limited capability for parameterized performance tuning.** Relational Data Bases allow the installation to tune system performance by providing a large number of parameters that can be set by the system administrator. The parameters include the number of memory buffers, the amount of free space reserved per data page for future insertions of data, and so forth.

Due to the limitations mentioned above, most of the relational and object-oriented database system products will require major enhancements. Vendors such as Informix (Universal Server) and Oracle (Universal Server, Oracle 8) that reengineer and rewrite the products from scratch will be in a leadership position.

### 4.3. Approaches to ORDBMS

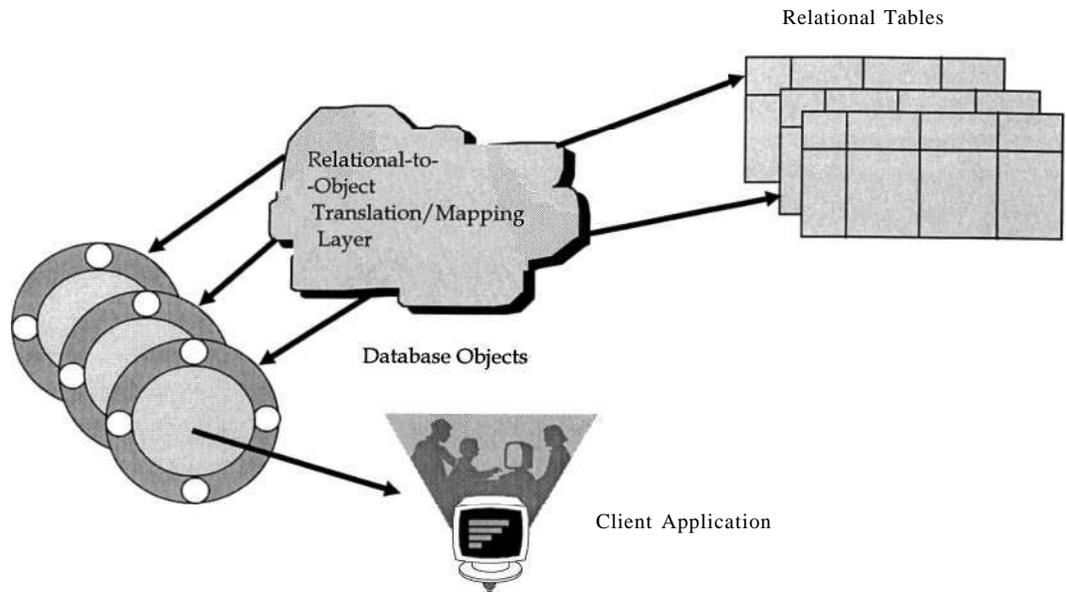
Because relational databases are primarily content-addressable, you don't need to know how to find a record in a database, only how to describe it. However, using a traditional relational database, you will, for example, have difficulty selecting a set of photographs depicting a sunset taken in a given town. A relational database has no easy means of accessing a photograph's content even though it can record the name of the town. The O/ERDBMS products are designed to bring relational's content-addressability to complex objects—as well as, usually, some form of pointer-chasing. A number of ODBMS products allow some form of content-addressability: The key difference with an O/ERDBMS is that it expresses content-addressability as an extension to SQL.

#### 4.3.1. Object-to-Relational Mapping<sup>10</sup>

Object-to-Relational mapping is an approach pursued by vendors such as Persistence Software, Enterprise Object Framework from NeXT computer, and Dbtools.h++ from Rogue Wave Software. Object-to-relational mapping mechanisms let you link the properties of a relational database. (See Figure 6). This displays the relational database as persistent objects inside the development environment. Usually this means making a connection between a data element that exists in an object and a data element that exists in a relational database.

Usually this means making a connection between a data element that exists in an object and a data element that exists in a relational database.

**Figure 6**  
**Object-to-Relational Mapping Mechanism**



You can map a single object, directly to a single table, several tables to an object, or link several objects to a single table. Once the mapping process is complete, any data modified in a mapped object will automatically modify any linked tabs or tables.

At the heart of this mechanism is the relational wrapper, the layer that sits between the objects and the database. The relational wrapper detects a change in the contents of an object and automatically generates SQL to make the changes in the linked relational database. At the same time it also detects changes to the relational database and moves that information back to both the developer and the end users.

