University of Central Lancashire

Department of Computing

Advanced Database Systems

Module – C03701

The development of object-oriented databases and their strengths and weaknesses.

David George
20.03.2003

The development of object-oriented databases and their strengths and weaknesses.

This paper will seek to outline the rationale for the emergence of Object Oriented database systems and will then review the arguments for and against presented in a paper by Won Kim, founder of UniSQL (Kim 1991). Industry developments during the last 10 years will then be considered. This will be followed by an evaluation and conclusion.

Emergence

Data structures and information storage and extraction have become key issues in an increasingly complex data management environment that has demonstrated successive changes in the period 1960 to 1990. Early file systems were superseded by hierarchical and network based systems, which in turn, were displaced by relational database systems (RDBMS). Relational systems, based on the relational model by E. F. Codd (of IBM) in 1970, allowed database developers to more efficiently link and access related data. This was further superseded through the development of the entity-relationship model (Chen 1976), that provided "a special diagrammatic technique … as a tool for database design" and interestingly "incorporated some of the important semantic information about the real world". This was itself further improved by the extended-relational model, which was designed to capture more meaning (Codd 1979).

However, throughout these evolutionary stages, common themes of escalating costs of developing, managing and upgrading database programs have emerged as key issues. Modern business computing needs - related to information access, retrieval and modelling techniques - have placed significant demands in the last 15-20 years. For example, transport organisations may now wish to store detailed travel maps along with consignment information and there are now more sophisticated scientific and commercial requirements for such issues as the storage of sound, graphics, fingerprinting, knowledge based systems and computer software-aided design, engineering & modelling.

Whilst conventional systems have adequately supported routine data processing tasks such as invoicing, stock control and management support applications, they have been unsuitable in coping with increasingly complex real-world problems. It is accepted that RDBMS can support the storage and retrieval of binary large objects (BLOBS), e.g. images, audio clips and video streams, but relational systems do not have the capability of supporting the interrogation of the properties of the BLOB e.g. in an engineering application it may be necessary to replace one component in a 3D model by another but the user would have to first interrogate the component's properties to accurately identify it.

The following conceptual view of data management (see Fig. 1) is considered appropriate at this stage and depicts the key elements required in developing database systems. Clearly, the building block of the database system is the core data model.

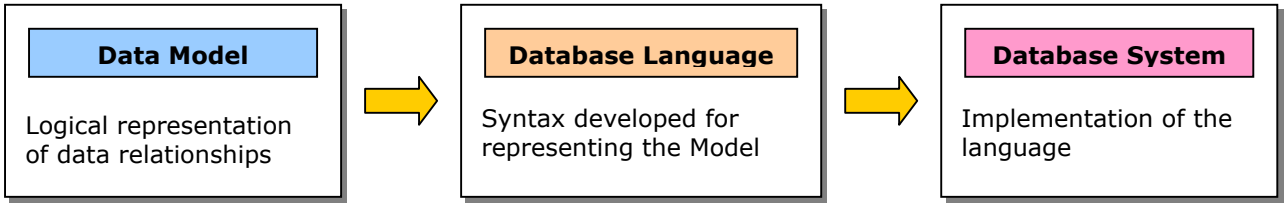

Fig. 1 – Database Implementation Conceptual Model.

The complexity and format of "new" data or object types demonstrated the need for data models that could more closely represent the real world. Further, a different approach was required to handle the way in which data entities, attributes and their relationships were stored and referenced i.e. through their representation as data "objects" in an object oriented (OO) approach.

The OO data model has its roots in Rank Xerox's Palo Alto Research Centre in the 70's (Rob, Coronel 2002), when the growing cost of developing complex programs focused efforts on achieving code reusability (Du, Wu 2001). The view was that an OO data model was a natural way forward in addressing complex data types. This, allied with rapidly expanding processing power, large-scale storage and high-performance compression (Cheng, Yang 2000), provided the platform upon which sophisticated transactions processing could be supported. According to Rob and Coronel, the first example of such a model was that developed by Hammer & Mcleod in 1978 and called the "Semantic Data Model" (SDM); semantic means capturing "meaning" (Codd 1979).

