ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

OBJECT-ORIENTED DATABASES

ECMA TR/59

June 1992

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

OBJECT-ORIENTED DATABASES

ECMA TR/59

June 1992

BRIEF HISTORY

Object-oriented databases, or OODBs, are likely to be the databases of the future. A flurry of activity over the next few years is expected - not least in the area of standardization and development of commercial object-oriented database products.

These databases are capable of handling any type of data that may be represented as objects. Such type of data include text, graphics (still and moving images), and audio data. Popularity of object-oriented database is assured by virtue of user-friendly features: as an example, a single retrieval command would send appropriate data to the relevant port, be this connected to a printer, a screen or a loudspeaker system.

This ECMA Technical Report provides general background information on object-oriented systems, object-oriented databases and object-oriented database management systems, and identifies the issues related to databases. It is intended as a tutorial for novices.

Work on this ECMA Technical Report started in ECMA TC22, Databases, in the middle of 1990 and was completed at the end of 1991.

Adopted as an ECMA Technical Report by the ECMA General Assembly of June 1992.

Table of Contents

.

1 Scope

This ECMA Technical Report provides general background information on object-oriented systems, object-oriented databases, object-oriented database management systems, and identifies issues related to databases.

General background information on the object-oriented approach is provided, and specifically the following items are discussed:

- significance of current interest in object-oriented systems;
- issues to be addressed by object-oriented database developers;
- standardisation activity.

The intent of the report is to inform readers who are unfamiliar with object-oriented databases and highlight the related issues.

2 Structure

The rest of this report describes the object-oriented (OO) approach and a number of characteristics of the objects. It gives considerations to the current standardisation work and identifies a number of sources that were studied or have influenced this report.

3 Acronyms

DB	database	DBL	database language
DBMS	database management system	OO	object-oriented
OODB	object-oriented database	OODBMS	object-oriented database management system
OOPL	Object-oriented programming language	OOS	object-oriented system
SQL	structured query language		

4 Object-Oriented Approach

The advantages of using an object-oriented approach have been recognised for some time and stem from the basic concepts of an object-oriented system (OOS). Before considering object-oriented databases therefore, these basic concepts are introduced first.

4.1 Object

4.1.1 Black-box view of an object

The most fundamental concept in an object-oriented system is that of a self-contained modularisation into entities called "objects". An object is an entity that can be identified by a name, however, it should be essentially regarded as a "black box". As such, an object is characterised by the public interface offered to the outside world and the behaviour exhibited when messages are passed to the object via that interface. The internal workings of an object is hidden from the surrounding external environment. See figure 1.

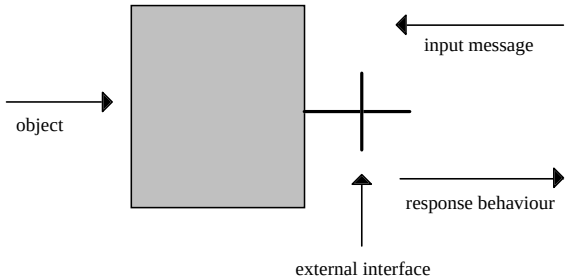


Figure 1 - Black-box view of an object

4.1.2 Object classes and instances

An object belongs to a class which identifies its type. A particular example of class of an object is termed an instance of an object. An object class may have subclasses below it. The object can then be referred to as the superclass of the subclass object and the classes form an hierarchy. The features of the class basic object are shared and propagated down the hierarchy of classes. See figure 2, giving examples of hierarchy of objects called shapes.