Additional layers in the object-to-relational wrapper approach has performance impact, especially in database intensive On Line Transaction Processing (OLTP) applications. This makes, for example, many Smalltalk tools that use relational databases less desirable in environments for which performance is a critical success factor.

### **43.2. Hybrid Databases (Universal Servers)**

The big relational database vendors such as Informix, Oracle, and Sybase have caught on to the object trend. There is a movement afoot among these vendors to build OO database capabilities into traditional relational databases. Such databases are known as hybrid databases or Universal

Servers. Hybrid databases are sold by a few small companies such as Illustra (now owned by Informix), OmniScience, and UniSQL. These databases store information using OO or relational models, and they let developers access the data using either method.

Hybrid databases can appear as both relational and OO databases, depending on the needs of the application. For example, OO databases are naturally better at storing complex data structures and binary information such as images, audio files, Web content, or even videos. To store the same type of content within a relational database, one may have to rely on the database's ability to handle binary large objects (BLOBs). In the relational world, BLOBs are opaque, meaning that information inside the BLOB is invisible to the database engine. OO databases solve this problem.

ORDBMSs such as Informix's Illustra attempts to offer the best of both worlds. It supports OO management of rich (complex) data types, but at the same time provides an efficient query language based on extensions to industry-standard SQL. Its support for inheritance speeds application development. Object extensions, called DataBlade modules, plug intelligence into Illustra for specific kinds of data, extending the SQL language with tailor-made functions and allowing Illustra to manage the data required by specific applications. The DataBlade modules can even include new access methods such as D-Tree for text retrieval.

## **5. A Framework for Classifying DBMS<sup>5</sup>**

The previous sections discussed the evolution of data base management systems. They do not, however, put the DBMS from both technical and marketplace perspectives. Stonebraker<sup>5</sup> introduced a framework to help classify business applications that require DBMS. He presents a classification of the applications that require DBMS technology to show where Relational DBMS, Object-oriented DBMS and Object-Relational DBMS fit. The purpose is to indicate the kinds of problems each kind of DBMS solves. He suggests that "one size does not fit all"; i.e. there is no DBMS that solves all the applications we encounter.

The classification scheme makes use of the two by two matrix indicated in Figure 7. It shows a horizontal axis with very simple data to the left and very complex data to the right. In general, the complexity of the data that an application must contend with can vary between these two extremes in a continuous fashion.

**Figure 7**  
**A Classification of DBMS Applications**

Query		
No Query		
	Simple Data	Complex Data

However, for simplicity, it assumes there are only two discrete possibilities, namely simple data and complex data. Similarly, on the vertical axis differentiates whether the application requires a query capability. This can vary between "never" and "always". Again, there is a continuum between the two extremes; however, for simplicity it assumes there are only two discrete possibilities. These are respectively "query" and "no query".

A user can examine an application and then place it in one of the four boxes in Figure 7 depending on its characteristics. In order to illustrate this classification process, an application in each of these four boxes is explored. In the process of discussing each application, the requirements that each has of a DBMS is considered and a natural choice of data manager for each of the four applications will be examined.

**Lower Left Corner**

Consider a standard text processing system such as Word for Windows, Framemaker, Word Perfect, vi, or emacs. All allow the user to open a file by name and manipulate its contents. At intervals, the object is saved to disk storage. Finally, when the user is finished he can close the file, thereby causing the virtual memory copy to be stored back to the file system

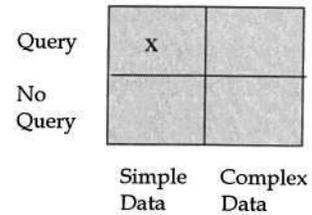
Query		
No Query	X	
	Simple Data	Complex Data

The only query made by a text editor is "get file", and the only update is "put file". As such, this qualifies as a "no query" application, and a text editor has no use for SQL. In addition, a text editor is satisfied with the data model presented by the file system, namely as an arbitrary length sequence of characters. As such, this is a "no query — simple data" application and fits in the lower left hand corner of our two by two matrix.

The bottom line is simple. If you have an application in the lower left hand corner of Figure 7, then you should deploy it on top of the file system provided with your computer.