Thus, it was considered object-oriented databases (OODB) offered an ideal technology; being designed to handle the multimedia data highlighted above and which is now so frequently found on the Internet. It was considered that OODBM systems (OODBMS) would soon become the primary database technology – "the next revolution" (Bertino, Martino 1993), superseding conventional relational database-management systems (RDBMS). Indeed, we can now be interacting with OODBM systems when we use pagers, access voicemail or book a flight according to Doug Barry, Executive Director of the ODMG (Barry, Duhl 2001). And yet, in a paper published in the Journal of Object-Oriented Programming, Won Kim said, "no existing object-oriented database (OODB) system has delivered on the promises of object-oriented technology". Was this view justified and how has the evolution of OODB progressed since then?

According to Won Kim, the so-called "richer" data structures of sound, 3D etc. are, on analysis, "complex nested entities … best served by using semantic concepts". Data manipulation tends to be "compute" intensive, e.g. simulation programmes, requiring significant memory allocation, high performance specifications and extended processing cycles; justifications behind why OODB systems were the "paradigm" (Du, Wu 2001) shift that sought to satisfy scientific needs through exploitation of applications based on object-oriented programming language principles. They offered data models that not only represented data, relationships and constraints but also allowed "encapsulation" i.e. declaration of attributes/state and behaviours/methods that relate to an object, together with support to "hide" an object's internal attributes & methods (Du, Wu 2001 & Shoval, Shiran1997) – only methods can access instance variables, thus maintaining object integrity. In addition, OO introduced the concept of "inheritance" or "reuse" of state and behaviour i.e. the ability of an object, within a class of objects, to inherit the data structure and methods of the classes above it i.e. the super class of any object. Consequently, in Fig. 2, the paradigm of OODB systems emerges:

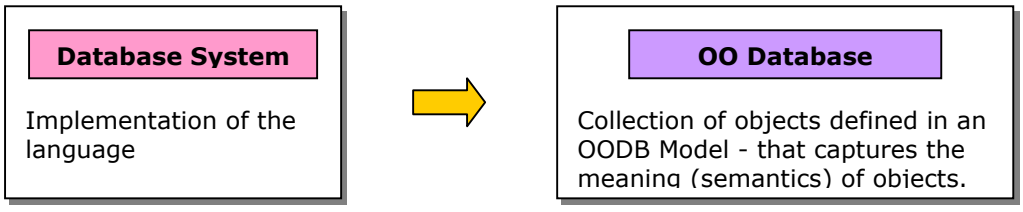

Fig. 2 - Emergence of OODB systems.

Whilst the concepts of encapsulation and inheritance addressed the challenge of reducing design difficulties and provided a means of creating large complex database

systems, in the early 90's they faced serious challenges. There were few application development tools available at the time, performance was an issue and the capability of the OODB model to handle complex structures was, in itself, an Achilles heel of "complexity" for the developer. Further, and often typical in new technology innovation, was the lack of industry standards on object semantics/data model (Leavitt 2000 & Shoval, Shiran1997). This restricted industry and client diffusion of OODB systems, which interestingly reflects the characteristics of the "innovation adoption" curve (Rogers 1995); the industry has yet to see technology "take-off".

Strengths of OODBMS versus RDBMS

Despite the above, what are considered to be the key features and advantages of OODBM systems? OODBM systems can provide dramatic performance improvements when managing rich data types, compared to relational systems (Rob, Coronel 2002 & Barry, Duhl 2001), although this requires platforms that allow efficient caching.

Kim said that one of the strengths of OODBMS was the ability to rely on the core object-oriented concepts of OO programming. A conventional relational database record or "tuple" equates to an object in an OODB, having attributes or state. However, it has been seen that objects also have the advantage of not only "encapsulating" state but also methods (Shoval, Shiran1997). Objects that have the same states and methods would form the basis of a class. Furthermore, those objects sharing the same class, e.g. car, bus of the class vehicle, through a class hierarchy (directed acyclic graph), would demonstrate "inheritance" of core states and methods e.g. in simple terms: number plate, gearbox, update and delete. Inheritance is one of the key features of OO design and is claimed to reduce the amount of maintenance and testing (Leavitt 2000 & Harrison 1999).

