A REVIEW OF OBJECT ORIENTED DATABASE CONCEPTS AND THEIR IMPLEMENTATION.

Narciso Cerpa
School of Computer Science and Engineering
Faculty of Engineering
University of New South Wales
P.O. Box 1, Kensington
Sydney, NSW
Australia

Roy Dean
School of Information Systems
Faculty of Commerce and Economics

ABSTRACT

Object Oriented design and databases has attracted a great deal of attention in recent years. This article outlines and discusses the semantic data principles used *inter alia* in understanding Object Oriented concepts. To illustrate and lend substance to this discussion a list is presented of OODBMS implementations. Their weaknesses and strengths are analysed. And their suitability for specific applications is assessed. Finally we offer some conclusions about research in this area and the directions in which further development should proceed.

INTRODUCTION

There is a twofold purpose to this paper. Firstly to present an insight into Object-Oriented Database technology in both research and commercial arena. Secondly, to provide a view of Object-Oriented Database Management Systems (OODBMSs). The range and breadth of industrial and commercial needs today are beyond the technological grasp of the more traditional Data Base Management Systems. Most of these older systems were developed for commercial applications, traditionally in accounting, administration, resource exploration and financial support systems. Today's applications are wider in scope.
These newer applications include Computer Aided Design (CAD), Computer Aided Software Engineering (CASE) and Hypertext applications. They have specific characteristics and requirements and they require a different DBMS with capabilities honed to their needs. There have been a number of approaches to this problem. For example Stonebraker, et al (1990) in their third generation manifesto of DBMS attributes, suggest a Darwinian view. They opine that the contemporary evolution of relational technology provides equally efficient solutions. Consequently they propose a contemporary view of relational technology as an alternative to the "newer" generation of DBMS.
Others refute this proposition. Khoshafian et al (1986), Jackson (1991) and Garvey et al (1989) all argue that even though some extensions have been made to the relational model,(to overcome some of its limitations) more flexibility is still needed. They propose the Object-Oriented DBMS approach as an alternative to contemporary relational technology. Because Object-Oriented Database technology is based on the Object-Oriented Programming Languages, some of the language and its terminology, is presented within this paper. This paper also highlights the importance of OODBMS components, analyses some of the OODBMSs currently available and presents an overview of their implementation. The expectations of OODBMSs in the commercial marketplace, highlighting the areas which require further research, are discussed. In addition some major problems in OODBMS technology are presented. Finally, some conclusions on the results from the last decade of research on OODBMSs are given, outlining the achievements and presenting directions for the future.

THE OBJECT-ORIENTED APPROACH

The newer applications require:

- a flexible abstract data typing approach
- the capability to encapsulate data and
- the operations on this data by using the message metaphor.

A combination of the object-oriented capabilities with the traditional storage management functions, will result in a system which keeps the capabilities of the traditional approach, but includes an extensible data typing facility. This facility enables the storing of information usually not suited for normalised relations (Maier et al, (1986)). In object-oriented systems, all conceptual entities are modelled as *objects*. A simple integer or string is as much an object as is a complex assembly of parts. An object has value and/or attributes. More complex objects contain instance variables. The behaviour of an object is encapsulated *(caught)* in methods *(processes)*. (Banerjee, et al , 1987A). According some researchers and practitioners, (for example Khoshafian et al (1986), Jackson (1991) and Garvey et al (1989)) an OODBMS should fulfil all the functions which are expected of data base systems, plus others features which are typical in object-orientism. These can be described as follows:

Object Identity: For an object to exist in its own right it requires the characteristic of uniqueness. In OODBMS terms uniqueness is called identity. Khoshafian et al (1986), Jackson (1991) and Garvey et al (1989) all say that identity is that property of an object which distinguishes each object from all others. When real world objects are modelled, it is usually with some particular purpose in mind. A subset of that object's description *relevant to the purposes of the model* is included in the model
This subset is not always complete. To reflect the object's uniqueness sometimes artificial identifier is required. The relational model introduces the notion of user-defined identifier keys to represent the identity of an object. These researchers suggested implementing the object identity by a substitution method called *surrogates*. In this way the unique identifiers are generated. Each system has its own set of surrogates. Surrogates are completely independent of any physical location. Each object of any type is associated with a surrogate at the moment it is instantiated. This association will internally represent the identity of its object throughout the lifetime of the object *from creation to deletion*.