### Upper Left Hand Corner

We can illustrate the upper left hand corner by treating a well known example. Suppose we want to store information about each employee in a hypothetical company, namely we wish to record the name of the employee, his age, his salary, and his department. In addition, we require information about the departments in this company, namely their department name, their budget and the floor number they are located on.



The schema for this information can be captured by the following standard SQL statements:

```
create table emp (
  name varchar(30),
  age int,
  salary float,
  dept varchar(20));
```

```
create table dept (
  dname varchar(20),
  budget float,
  floor int);
```

The records utilize the standard data types found in SQL-92. As such, we classify this data as "simple". Users wish to ask a collection of queries against this data such as:

Find the names of employees who are under 40 and earn more than \$40,000.

```
select name
from emp
where age < 40 and salary > 40000;
```

Find the names of employees who work on the first floor.

```
select name
from emp
where dept in
(select dname
from dept
where floor = 1);
```

This application can be viewed as one in the "simple data -- query" box. Applications in the upper left hand corner that have this characteristic tend to be identified as "business data processing" applications and have the following requirements:

**Query Language:** Requirement for SQL-89. Desirable to also have the newer SQL-92 standard.

**Client Tools:** Requirement for a tool kit that allows a programmer to set up forms for data entry and display. Tool kit must also allow sequencing between forms through

control flow logic. Such tool kits are called fourth generation languages (4GL). Example 4GL include Powerbuilder from Sybase, Windows/4GL from Computer Associates, or SQL-forms from Oracle.

Performance: Much of the business data processing marketplace entails so-called transaction processing. Here, many simultaneous users submit requests for DBMS services from client terminals or PCs. This has led to the notion of two phase locking which ensures so-called serializability. In addition, there is an absolute requirement to never lose the user's data, regardless of what kind of failure might have occurred. Together, these requirements provide transaction management; i.e. user queries and updates are grouped into units of work called transactions. Each transaction is atomic (i.e. it either happens completely or not at all), serializable (i.e. appears to have happened before or after all other parallel transactions) and durable (once committed, its effect can never be lost).

Security: Since users put sensitive data, such as salaries, into business data processing data bases, there is a stringent requirement for DBMS security.

Business data processing is a large and demanding market. Vendors have their hands full with the following kinds of tasks:

- run well on shared memory multiprocessors
- interface to every transaction monitor under the sun
- run on every hardware platform known to man
- provide a gateway to all the other vendor's DBMS
- provide parallel execution of user queries
- solve the "32 bit barrier", i.e. run well on very large data bases
- provide "7 times 24", i.e. never require taking the database off-line for maintenance

As such, the Relational DBMS vendors have large development staffs busily improving their products in all of these areas.

### Lower Right Hand Corner

Consider a simple application which fits in the lower right hand corner. Suppose the user is the facilities planner for a company that has an "open floor plan", i.e. nobody gets an office. Hence, all employees are arranged into cubicles, with partitions separating them. In such a company, departments grow and shrink and employees get hired and quit. Over time, the arrangement of employees on the physical real estate of the building becomes suboptimal, and a global rearrangement of space is warranted.

Query		
No Query		X
	Simple Data	Complex Data

This "garbage collection" of free space and concurrent rearrangement of employees is our target application. The database for this application can be expressed in the following SQL commands:

## Upper Right Hand Corner

We now turn to an application that is query-oriented and requires complex data. As such, it is representative of the upper right hand corner. The State of California Department of Water resources (DWR) is charged with managing most of the waterways and irrigation canals in California, as well as a collection of aqueducts, including the massive state water project.

Query		X
No Query		
	Simple Data	Complex Data

To document their facilities, they maintain a slide library of 35mm slides. Over time this library has grown to 500,000 slides and is currently accessed between 5 and 10 times per day by DWR employees and others.

The client is typically requesting a picture by content. For example, an employee might have to give a presentation in which he needed a picture of "the big lift", a massive pumping station that lifts Northern California water over the Tehachapi Mountains into Southern California. Another request might be for a picture of San Francisco Bay at sunset, while a third might request a picture of a reservoir whose water level was very low.

DWR has found that it is very difficult to find slides by content. Indexing all the slides according to a predefined collection of concepts is a prohibitively expensive job. Moreover, the concepts in which clients are interested changes over time. For example, low water in a reservoir was never of interest until the current drought started 7 years ago. Similarly, endangered species are currently a heated issue, but only in the last few years. At the current time, they have a written caption about each slide. One example might be:

*Picture of Auburn Dam taken during scaffold construction*

and a fairly primitive system which can identify slides from specified keywords. This keyword system is not operating very well because many concepts of interest are not mentioned in the caption, and hence cannot be retrieved.