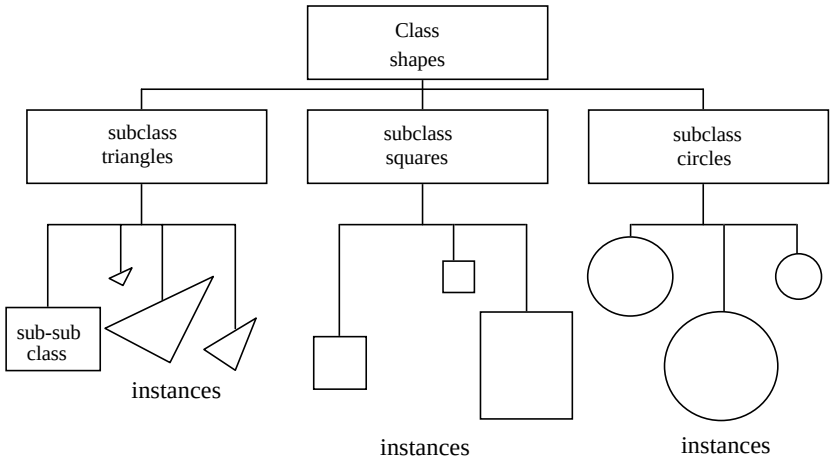


Figure 2 - Object classes and instances

4.1.3 Data abstraction and extendibility

Generic messages can be used to invoke specific methods on the hidden data. Table 1 is an example of the responses to a message, depending on the real structure of the object.

Table 1 - Relationship between messages and responses

Input message to an object: add (A+B)	Response
A and B real numbers	(A+B)
A and B complex numbers (a+jA) + (b+jB)	(a+b) + j(A+B)
A and B characters	string AB
A and B others (not recognised type)	reject message

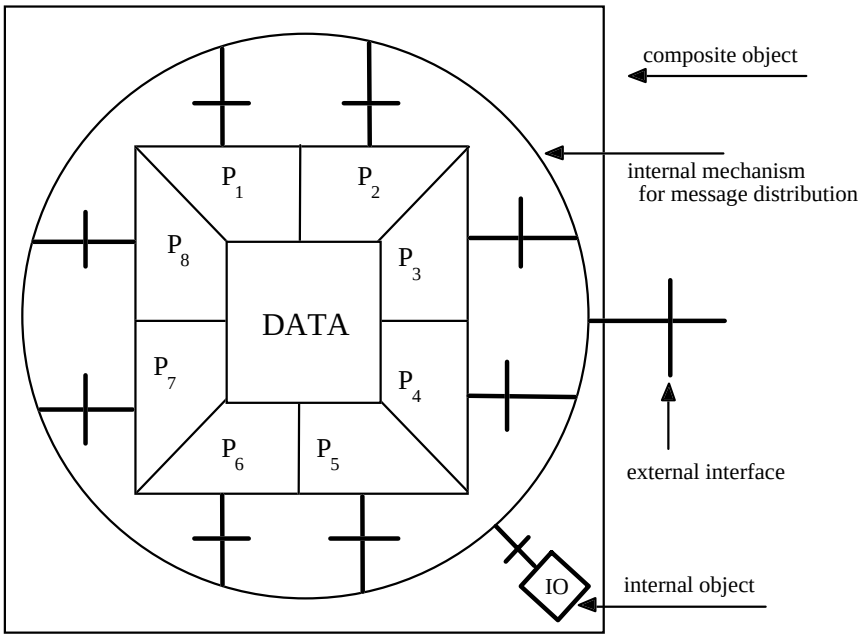
This characteristic makes the addition of new features to produce a new class of specialised objects from an existing one relatively easy.

4.1.4 Inside view of an object

The objects contain data and code for procedures to manipulate the data. These procedures are referred to as "methods" in the object-oriented terminology. This containment of data and procedure within an object is called encapsulation.

The data within an object can only be accessed by the internal procedure and in order to access and manipulate this data, an appropriate method needs to reside inside the relevant object, see figure 3. If the required method does not exist within the object then the request would be ineffective and rejected.

Any object can be a composite object made up of any number of one or more objects, not visible externally.



P_1 - P_8 - Procedures 1 - 8

Figure 2 - Inside structure of an object

4.2 Object-oriented programming languages

Writing programs using an object-oriented approach requires object-oriented programming languages (OOPL).