Inheritance:According to Paton, et al (1991) inheritance can be supported in three ways. The first kind of inheritance support, occurs when the structure of a superclass is inherited by its subclasses. The second kind occurs when the behaviour is inherited. The third kind is a combination of both. In Object-Oriented data bases, inheritance is possible by the modelling of IS_A relationships which are implemented by the OODBMS. (For a more extensive coverage of semantic data modelling see Elmasri and Navathe (1989), Hull and King (1987)). Inheritance reduces the need to specify redundant information, and simplifies updating. (Garvey, et al (1989)) Information about many object instances can be changed in a single update action.

Abstract Data Types are defined by Garvey, et al (1989) as data types which allow users to define new data structures. The object-oriented approach enables objects to possess attributes which are themselves objects (Jackson (1991)). This is achieved by using the ADT facility and building objects from existing data structures. ADTs provide a set of methods which may be used to manipulate the attributes of the object, but hiding the physical storage representation of the information.

Method and Message: a message is defined as the way in which objects communicate and perform all their computations (Garvey, et al (1989)). A message can be divided into the object, the method and the list of arguments It represents the interaction between objects. message can also be a request for an object to carry out one of its operations. Message sending supports data abstraction. The calling program does not make any assumptions about the implementation and data type used.

Overloading: occurs when the same operator can be used for the same operations on different data types. (Garvey, et al (1989)) So, distinct methods can be given the same name for two different classes of objects. Overloading is also related to the term polymorphism described above.

Late Binding: is a run-time interpretation of message passing, also called dynamic binding (Thomas (1989)). The language is interpreted, instead of compiled. (Garvey et al, (1989)) This allows features to be added to a method without affecting existing programs. The "how to do it" question, is resolved at run-time.

Encapsulation An object consists of an encapsulated representation or state and a set of messages which can be applied to that object (Thomas (1989)). This concept is also called data abstraction or modular programming. The data is packaged together with the procedures for accessing the data. This data can only be accessed by using the methods or messages provided by the implementer (Zdonik, et al (1990)).

Complex/Composite Objects: are built by applying constructors to basic objects (Jackson (1991)). Oxborrow, et al (1991) state that composite objects represent one kind of complex object whereas Kim, et al (1987) define a composite object as an object with a hierarchy of exclusive component objects. The instances which constitute a composite object belong to classes that are also organised in a hierarchy. This hierarchical collection of classes is called a composite object schema. Composite objects augment the semantic integrity of the object-oriented data base model through the notion of dependent objects. To implement a complex object, requires clustering and locking support.

Object Oriented Summary

A definition of OODBMSs was presented by Atkinson et al (1990) at the First International Conference on Deductive and Object-Oriented Data Bases. This definition included the features mentioned by researchers and practitioners as discussed above. To this definition Atkinson et al (1990) add the concept that the data manipulation language of the data base should allow the user to express any computable function. This is in sharp contrast with the current SQL approach in relational data bases where computational characteristics are very limited.
Other functions, which are described by some researchers include:

- the graphical interface for supporting the interaction with complex objects structures,
- and a version schema feature.

TECHNICAL FUNCTIONALITIES FOR OODBMS IMPLEMENTATIONS

There are technical functionalities which are critical to the success of all DBMSs including OODBMS. These functionalities include:

Persistence: Persistent data is described by Khoshafian (1990) as data which is stored outside a transaction context and so survives transaction updates. There are two principal strategies to determine whether objects should become persistent. These are persistence extensions and persistence through accessibility. The former incorporates the notion of a class extension to make the instances of a class persistent. The latter enables one or more persistent data base roots and every object, reachable from these roots, persistent.