Equally, objects/classes can have multiple inheritance i.e. they can have more than one immediate parent, although this can lead to issues if similar variables and methods exist in each parent. Additionally, the class may modify or specialise an inherited method by redefining the method at subclass level e.g. method "bonus calculation" for "employee" may be defined as salary(*s*)**0.07* but for "manager" it could be salary *s**0.10* and for "sales" it could be (*s*+*commission*)**0.05*. Thus by calling the same method for different objects the system would generate different responses. This gives rise to "polymorphism" i.e. a method may be invoked differently by objects of different classes (Rob, Coronel 2002). It is this programming efficiency, through "re-use", that makes OODB systems much more powerful.

The grouping of objects (specialised subclasses) permits a semantic representation in the form of a generalised class (superclass) and vice versa. An examination of relational database structures reveals no similar inheritance or generalisation features. The hierarchy of classes and subclasses provides both an inheritance path, facilitating database design through re-use, and a means by which developers can define and modify classes without the need to explicitly program in changes for associated classes; thus providing significant programming flexibility.

In OODB an object attribute may be any class, be it a primitive data type e.g. integer, and also an object in its own right i.e. it may be a general class and have subclasses (aggregation). Therefore OODB attributes can have a single or, a set of values, whilst the "type" of an attribute is restricted to a finite group of primitive data types in relational database systems.

From a recursive perspective, the class hierarchy, as a complex nested object, is an instance of an aggregation hierarchy: i.e. what defines the superclass (or what it is) is the aggregation of all the object classes contained therein, each of which would have its own object identifier (OID). Therefore the combination of class and aggregation hierarchies allows developers to represent any data type because aggregation hierarchies can be cyclic. Fig. 3 demonstrates how this is represented in a simple business invoice and also illustrates OO semantic content compared to an entity-relationship schema.

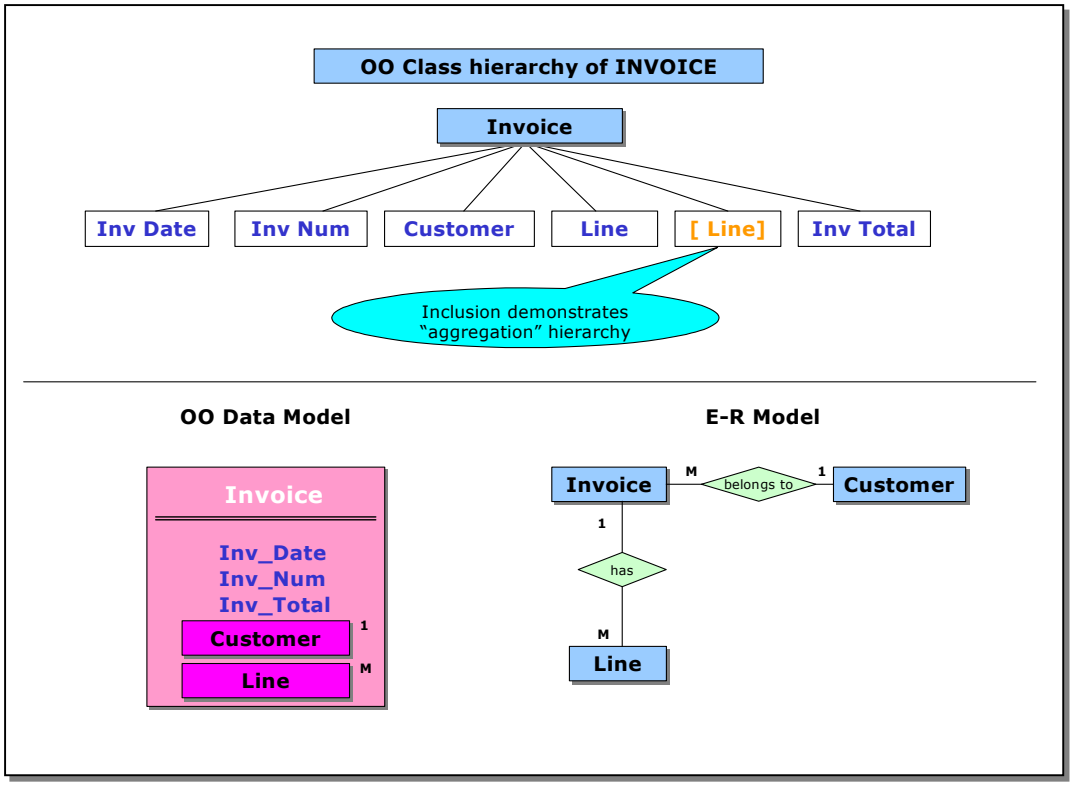

Fig 3 – Class hierarchy showing aggregation and semantic OO data model versus ER Model