As a result, DWR is scanning the entire slide collection into digital form and is in the process of constructing the following database:

```
create table slides (  
    id      int,  
    date   date,  
    caption document,  
    picture photo_CD_image);  
  
create table landmarks (  
    name  varchar(30),  
    location point);
```

Each slide has an identifier, the date it was taken, the caption mentioned above, and the digitized bits in Kodak Photo-CD format. In fact, Photo-CD format is a collection of 5 images ranging from a 128 X 192 "thumbnail" to the full 2K X 3K color image. At the

current time, DWR has digitized about 20,000 images and is well on their way to building a data base whose size will be around 3 terabytes.

DWR is very interested in classifying their images electronically. As noted above, classifying them by hand is implausible. One of the attributes they wish to capture is the geographic location of each slide. Their technique for accomplishing this geo-registration involves a public domain spatial data base from the US Geologic Survey. Specifically, they have the names of all landmarks that appear on any topographic map of California along with the map location of the landmark. This is the table landmarks mentioned above. Then, they propose to examine the caption for each slide and ascertain it contains the name of a landmark. If so, the location of the landmark is a good guess for the geographic position of the slide.

In addition, DWR is interested in writing image understanding programs that will inspect an image and ascertain attributes of the image. In fact, one can find sunset in this particular slide library by looking for orange at the top of the picture. Low water in a reservoir entails looking for a blue object surrounded by a brown ring. Many attributes of a picture in which DWR has an interest can be found using fairly mundane pattern matching techniques. Of course, some attributes are much more difficult, such as ascertaining if the picture contains an endangered species. These harder attributes will have to wait for future advances in pattern recognition.

As a result, the schema mentioned above contains a caption field that is a short document, a picture field that is a Photo-CD image and a location field that is of type geographic point.

Moreover, the clients of this data base will all submit *ad hoc* inquiries. One such inquiry would be to find a sunset picture taken within 20 miles of Sacramento. Clients want a friendly interface that would assist them in stating the following SQL query:

```
select id  
from slides P, landmarks L S  
where sunset (P.picture) and  
contains (P.caption, L.name) and  
L.location |20| S.location and  
S.name = 'Sacramento';
```

This query can be explained by talking through it from bottom to top. First, we need to find Sacramento in the Landmarks table. This yields the geographic location of Sacramento (which is on several topographic maps). Then, we find other landmarks (L.location) which are within 20 miles of S.location. |20| is a user defined operator defined for two operands, each of type point, that returns true if the two points are within 20 miles of each other. This is the set of landmarks which we can use to ascertain if any appear in a caption of a picture. *contains* is a user defined function which accepts two arguments, a document and a keyword, and ascertains if the keyword appears in the document. This yields the set of pictures that are candidates for the result of the query. Lastly, *sunset* is a second user-defined function that examines the bits in an

image to see if they have orange at the top. The net result of the query is the one desired by the client.

Obviously, this application entails "query mostly" on complex data. As such, it is an example of an upper right corner application. Such DBMS that support a dialect of SQL-3, that include non-traditional tools, and which optimize for complex decision support SQL-3 queries, are termed as Object-Relational (OR) or Extended-Relational (ER) DBMS. They are relational in nature because they support SQL; they are object-oriented in nature because they support complex data. In essence they are a marriage of the SQL from the relational world and the modeling primitives from the object world. Vendors of Object-Relational DBMS include Illustra, UNISQL, and Hewlett-Packard (with their Odaptor for Allbase and more recently for Oracle).

## 6. The Business Case for ORDBMS <sup>7</sup>

As explained above, there are three different kinds of DBMS, each with its own focus on a particular segment of marketplace. These segments require very different query languages and tools, and are optimized with different kinds of engine enhancements. Moreover, the requirements for security differ among the segments. In effect, each kind of engine has carefully "scoped out" a segment of the marketplace and then optimized for that segment.

In summary, classify your problem into one of the four quadrants, and then use a DBMS optimized for the quadrant. This advice is summarized in Figure 8.