Transaction Management and Concurrency Control: Garza et al (1988) present a good analysis of transaction management. They show how the transaction subsystem provides a concurrency control mechanism to allow interleaved execution of multiple concurrent transactions (serializability). The two-phase locking protocol is used. The granules for locking are the data base, the index, the class, and the instance levels.
Composite object locking is used for defining and manipulating a set of objects as a single entity. It also delivers semantic integrity. To achieve class lattice (network), and composite

object locking, new lock modes have been added to the conventional compatibility matrix for granularity locking. The aim is to increase efficiency of storage and retrieval functions.

Recovery: Garza et al (1988) describe a recovery process which supports recovery only from soft crashes (leaving the content of the disk intact). When this occurs user-initiated transactions abort. This does not support recovery from disk head crashes. The implemented approach uses the Undo from log procedure. This procedure requires that pages containing updated objects are forced to disk at the end of a transaction.

Security: is both mandatory and vital in OODBMSs. (Thuraisingham (1989)) It is largely based on subjects (processes) and entities (objects) with assigned security levels. The following rules apply:

. A subject has read access to any entity if the subject's security level dominates the security level of the entity.
. A subject has write access to an entity, if the subject's security level equals the security level of the entity.
. A subject can execute a method if the subject's security level dominates both the security level of the method and the type on which the method is defined.
. A method executes at the security level of the subject who initiated the execution.
. During the execution of a method M1, if another method M2, is executed, then M2 can execute only if the execution level of M1 dominates both the security level of M2 and the security level of the type on which M2 is defined.
. If a new object is created as a result of executing a method, the object is created at the security level of the subject who initiated the execution of the method.

This security system is the multi-level security system on which the subjects are assigned a level of security (e.g. unclassified < confidential< secret < topsecret) and the security is enforced based on the level of security of the subject and the object as created or accessed.

Data Access Performance

Effective clustering can have a significant impact on performance particularly when accessing data. Chang et al (1989) discuss effective clustering in an OODBMS based on a prototype object-oriented data manager developed by the UC-Berkeley CAD group. They propose using inheritance and structure semantics for achieving such improved performance, in the same way that the relational query optimiser uses the table cardinality and indexing information for producing efficient access plans. They also propose a run-time clustering algorithm, which, under high structure density and large read/write ratio conditions, can greatly improve system response time. A simulation study conducted by Cheng et al (1991) showed an application where a high read/write ratio justified a fully dynamic reclustering strategy. Conversely, when the ratio is not high enough, the reorganisation overhead degrades the overall performance.
Performance enhancement by using data replication has been proposed by Shekita, et al (1989). Here replicated data eliminates the use of functional joins that would otherwise be required during query processing. This technique is also called, field replication, and this should only be specified on reference paths which are frequently accessed although infrequently updated. Two strategies are proposed. They are:

- in-place replication, where the replicated values are stored directly in the objects that cause replication to take place, and
- separate replication when replicated values are stored in different objects that are shared within the set.

One of the issues is the propagation of updates for both strategies (in the same way as replication in the relational model). Kemper, et al (1990) propose creating redundant, and separate structures (*access* *support relations*) to store frequently traversed object references in data base queries. This *access support relations technique* is combined with a query optimiser to transform queries into expressions. This addition is needed to take full advantage of *access support relations.*

Query Facility

A proposed query model (Kim 1989) restricts the query target to a single class or a class hierarchy rooted at that class. This model excludes operations comparable to relational joins and set operations. It reflects the semantics of the class hierarchy. In an object-oriented data base, the operand and the result of the set operation may be a heterogeneous set of instances.

DEVELOPMENTS IN OBJECT-ORIENTED DBMSs

There are a number of implementations of OODBMS with both research and commercial applications. Each implementation differs. This is mainly due to the differences between the OOL which forms their genesis.
A list of OODBMSs follows with a short discussion outlining their weaknesses and strengths. The list is largely but not exclusively chronological in development. Languages and OODBMSs named in this list are proprietary. The characteristics of each item on the list correspond to the date of the research conducted by the cited researchers.