The OO data model represents an object as a single block, containing the object attributes (themselves objects) together with relationships to other objects. In this case, "Invoice" groups all related objects within the same object box and it would be possible to add new data types e.g. telephone number, address etc. The OO data model clearly demonstrates semantic content, "providing a more natural and realistic representation" (Rob, Coronel 2002) of an invoice. This can be compared to the E-R model, which represents it as three entities and two relationships and does not readily convey the concept of an invoice object.

Won Kim identifies that a further advantage of OODB concepts is that OODB "states" and "methods" may be invoked from outside an object – i.e. such a request may be bound to a method contained in a superclass – whereas RDBM systems can only pass requests direct. OODB systems also support a feature called "versioning" i.e. the ability to track the history of changes in the state of an object. This is a powerful tool when used in CAD/CAE environments.

Finally, as in the hierarchical model, the OODM's inheritance serves to protect database integrity (Du, Wu 2001). OODM object autonomy ensures both structural and data independence i.e. the ability to make changes to the database structure without compromising the ability to access data (Rob, Coronel 2002).

Weaknesses of OODBMS compared to RDBMS

In the early1990s Won Kim considered that there were a number of issues as a consequence of the relative immaturity of OODB systems. These centred on the data model, database language and on the data model's complexity issues for developers.

At the time, proposals were being put forward for agreement on the data model standard, or high-level rules, through industry-wide efforts ostensibly via the Object Management Group (now O*Data*MG). The ODMG sought consensus on a reference data model for OODB systems, with little success (Leavitt 2000), and was supported by the similar activities of the ANSI SPARC OOBB Task Group. Proposals were also being put forward for OO extensions for SQL, in particular via UniSQL and Hewlett-Packard's IRIS initiative. Won Kim considered that ultimate progress on data model standards would also be dependant on an OODB vendor gaining significant market penetration or hinge on a major player's entry onto the OODB market such as IBM.

Nevertheless, are there really significant differences between OO and E-R? The OO data model could be argued to be simply an extension of the relational model because of the obvious similarities between a "relation" and a "class" e.g. classes are organised into a hierarchy because of the generalisation/specialisation *relationship*.

A database user relies on the ability to interrogate the system but in the 1990's OODB query language was relatively undeveloped and considered immature (Rahayu, Chang 2001), essentially because little attention had been given to what is termed "external schema" or organisation description; in relational systems this forms the basis of query language. The level of syntax complexity has further compounded this, when compared to RDBM query language, i.e. the OO data model's complexity of methods, class and aggregate hierarchy.

Development of new database systems in the past, particularly relational DB systems, has been characterised by programmers benefiting from improved functionality to assist in programming databases. In addition, the provision of query functions that could be managed by users has released developers from day to day data extraction programming. However, whilst the OO data model offers a richness of data modelling features this, in itself, has created a significant cognitive issue for users to grasp. This complexity demanded the support of friendly application development and design tools, for logical and physical database creation, and data browsers and report writers – tools that were unfortunately not readily available at the outset.

Last but not least, Won Kim highlighted that OODB systems in the early 90's were not fully comparable with relational systems in terms of providing for adequate security, transaction management, concurrency and recovery mechanisms.

Developments in last 10 years

So what has changed since Won Kim's paper? The issue of OODM standards has still not been resolved and of particular concern is the lack of a standard data access method. Different vendors support different access methods, resulting in compatibility issues when data is accessed from various sources. The lack of standards has continued to ensure that there is a steep learning curve for developer/users.

Whilst Object-oriented databases have shown their superior capabilities in more effectively modelling the real world, they are still regarded as immature when compared to their conventional counterparts (Rahayu, Chang 2001). This has coincided with the

emergence of a hybrid in the form of Object-Relational systems and this has probably been one of the reasons why relation-based systems continue to dominate database implementations, with some 90% of the market.

Object/Relational Database Management Systems (O/RDBMS)

Object oriented systems can be created with object databases or by using tools that give "transparent" access to objects that are stored in relational databases (Riccardi 2002).