Figure 8

A Classification of DBMS Applications

Query	Relational DBMS	Object-Relational DBMS
No Query	File System	Object-Oriented DBMS
	Simple Data	Complex Data

Further examination of the above classification reveals relative market share of each of the segments. The relational market is about \$8 billion per year, while the object-oriented database market is a factor of 100 smaller. Stonebraker <sup>5</sup> believes that the growth rate of both markets is substantial, and expects their relative sizes in 10 years to approximately preserve the factor of 100 difference, as illustrated in Figure 9. Figure 9

also shows the expected size of the object-relational market is 50% larger than the relational market by the year 2005. He cites two forces that will cause the object-relational market to dominate, generating the "next great wave":

**Figure 9**  
**Relative Size of DBMS Markets in Year 2005<sup>5</sup>**

Query	Relational DBMS 100	Object-Relational DBMS 150
No Query	File System	Object-Oriented DBMS 1
	Simple Data	Complex Data

**Force 1: Computerization of New Multimedia Applications**

Users are computerizing complex data at an astonishing rate. As described above, the DWR application is scanning data not currently in electronic form. It is estimated that 85% of the world's useful information is not in electronic form. As significant amounts of this data are captured, they will generate a huge market for primarily upper-right quadrant applications. The World Wide Web, which was virtually non-existent three years ago, is one example of an explosive new market that will be query oriented on complex data. A second example where rapid growth is occurring is digital film. Over the next decade conventional film may well disappear as a storage medium for data. This will occur at the high end in medical devices, such as X-ray and ultrasound systems, as well as low end home photography.

**Force 2: Business Data Processing Applications**

A growing need for decision support queries on complex data, hastened by rapidly declining costs in hardware, will cause upper-left quadrant applications to move to the upper-right. A typical application could be for an insurance company that wants to add a diagram of each accident site, the scanned image of the police report, the picture of the dented car etc. All of this new data can be easily handled by an ORDBMS.

## 7. Business Benefits of ORDBMS <sup>7</sup>

Neither the network databases that first addressed the task of data management, nor the relational databases that superseded them were properly able to manage data much more complex than numeric and character data. However there is a wide variety of data that software can usefully manipulate which falls far outside the limited range of character and numeric data.

For example, "What are the 13-week average sales for our top five profitable products ?" Business managers asking this simple question do not know that to implement it, an RDBMS programmer must churn out and test several pages of SQL code to first calculate the profitability of products, then rank them by profitability, and then calculate the 13-week average sales. Nor do they realize that the query must be reworked the following week. The 13-week average changes every week, but because the RDBMS does not understand time series, moving averages or ranking, the programmer must force-feed it with a program embodying these "complex data types/<sup>7</sup>

**Table 4**  
**Business Opportunities for ORDBMS (Universal Servers)**

Area	Opportunity
Financial/Insurance	Derivative calculation, quantitative-model scaling, actuarial tables, currency conversion
Manufacturing	Bill-of-Materials explosion, economic order quantity computation
Healthcare	Treatment of coding hierarchies, image and document management
Data Warehouses	Aggregates, time series, business-model-based data mining
Sales and Marketing	Geographical, spatial, and demographic, (e.g., sales by area) data, customized multimedia demos
Security	Monitoring video cameras for changing patterns
Entertainment	Querying videotape archives, retrieving live-broadcast material for immediate playback, supporting pay-per-view
Pharmaceutical	Molecular modeling and computational chemistry
Communications	Parsing telephone numbers, decoding IP addresses (e.g., for the Internet)

Source: Aberdeen Group, 1996

RDBMS vendors are planning to offer OO features in their future releases of products termed "Universal Servers"<sup>7</sup>. The long-term benefits of Universal Servers will be significant, since they will apply to commercial applications in nearly all industries. For example, exploding bill-of-materials and calculating economic order quantities are difficult tasks today with RDBMSs. But with Universal Servers with OO extensions, they will be relatively straight forward, allowing more effective just-in-time resource planning.

Enterprises will also be able to query their videotape records and onsite-camera video feeds for particular patterns. For example, video camera monitoring an assembly line can feed video data into an ORDBMS that can detect anomalies such as defects and trigger corrective action, thus improving product quality at lower cost.