Gemstone

This data base system resulted from a project where the objective was to merge object-oriented language concepts with data base systems concepts. (Described in Maier, et al (1986)). Gemstone was mainly based on the Smalltalk-80 language, and its data definition and manipulation language, called Opal, was also based on Smalltalk-80 capabilities and features. Gemstone attempts to permit variations in structured objects, arbitrary data items as values and modification of data base schemes without physical data base restructuring.
Gemstone also provides a mechanism for accessing an entity as a unit. Gemstone provides most of the object-oriented features, but not totally. The implementation of Opal as the data definition language for Gemstone, had as an objective to implement a computationally complete data manipulation language . All data base access and operations use only Opal.
According to Dittrich (1986) there are three levels of object-orientism definition, namely structural, operational, and behavioural. Gemstone supports complex objects, and allows the definition of abstract data types. Therefore it is labelled as having behavioural object-orientism. However it does not supply multiple inheritance.
The Gemstone architecture, presented by Maier, et al (1986), provides most of the features of data base systems, such as secondary storage management, concurrency control, authorisation, transactions, recovery, and support for associative access. Concurrency is supported by providing each user with a workspace area. This area contains a shadow copy of the object derived from the most recently committed object table. The concurrency control schema used, is the optimistic schema, in which conflicts are checked at commit time, rather than prevented from occurring through locking. This schema ensures that read-only transactions never conflict with other transactions.
The recovery process is based on the use of the shadow and shared copy, not making use of logging files, but the important part of recovery is the garbage collection, when removing detritus of the transactions which had not committed before the crash. The authorisation access and ownership are based on segments, which are the areas assigned to users for the creation of new objects. A user can grant read or write permission on a segment to other users, being the grantor (i.e. the original owner).

Postgres

Postgres is described and defined by Rowe, et al (1987), as a relational model extended to include features such as abstract data types, relation attribute of type procedure, and attribute and procedure inheritance. These features can be used to simulate object-oriented modelling constructs such as aggregation and generalisation, complex objects with shared subobjects, and attributes which are able to reference tuples in other relations. Postgres has a relational database, taking the best features and

concepts from the relational approach (in this specific case, from Ingres) It includes new objectives to add semantic capabilities to the relational model.
Postgres provides better support for complex objects, extensibility for data type, operators and access methods. It also gives facilities for active data bases and inferencing, providing support for forward and backward chaining.This data base system provides a collection of atomic and structured types, where all atomic data types are defined to the system as ADTs. These ADTs are defined by specifying their names, lengths, input and output procedures, as well as their default values. The input and output procedures are written in a conventional programming language such as C. Although this ADT mechanism is provided by Postgres, it is limited in comparison with some implementation of abstract data types in some object-oriented programming languages. There is no inheritance mechanism for ADTs in Postgres.
Based on Dittrich (1986) object orientism definition, Postgres has some elements of both structural and operational object-orientism; particularly in the way that it implements ADTs and complex objects. However these characteristics are not combined. Its features agree with the third generation manifesto of Stonebraker, et al (1990). The Postgres architecture supports transaction management, but does not use the conventional write ahead log. (Stonebraker (1987)). The main characteristics of the approach used, are a collection of modules which provide transaction management. No recovery code is used. All updates are turned into insertions, and take advantage of specialised hardware. It is assumed that non-volatile main memory exists in some reasonable quantity
Concurrency control is implemented by the conventional two-phase locking protocol and using a main memory lock table. Postgres also keeps archival records on an archival medium by having an archival storage system compatible with WORM (write once - read many) devices. Postgres provides housekeeping to maintain the archive system free of invalid records. While Postgres is not classified as an OODBMS by its creators. However it is in this analysis for its capabilities in simulating some semantic and object-oriented features. All the features of Postgres agree more with the third generation manifesto, than with the object-oriented approach. But its variety of features could classify it as a very interesting multi-featured data base system.

Iris

Iris is a research prototype. Its main purpose is to meet the needs of current data base applications (e.g. CASE tools, CAD, etc) (Fishman, et al (1987)). Iris provides a different facility to other OODBMSs. It was designed to be accessible from any number of programming languages. Two lexically oriented interfaces are also supported. They are OSQL, an object-oriented extension to SQL, and Inspector, an extension of a LISP structure browser.
The Iris object manager provides the support for schema definition, data manipulation, and query processing. The data model is mainly based on constructs such as objects, types, and operations, providing support for inheritance, constraints, complex objects or normalised data, user defined operations, version management, inference and extensible data types.
Iris architecture is built over conventional relational storage manager, supporting transactions, concurrency control, logging and recovery, archiving, indexing and buffer management. The transaction management is based on the conventional two-phase locking protocol. Research is being conducted to improve concurrency, to provide prolonged access and to manipulate data base objects.