O/RDBMS or Extended-Relational DBMS is effectively, the relational database response to OODBS (Rob, Coronel 2002). An O/RDBMS contains features of the OO model allied to a simplified relational database structure, thus providing a data model that is similar to the OODM in that it seeks to address the issue of semantic content. In addition, the object/relational approach supports reusable methods to enable table access and has demonstrated that it can be a superior combination than a pure relational methodology; providing improved real-world modelling capability (Rahayu, Chang 2001).

However, not everyone considers O/RDBMS an acceptable long-term solution. In a bulletin presented by the International Data Corporation, (McLure 1997), it was remarked that whilst O/RDBMSs would support some object extensions needed by complex applications, putting those extensions on RDBMSs was akin "to adding stereo radios and global navigation systems to horse-drawn carriages"! Nevertheless, it was accepted that because of the size of RDBMS vendors in the market, the O/RDBMS market would soon exceed the OODBMS market.

Products

O/RDBMS developments have been mainly targeted at business applications compared to the specialised engineering and scientific applications supported by OODBMS. Interestingly, the later versions of Oracle (Oracle 8 onwards) represent object-relational hybrids because they support both relational and object-oriented features.

According to the IDC vendors supplying O/RDBMSs range from Oracle through to UniSQL, Informix and IBM and clearly demonstrate the involvement of the major industry vendors, whilst OO vendors include Computer Associates (with its Jasmine system used by Toyota) and Versant (Versant ODBMS).

Evaluation

There can be little doubt that the semantic content realised in the OODM provides a far superior conceptual mapping with real world objects and developers with a sound understanding of OO programming principles should find that objects are a natural way to model and can accommodate a wide variety of types and relationships.

The reuse of software through inheritance is claimed to reduce the amount of software maintenance necessary and produce more understandable and reliable software. However, results from experiments (Harrison 1999) indicated that systems without inheritance were easier to modify than systems containing three or more levels of inheritance. It was also found that a system with no inheritance was easier to understand. And yet, in relational systems is there not hierarchy implicit in attributes within entities? After all "date" represents all possible dates and "number" can be single, double, integer, long integer etc.

OODB versioning clearly offers a powerful modelling feature especially in CAD. The facility to prototype designs, track the history of changes to the state of an object and then reverse undesired results, provides critical and core functionality for such applications.

Complexity and the steep learning curve have been levelled at OODBM systems. However it is considered that this is something of a myth, according to Doug Barry, (Barry, Duhl 2001), because using an OODBMS primarily requires knowledge of object programming languages. Whilst it can be argued there are additional database commands e.g. committing transactions and managing databases, the syntax is very similar to Java or C++.

On the question of processing speed, could it be simply that OODBMSs are simply focused more on handling more complex data than RDBM systems? Again, according to Barry & Duhl, ODBMSs are capable of running significantly faster than an RDBMS – a view supported by Rob & Coronel.

Market Research firm IDC, in their report No. 22542, "Enterprise Database Management Systems Market Forecast and Analysis, 2000-2004" identified global 1999 sales revenue of $11.1 billion for relational and object-relational databases, but only $211 million for OO databases. It expected these totals to increase to $15.6 billion and $265 million, respectively in 2001/2. Clearly a lack of progress in securing market penetration could hinder the adoption of OODB as a preferred technology and the problem may be more critical according to "Industry Trends" who quoted Michael Stonebraker of Informix: "ODBMSs occupy a small niche market that has no broad appeal. The technology is in semi-rigor mortis, and ORDBMSs will corner the market within five years" (Leavitt 2000).

Finally, an experimental measure of "design quality" (Shoval, Shiran 1997) - where quality was measured in terms of correctness of conceptual schemas, the time taken to complete the design task and designers preferences – concluded that the ER data model surpassed the OO model in all aspects.

Conclusion

Whilst OODBMSs are likely continue to occupy a niche in the market and remain a logical choice for such areas as specialised multimedia, CAD/CAM, medical applications and mapping applications, there can be little doubt that the more "conventional" O/RDBMS are likely to predominate. Indeed, Date (Date & Darwen 1998) believes "that a relational system that supported domains properly would be able to deal with all of those "problem" kinds of data that … OO systems can handle and relational systems cannot". He further says, "a true "object/relational" system is nothing more than a true *relational* system i.e. a system that supports the Relational Model".