Table 4 lists examples of business opportunities that could benefit from ORDBMS (soon to be offered as Universal Servers by leading RDBMS vendors). In the next section Text and Document retrieval applications of ORDBMS are examined in depth.

## 8. ORDBMS for Text and Document Retrieval

An area where relational databases have not been able to make much of a contribution to data storage is in the area of text. Text is by its nature contextual and cannot be easily represented in relational tables. It is, in a strange way, multi-dimensional. A given sentence within a paragraph can only be understood within the context of the paragraph, similarly the paragraph falls within the context of the chapter and so forth. However it is often useful to be able to skip from one part of a book to another. It is often useful to be able to do proximity searches, such as retrieving all paragraphs where the words 'database' and 'API' occur together. You could also wish to retrieve all paragraphs where they occurred separately. This is, in effect, a kind of two dimensional search.

Such capabilities have long been the domain of niche text database products and hence there has been difficulty in integrating access to such information with access to other data. Several RDBMS vendors have announced plans for supporting these capabilities. Oracle has announced the Oracle7 Release 7.3 ConText Option which is an add-on to their base RDBMS server. Informix has taken a different approach by including the Verity topic®SEARCH DataBlade® with their Universal Server.

The rest of this section deals with Informix's approach to text searching capabilities.

### 8.1. Text Retrieval with Informix ORDBMS

Traditional text and document retrieval engines are based on proprietary technology. Relational databases do offer clear advantages in *ad hoc* query processing over nonrelational systems and *ad hoc* inquiry is essential to effective information retrieval<sup>14</sup>. The Informix Universal Server combines the best of the

relational approach and natural language based approach (used by traditional document retrieval engines). At the heart of the Informix Universal Server's extensible architecture is their DataBlade technology. DataBlades are object extensions that plug intelligence into their Universal Server for specific kinds of data. DataBlade modules can even include new access methods, for speedy access to data not well served by the B-Trees of the RDBMS vendors. Additional DataBlade modules can be developed by the customer or third-party over time as either core services or as options, thereby providing customers with greater overall functionality and flexibility.

The Informix Universal Server will provide a choice of text searching capabilities based on different DataBlades from Verity, PLS, and Excalibur. These DataBlade modules enable one to perform a variety of Text searches on documents stored in the database. The following example illustrates the mechanics of using a DataBlade for text retrieval.

1. **Install the appropriate DataBlade.** In this example we will choose the Verity Text Search DataBlade which is included with the Informix Universal Server. A DataBlade module comes with its own UDT (User Defined Datatype or Object), its own access methods, and its own functions (methods) that operate on the object or UDT. Installation procedures will be included by the DataBlade provider.
2. **Create the document table using SQL CREATE TABLE.** The following example shows how a two-column table named VTSdocuments.

```
CREATE TABLE VTSdocuments (  
  vname      varchar(25),  
  vdocument  doc )
```

New UDT (user defined data type) previously created using 'create type' statement

3. **Insert the Documents.** The following statements loads an ASCII document, which describes Informix Products:

```
INSERT INTO VTSdocuments values (  
  'DBMS Overview',      'format(ascii):tmp/dbms.txt' );
```

Tells UDT, document type and location to read from

**Create DataBlade Index.** The following statement creates a Verity index on the document column, enabling the use of text searching functions:

Tells ORDBMS engine that this is a special index using Verity Text Search DataBlade

CREATE INDEX VTSdocidx on VTSdocuments *using vts* (vdocument)

5. Perform text search. The text searching functions provided by the DataBlade, can be performed now . Several examples are shown below.

Table: VTSdocuments

vname	vdocument
DBMS Overview	format(ascii):/tmp/dbms.txt
Spatial DataBlades	format(ascii):/tmp/2d.txt
Virage DataBlade	format(ascii):/tmp/vir.txt
Web DataBlade	format(ascii):/tmp/web.txt

## 8.2. Sample Text Searches

Select documents that include the word "image"

```
SELECT vname, VTSContains (vdocument, 'image') from VTSdocuments
```

A Verity function with syntax

VTSContains (columnname, querytext)

Returns a boolean value (TRUE or FALSE) indicating the result of the specified query

### Query Results

vname	VTSContains
DBMS Overview	t
Spatial DataBlades	t
Virage DataBlade	t
Web DataBlade	t