Vbase

Vbase is a commercially available data base system. It was built with a schema definition language concept where objects are defined as data types. These data types have attributes also defined as being of a specific type (Andrews et al (1987)). The main purposes of Vbase are to combine a procedural language with support for persistent objects. It also has the strong typing inherent in object systems for both language and data base.
Vbase has the most important influences from CLU, a programming language developed at MIT. Vbase is based on the abstract data typing paradigm, rather than the object/message paradigm. In Vbase object behaviour is required by a combination of properties representing static behaviour, and operations representing dynamic behaviour.
The Vbase architecture is based on a four layer approach, namely language layer, abstraction layer, representation layer, and storage layer. The language layer consists of the compilers for defining and implementing the behaviour of objects. The abstraction layer provides support for inheritance,

operation, dispatching, method combination, and property manipulation. The representation layer is the basis for the reference semantics. The storage layer is responsible for the management of objects and their persistence. Vbase provides features such as clustering objects on disk and in memory, triggers implementation, customised types, but very little time is given in this paper by the authors, to data base features, such as transaction management, concurrency, recovery, and others.

Orion

Orion is a prototype Object-Oriented data base system, used for supporting the data management needs of Proteus (an expert system). There are two versions of Orion, a single-user and multi-task system called Orion-1, and a multi-user and multi-task system called Orion-1S, in which a single server provides persistent object management on behalf of several workstations. Orion supports functions such as version and change notification, composite objects, dynamic schema evolution, transaction management, associative queries, and multimedia data management. Orion has been implemented in Common LISP, to produce the integration of a programming language with a data base system.
The Orion architecture consists of:

. a message handler receiving all messages sent to Orion objects,
. an object subsystem providing high-level functions, schema evolution, version control, query optimisation, and multimedia information management,
. a transaction subsystem providing concurrency control and recovery mechanism to protect data base integrity, while allowing the concurrent execution of multiple transactions. The conventional two-phase locking protocol is used for concurrency control. A logging mechanism is used for recovery from system crashes and aborts.
. a storage subsystem providing access to objects on permanent storage (disks). It also finds and places objects on pages and also manages the movement of pages to and from permanent storage.

Encore

Encore is described by Hornick (1987) as a data base system in which all objects are considered instances of some type which describe the behaviour of its instances. Types, operations, and properties are all objects in their own right and as such have a type that describe their behaviour. Types can be related to each other by means of the property IS_A, which induces an inheritance relationship between types. Encore supports multiple inheritance. Operations are active objects which are supported by programming code, and operation types correspond to a procedure definition, while the instances of operation types correspond to procedure activations. Properties are objects used to relate other objects. These properties are called association in semantic terms, and can also be used for expressing part-of relationships when modelling composite objects. Encore also allows dealing with changes, by providing a version control mechanism.

Other Implementations

O2 is based on the framework of a set-and-tuple data model. O2 was designed with the purpose of integrating data base technology with the object-oriented approach, all in one system (Lecluse, et al, 1988). This language is also data model where objects may have a tuple or set structure, or be atomic. These objects form a directed graph with potential cycles. Consistent sets of objects are used to interpret domains for type structures and method signatures. Types consist of a type structure and set of methods.
Cactis was built from scratch. The focus is on deriving values from both attributes within the object itself and in other objects (Heintz (1991)). Relationships are defined to allow accessing of information external to the object. Cactis does not support abstract data typing, except when an object refers to others through a relationship link. This system supports the specification of constraints and class/subclass hierarchies, also given capabilities for rolling back and recovering the effect of a change.