OODBM systems represent just one strategy in addressing the challenge of storing complex data structures. They will have a part to play, along with the more conventional RDBMS solution and the hybrid O/RDBMS.

Finally, despite the creation of the object database standard ODMG 2.0, ODMG needs more vendors on board to make the standard an important factor in the industry and yet ironically, the lack of success in the development of ODMG standards across the OODB segment is unlikely to encourage more vendors into the market.

D George, ISD Year 3

References

Kim 1991	Kim, W., "Object-oriented database systems: strengths and weaknesses", *Journal Object-Oriented Programming*, Vol. 1, 4, July/August 1991.
Rogers 1995	Rogers, EM., (1995), "Diffusion of Innovations, 4th edition.", *The Free Press*, New York, ISBN: 0029266718.
Rob, Coronel 2002	Rob, P., & Coronel, C., (2002), "Fifth Edition - Database Systems Design, Implementation & Management – Fifth Edition", *Course Technology*, ISBN: 0-619-06269-X.
Shoval, Shiran 1997	Shoval, P., & Shiran, S., "Entity-relationship and object-oriented data modelling - an experimental comparison of design quality", *Data & Knowledge Engineering 21* (1997) 297 – 315.
Barry, Duhl 2001	Barry, DK., & Duhl, J., "Object Storage Fact Book: Object DBMSs Release 5.0", *Barry & Associates*, July 2001" URL: visited 10.3.2003, http://www.odbmsfacts.com/object-database.html
Rahayu, Chang 2001	Rahayu, JW., Chang, E., Dillon, TS., & Taniar, D., "Performance evaluation of the object-relational transformation methodology", *Data & Knowledge Engineering 38* (2001) 265 – 300.
Harrison 1999	Harrison, R., Counsell, S. & Nithi, R., "Experimental assessment of the effect of inheritance on the maintainability of object-oriented systems", *The Journal of Systems and Software*, 52 (2000) 173 –179.
Codd 1979	Codd, EF., "Extending Database Relational Model to Capture More Meaning", *IBM Research Laboratory*, ACM Transactions on Database Systems, Vol. 4, No. 4, December 1979, URL: visited 10.3.2003 http://www.scism.sbu.ac.uk/~rmkemp/codd1979.pdf
Date & Darwen 1998	Date, CJ., & Darwen, H.,(1998) "Foundation for Object / Relational Databases: The Third Manifesto", *Addison-Wesley*, ISBN: 0-201-309785
McLure 1997	McClure, S., *International Data Corporation,* Bulletin #14821E*,* August 1997, URL: visited 10.3.2003 http://www.cai.com/products/jasmine/analyst/idc/14821Eat.htm
Riccardi 2001	Riccardi, G., (2001), "Principles of Database Systems with Internet and Java Applications", *Addison Wesley*, ISBN: 0-201-61247-X
Chen 1976	Chen, PP., "The Entity-Relationship Model: Toward a Unified View of Data", *ACM Transactions on Database Systems*, Vol. 1, No.1, pages 9-36, March 1976, URL: visited 20.3.2003 http://bit.csc.lsu.edu/~chen/pdf/erd.pdf
Bertino, Martino 1993	Bertino, E., & Martino, LD., (1993), "Object-oriented Database Systems: Concepts and Architecture", *Addison Wesley*, ISBN 0-201-624397.
Leavitt 2000	Leavitt, N., "Whatever Happened to Object-Oriented Databases?", Industry Trends, August 2000, URL: visited 20.3.2003, http://www.leavcom.com/pdf/DBpdf.pdf
Cheng, Yang 2000	Cheng, P., & Yang, W., (2000), "Composition and Retrieval of Visual Information for Video Databases", Journal of Visual Languages and Computing, 12, 627-656 available via http://www.sciencedirect.com/science.
Du, Wu 2001	Du, T., & Wu, J., "Using object-oriented paradigm to develop an evolutional vehicle routing system", *Computers in Industry*, 44 (2001) 229 – 249.

Bibliography

1. Connolly, T., Begg, CE., (2001), "Database Systems: A Practical Approach to Design, Implementation and Management", *Addison Wesley*, ISBN 0 201 708574
2. Stair, RM., Reynolds, W., (2001), "Principles of Information Systems: A Managerial Approach, Fifth Edition", *Course Technology*, ISBN: 0-619-03357-6.