Select documents that include the word "image" ranked in the order of decreasing relevance

```
SELECT vname, VTSRank (vdocument, 'image') from VTSdocuments order
by 2 desc
```

A Verity function with syntax

VTSRank (columnname, querytext)

Performs a query on a text item and returns a value between 0 and 100

### Query Results

vname	VTSRank
Virage DataBlade	99
Spatial DataBlades	84
DBMS Overview	80
Web DataBlade	79

Select list of words starting with "data" from the Web DataBlade document:

```
SELECT distinct upper(Word) from
( SELECT unique VTSHighlight (vdocument, 'data*')
from VTSDocuments where vname = 'Web DataBlade' );
```

A Verity function with syntax

VTSHighlight (columnname, querytext)

Returns a table containing information about the location of all instances of the string specified by the query. Result table contains the following columns:

WordNumber	Wordoffset from the start of document
WordOffset	Byte offset from start of document
WordLength	Number of characters in word
Word	The word as specified by the query

### Query Results

upper
DATA
DATABASE
DATABLADE

The above examples illustrate only the approach taken by an ORDBMS model for text and document retrieval. There are many more capabilities offered by other third-party text management DataBlade modules. Some of these are shown in Table 5.

**Table 5**  
**Text based DataBlade Comparisons**

<p><b>I Verity Text DataBlade</b> from Verity Inc.</p>	<ul style="list-style-type: none"> <li>• Search-term stemming</li> <li>• Relevance ranking</li> <li>• Natural language searching</li> <li>• Proximity searching (phrases or words "near" each other)</li> <li>• Fuzzy searching</li> <li>• Supports a wide range of formats (MS-Word, PowerPoint, Excel, HTML, SGML)</li> <li>• Add-on DataBlades to include functionality in Verity topicSEARCH product</li> </ul>
<p><b>PLS Text DataBlade</b> j from Personal Library 1 Software Inc.</p>	<ul style="list-style-type: none"> <li>• Relevance ranking</li> <li>• Concept searching</li> <li>• Natural language searching</li> <li>• Wildcard searches</li> <li>• Proximity searching</li> <li>• Supports a wide range of formats (as in Verity)</li> </ul>
<p><b>Excalibur Text DataBlade</b> from Excalibur Technologies Corporation</p>	<ul style="list-style-type: none"> <li>• Performs neural-net searching. This means you can teach the engine that certain words or phrases have a relevance by correcting the output of searches. This enables the software to learn that seemingly unrelated words or phrases have conceptual relevance.</li> <li>• Phonetic searching (similar to fuzzy searching from Verity and PLS)</li> <li>• Supports wide variety of documents</li> </ul>

## 9. Picking the Right Universal Server<sup>7</sup>

The Aberdeen Group recommends several yardsticks for determining how well RDBMS vendors have implemented Universal Server Technology. The two factors to consider are 1) Degree of extensibility/flexibility and 2) Integration of Universal Server technology with the main components of the core RDBMS engine. Table 6 provides a checklist for the IS manager .

**Table 6**  
**The IS Buyer's Universal Server Checklist**

<p><b>Is it a Universal Server ?</b></p>	<ul style="list-style-type: none"> <li>• Does it support the major complex data types (e.g., text, video, audio, image, and spatial) ?</li> <li>• Is it extensible, e.g., via user-defined data types and open application programming interfaces (APIs) ?</li> <li>• Is it integrated with a distributed, open, scalable RDBMS (e.g., does it offer parallel scalability and replication technology for distributed database support) ?</li> </ul>
<p><b>How Effective a Universal Server is it ?</b></p>	<ul style="list-style-type: none"> <li>• Is it architected to be highly extensible and flexible , via a wide range of complex data types supported, with powerful development tools to create further extensions and with broad third-party and VAR support ?</li> <li>• Is the technology driven deep into the architecture - for instance, does the query optimizer understand specific complex data types and what to do with them ?</li> <li>• Does it deliver high performance and scalability for the complex data types that the enterprise needs ?</li> <li>• Do development and administrative tools support complex data types ?</li> <li>• Can RDBMS data-access operations (e.g., join) be applied across data types ?</li> </ul>

Source: Aberdeen Group, 1996

### Major Universal Server Players

With its acquisition of Illustra, Informix appears to be the leader in Universal Server technology. Full integration between its scalable Informix-Online architecture with Illustra DataBlade modules is scheduled for 4Q1996.