Development Summary

Most of the implementations of OODBMSs discussed in this section, try to meet in some way the object-orientism definition of Dittrich (1986). They do this by implementing different aspects of the object-oriented approach. Varietal approaches are heavily influenced by a specific language and/or the base framework taken for implementation. There are major differences in the physical implementation of each model as well as in the number of features implemented. This varietal approach is further compounded by major differences adopted by different researchers and practitioners. Some emphasise the abstract data type paradigm, while others emphasise the message and method paradigm. At this time there is no clear path for researcher or practitioner.

THE EXPECTATIONS OF OBJECT-ORIENTED DBMSs

From this discussion we can draw expectations of the OODBMSs. The aim is to make them more powerful than the current Relational model, for certain specific applications. The quest is not for a universal superiority. Nevertheless OODBMS need a strongly developed foundation if they are to succeed in the commercial arena. From the implementations described in these preceding pages, we believe that some of the issues which require solutions are:

. Rationalisation of Object-Query languages and programming languages
. Development of theory to support OODBMSs
. Include techniques from deductive and /or semantic data base technology
. Standardisation of implementation approaches.
. Keep or improve features of current data base technology

It is also expected that OODBMSs will be a good alternative to applications which require management of unformatted data such as image, picture, and voice, as well as alphanumeric data. In these applications, the implementation of the relation model is not feasible. Object-Oriented databases can satisfy the needs of users whose applications have requirements such as a variety of data types and type constructors, modelling accuracy, derived data, set value attributes, and the ability to model actions.
Current researchers in Knowledge Data Base Management Systems and intelligent databases often include OODBMS as a subset. These kinds of databases will represent a new technology for information systems management. They are the result of the integration of traditional approaches to data base systems, with new developments, such as Object-Oriented concepts, Expert systems, Hypermedia and Information retrieval.
Although some research has been done on query model, languages, and optimisation, there are still problems which must be resolved before OODBMSs can be commercially competitive. The current problems in this area, are related to the complexity of the data structure in object-oriented data bases, and to the lack of an equivalent of the simple and concise relational algebra used in the relational model. The interfaces used by DBMSs in general lack friendliness and quality. Most of the work on user interfaces has been related to specific implementations, or specific kinds of systems such as Decision Support Systems, Window-Based systems, etc, but the user interface topic, deserves to be treated as a separate technology.
In the case of OODBMSs the user interface plays an important role when representing complex objects, by using graphical capabilities or hierarchical structures. Other problems such as the representation of multimedia objects, manipulation of large objects and handling of active objects are also a concern in OODBMSs user interfaces. We believe that most methodologies for designing data bases cannot cope with object-oriented data base design. There are some object-oriented software design methodologies extant, but they include very little about object-oriented data base design. Computer aided software engineering tools in the same way as they exist for current data base models is a reasonable expectation.
The class hierarchy and class composition hierarchies can make the logical and physical data base design very difficult. The three level ANSI/SPARC architecture presented in the ISO standards, describes the view mechanism as a powerful tool in data base design (ANSI/X3/SPARC (1978)). The current implementation of OODBMSs does not support a mechanism for views, because of the lack of a complete query language, and the difficulty presented by the object identity concept. Views should have new object identifiers, but it is not clear if they should be persistent or transient. There

are other problems with views, even in current data base systems implementations (relational), but the powerful use of them made in the relational model indicate that they are necessary and that some research in this area is required.
Most of the current implementations of an OODBMS, are based on specific object-oriented programming languages (these languages in some cases are not compatible). This is very important, because a standardisation of the programming languages, could provide interoperability. This requires a common notion of object-oriented semantics, in aspects such as type, computation, query, identity and inheritance.
The performance issue has been a problem with the transition from one data base technology to the next, because making the programmer's job easier has resulted in performance cost. This problem was created with the transition to the second generation of data base systems (relational), because the use of declarative queries implies the need for some sort of system or expert system, to optimise the access path to the data base.
Object-Oriented data base research will also have to keep up to date with new researches in technology for relational data bases. This will include technology used by distributed databases, tools for data base design, etc. These new findings will require adoption to the object-oriented data base approach.
Based on the literature review, as well as from practical experience in current data base environments (hierarchical, network, and relational), the issues shown above are the main areas for research in object-oriented databases. The basic building blocks are in place but there are still some important areas which require improvement.

CONCLUSIONS

Object-Oriented data bases has become a major area of research. Early research results pointed the way to serious scholarship and experimentation. From these activities the attention of practitioners and researchers has become more focused. Data base technology has carried techniques from previous data base generations, such as concurrency control, recovery techniques, storage management, distributed data, and query optimisation. These techniques have been reused and improved, with the purpose of implementation in the object-oriented model. The development of new technology to support object-oriented databases, is also an important outcome of the research in this field. Technologies, such as indexing techniques, clustering techniques, schema modification, performance metrics, client/server architecture, etc., are in part, results of some research in object-oriented databases.
There are several commercial implementations, research prototypes, which were described in this review, and which were built with the purpose of experimentation and for evaluating the performance and functionalities of these object-oriented data base implementations, as well as creating an alternative to conventional DBMSs. Important definitions, such as the object-orientism by Dittrich (1986) , and the manifesto for object-oriented data bases by Atkinson, et al (1989) have set up touchstones. Greater formalisation of these standards at the level of international committees such as ISO task groups, will be required for making OODBMSs the next generation of data base management systems.
From the implementations described in this paper, Gemstone, Ontos (successor of Vbase), and Orion are OODBMSs which have received better reception from practitioners. These products have reached a level of maturity which make them very tempting in the marketplace. Another issue is the need for migration paths from current data base systems to object-oriented data bases. This will be essential for the success of OODBMSs in the commercial market. There is also a need for OODBMSs to participate in heterogeneous distributed data base systems. This could imply that organisations, use different kind of DBMSs for different purposes (e.g. relational for current or standard applications, object-oriented for design databases, etc). Thus the migration path becomes easier and more reliable.
We draw the conclusion that research on object-oriented databases, has been highly successful, contributing new technology for Information Systems, and improving current technology. Our understanding of systems development and data storage is constantly undergoing change. OODBMS are contributing to that change. These newer contributions are driven by commercial needs and research findings. This heady mixture fuels the ongoing debate and fires a high level of expectations for the next generation of DBMSs.

REFERENCES

Andrews T. & Harris C., (1987) Combining Language and Database Advances in an Object-Oriented Development Environment. **Proceedings of International Conference on Object-Oriented Programming Systems, Languages, and Applications**, Orlando, Florida, 1987. Sigplan Notices. 1987, Vol 22, No 12, 430-440.

ANSI/X3/SPARC Study Group Data Base Management Systems, **"Framework Report on Data Base Management Systems"** AFIPS Press, Montvale NJ, 1978

Atkinson M., Bancilhon F. & de Witt D. (1990) The Object Oriented Database System Manifesto. **Proceedings First Conference Deductive and Object-Oriented Databases**. Kyoto, Japan.

Chang E.E. & Katz R.H. (1989) Exploiting Inheritance and Structure Semantics for Effective Clustering and Buffering in an Object-Oriented DBMS. Proceedings of International Conference on Management of Data, **Sigmod Record.**, Vol 18, No 2, June, 348-357.

Cheng J R. & Hurson A R. (1991) Effective Clustering of Complex Objects in Object-Oriented Databases. Proceedings International Conference on Management of Data. **Sigmod Record.**Vol 20, No 2, June, 22-31.

Dittrich K. (1986) Object-Oriented Database Systems: the Notions and the Issues. **Proceedings of International Workshop on Object-Oriented Database Systems** IEEE, Asilomar, CA,

Elmasri R. & Navathe S. (1989) **Fundamentals of Database Systems.** Sydney : The Benjamin/Cummings Publishing Company, Inc.

Fishman D., Beech D., Cate H P, Chow E C, Connors T., Davis J W., Derret N., Hoch C G., Kent W., Lyngbaek P., Mahbod B., Neimat M A., Ryan T A., and Shan M C. (1987) IRIS: an Object-Oriented Database Management System. **ACM Transactions on Office Information Systems.**Vol 5, No 1, January, 48-69.

Garvey M A and Jackson M S, (1989) Introduction to Object-Oriented Databases. **Information and Software Technology.** Vol 31, No 10, December, 521-527.