Oracle has folded its Video Server, ConText and Spatial Data options into Oracle 7.3, their version of the Universal Server. Each component is a distinct database server. It is neither fully integrated with Oracle7 nor highly extensible. For more extensive integration and user-driven extensibility, customer will have to wait for the company's "object" release, Oracle 8.0, in 1997.

IBM's DB2 Common Server (for OS/2 and Unix) offers functions to access parts of a data type, as well as the ability to insert a data type too large for main memory into the database. User defined datatypes and functions are supported through Relational Database Extenders. These extenders will support fingerprint analysis and querying by SQL of image content (color, shape or pattern).

Computer Associates<sup>7</sup> dual-database strategy include CA-Ingres and Jasmine, an OODBMS with a multimedia- and Internet-enabled toolset. CA has no plans to combine the two or otherwise offer Universal Server functionality.

Neither Microsoft nor Sybase yet offer support for complex data types, although Sybase has announced that, to allow independent software vendors to link snap-in complex data types with SQL Server System 11, it will provide an Adaptive Server combined with its ObjectConnect middleware.

## **10. Conclusions**

What will move relational databases into the next millennium are the simplicity of the relational model, the performance, and the number of existing systems that leverage the services of relational database engines. Relational databases are easier to design and understand than other database models such as the OO, hierarchical, and multi-dimensional database models.

As the demand for OO databases rises and the popularity of RDBMS remains, developers will continue to seek a quick fix via object-to-relational translation layers. This method offers the benefits of object orientation to new systems while retaining the information inside the legacy database. In the long term, the movement toward relational and OO data mixing (a.k.a. ORDBMS) will occur at the back end in the form of Universal Servers. Universal servers offer a choice as to how to present data to application. These databases will support legacy applications using traditional procedural and relational access methods, and they will support OO features at the same time. Enterprises can gain strategic advantage by using this technology for deeper data mining, better text and document management, multimedia and Internet and Intranet architectures, and adding complex-mathematics and data manipulation features to current customer-interface and back-office systems.

Long-term benefits, however, are likely to come from innovative functional or vertical-industry applications. To succeed in these, users should start learning the ropes in such areas as design, administration and scaling performance. IS managers should choose a Universal Server wisely, target strategic opportunities proactively, and begin planning and prototyping implementations.

## 11. References

1. David C. Blair, *CIS-577 Class Notes 1996*, University of Michigan Business School
2. Mike Norman, Robin Bloor, *The 1996 Object/Relational Summit: Recommended Reading*, <http://www.dbsummit.com/reading.htm>
3. David A. Taylor, *Object Oriented Technology, A Manager's Guide*, Addison-Wesley Publishing Company, 1990
4. *The Web Changes Everything, Marketing Material*, <http://www.ilustra.com>
5. Michael Stonebraker, *Object-Relational DBMS - The Next Great Wave*, Morgan Kaufmann Publishers, Inc., 1996
6. Craig Stedman, *Object Databases Lag*, Computerworld, August 19, 1996 pp. 43
7. Peter Kastner and Wayne T. Kernochan, *Universal Servers - RDBMS Technology for the Next Decade*, Special Advertising Supplement, Aberdeen Group, August 19, 1996
8. C. J. Date, *An Introduction to Database Systems - Sixth Edition*, Addison Wesley, 1995
9. Rex Hogan, *A practical Guide to Data Base Design*, Prentice-Hall, Inc., 1990
10. David S. Linthicum, *Objects Meet Data*, DBMS, September 1996
11. Atkinson, M., et al., *The Object-Oriented Database Manifesto/ Proceedings of the First International Conference on Deductive and Object-Oriented Databases*. Kyoto, Japan, 1989. New York: Elsevier Science, 1990
12. Stonebraker, M., et al., *Third-Generation Database System Manifesto/ ACM SIGMOD Record*. 19(3), September 1990
13. Won Kim, *Object-Relational Database Technology Whitepaper*, UNISQL Corp., <http://www.unisql.com/technology/papers/kim/whtpaper.html>
14. David C. Blair, *An Extended Relational Model*, Information Processing & Management, Vol. 24, No.3, 1988
15. ILLUSTRATE Verity Text Search DataBlade Module - User's Guide, Informix Software 1996