Garza J F. and Kim W., (1988) Transaction Management in an Object-Oriented Database System. Proceedings International Conference on Management of Data. **Sigmod Record,** 37-45.

Heintz T J, (1991) Object-Oriented Databases and their Impact on Future Business Applications. **Information & Management.** Vol 20, 95-103.

Hornick M F. and Zdonik S B., (1987) A Shared, Segmented memory system for an Object-Oriented Database. **ACM Transactions Office Information Systems.** Vol 5, No 1, January, 70-95.

Hull R. and King R., (1987) Semantic Database Modeling: Survey, Applications, and Research Issues. **ACM Computer Surveys,** Vol 19, No 3, September, 201-255.

Jackson M S. (1991) Tutorial on Object-Oriented Databases. **Information and Software Technology.** Vol 33, No 1, January, 4-12.

Kemper A. and Moerkotte G., (1990) Access Support in Object Bases. Proceedings International Conference of Management Data. **Sigmod Record** Vol 19, No 2, June, 364-374.

Khoshafian S., (1990) Insight into Object-Oriented Databases. **Information and Software Technology.** Vol 32, No 4, May, 274-289.

Khoshafian S. and Copeland G., (1986) Object Identity. **Proceedings of International Conference on Object-Oriented Programming Systems, Languages, and Applications.** Portland, September. Sigplan Notices. 1986, Vol 21, No 11, November, 406-416.

Kim W., (1989) A Model of Queries for Object-Oriented Databases. Proceedings **of the Fifteenth International Conference on Very Large Data Bases,** Amsterdam, The Netherlands, 423-432.

Kim W., Banerjee J., Chou H., Garza J F., and Woelk D., (1987) Composite Object Support in an Object-Oriented Database System. **Proceedings of International Conference on Object-Oriented Programming Systems, Languages, and Applications.** Orlando, Florida, Sigplan Notices. Vol 22, No 12, December, 118- 125.

Lecluse C., Richard P., and Velez F., (1988) O2, an Object-Oriented Data Model. Proceedings of International Conference on Management of Data. **Sigmod Record.** 424-433.

Maier D., Stein J., Otis A., and Purdy A., (1986) Development of an Object-Oriented DBMS. **Proceedings of Object-Oriented Programming Systems, Languages, and Applications,** Portland, Sigplan Notices. Vol 21, No 11, November, 472-482.

Oxborrow E., Davy M., Kemp Z., Linington P., & Thearle R., (1991) Object-Oriented data Management in Specialised Environments. **Information and Software Technology.** Vol 33, No 1, January, 22-30.

Paton N.W., and Diaz O., (1991) Object-Oriented Databases and Frame-Based Systems : Comparison. **Information and Software Technology.** Vol 33, No 5, June, 357-365.

Rowe L. and Stonebraker M. (1987) The POSTGRES Data Model. **Proceedings of the Thirteenth Conference on Very Large Data Bases**, Brighton, England, 83-95.

Shekita E. J. & Carey M. J., (1989) Performance Enhancement through Replication in an Object-Oriented DBMS. Proceedings of International Conference on the Management of Data. **Sigmod Record.** Vol 18, No 2, June, 325-336.

Stonebraker M. R. (1987) The Design of the POSTGRES Storage System. **Proceedings of Thirteenth Conference on Very Large Data Bases.** Brighton, England. 289-300.

Stonebraker M. Rowe L. A., Lindsay B., Gray J., Carey M., Brodie M., Bernstein P., and Beech D. (1990) Third-Generation Database System Manifesto. **Sigmod Record** Vol 19, No 3, September, 31-44.

Thomas, D., (1989) What's in an Object. **Byte.** March, 231-240.

Thuraisingham M. B., (1989) Mandatory and Discretionary Security Issues in Object Oriented Database Systems. **Proceedings of International Conference on Object-Oriented Programming Systems, Languages, and Applications**, New Orleans. 203-210.

Zdonik S. and Maier D., (1990) **Fundamentals of Object-Oriented Databases. Readings in Object Oriented Database Systems.** Morgan-Kaufman, San Mateo, CA