A number of object-oriented programming languages have emerged such as SIMULA, LISP, MODULA, ADA, SMALLTALK-80 and, as noted above, C++. None of them are perfect object-oriented programming languages, but each addresses some of the requirements . In addition, object-oriented features have been added to other popular programming languages such as PASCAL.

C++ and SMALLTALK-80 appear to be emerging as the leading object-oriented programming languages.

4.3 Benefits of object-oriented systems

4.3.1 Increased productivity

Object-oriented technology increases productivity through the software life cycle offering return on investment.

With the object-oriented approach time required for software system design phase is found to be reduced by as much as half with perhaps only a quarter of the effort in man-years being required compared with a conventional approach.

4.3.2 Models complexity

A physical system can be mapped on to a software design by identifying all significant physical objects and mapping them on to classes of corresponding software objects which could be arranged in a hierarchy.

The approach consists of defining an object class incorporating all the common functionality. Further subclasses are then defined for each type of object required in the system for its specific features that are over and above the common features.

4.3.3 Designed for change

This benefit arises from the fact that messages are used for communicating between objects. Thus if one type of object becomes obsolete, it can be replaced with an object providing 'higher functionality' which is a member of a newly defined subclass of the existing superclass. The main body of the system including the messages remain the same.

For example in 4.3.2, an object that is the instance of a newly defined subclass could replace an instance of an existing subclass.

4.3.4 Reusability

In the past libraries of subroutines have been available to software developers to handle standard tasks. Object-oriented systems provide scope for reuse on a wider scale.

It has been found that the amount of code needed for a new application could be reduced by as much as 5:1 through the reuse of existing object-oriented code.

4.3.5 Maintainability

In an object-oriented system, data and procedures are treated together as part of one package: the object. If data is changed all the procedures affected are easily identified and changed at the same time. Because change is limited to one area of the system, its impact is reduced.

Maintenance accounts for anything up to 80 per cent of the total life-cycle cost of a software system. Any technique that attacks this overhead is of benefit. The object-oriented approach not only makes it easy to maintain source code but also improves the reliability of the product.

4.4 Limitations of object-oriented systems

The limitations of present day object-oriented systems are largely due to problems of immaturity:

- limited availability across a range of standard platforms;
- the need for integration with existing systems and databases;
- lack of support for large-scale software development;
- object-oriented systems are limited in their scope by:
  - lack of persistence of objects;
  - the amount of memory available (real and virtual);
  - inability to share objects within a community of users;
  - limited form of version control;
  - no access to external data such as SQL database tables.

5 Object-oriented databases

Object-oriented databases (OODB) are required to store objects and an object-oriented database management system (OODBMS) designed for these databases could potentially solve all the problems identified above. So there is a considerable interest in object-oriented databases.

5.1 Viable object-oriented database solutions

Contributory factors to the upsurge in interest are listed in 5.1.1 to 5.1.4.

5.1.1 Popularity of C++

The gain in popularity of C++, a programming language from AT&T, has provided an increased interest in object-oriented programming languages. C++ is programming language C with object-oriented extensions added to it and which facilitates an object-oriented programming approach as a result.

C++ is significant for the fact that its object-oriented approach makes it possible to write re-usable code. Software development time can be reduced through this re-useability of code and considerable savings can result.

5.1.2 Looking for viable solutions

The relational model of data is found lacking in support for features of inheritance, deductions, and recursions, but users want their applications to embody these features and are looking for viable solutions.

5.1.3 Reduced cost

The cost of hardware has been falling and the current generation of hardware provides the computing power and amount of memory required by the object-oriented approach to databases in a cost effective way.

Early research work in the object-oriented approach had been started in the sixties but was found to be very expensive in terms of computing resources. Hardware technology was not advanced then and memory was very expensive.

Emergence of the relational model of data with its formal algebra was seen as the ultimate in database model and subsequently the object-oriented database work suffered a setback.

5.1.4 Progress in design management

Progress in computer aided design (CAD), and design management has also provided an impetus for the work on object-oriented databases especially if the large number of versions and generations that are created during the development process need to be controlled, managed and maintained.

5.2 Problems and issues to be resolved

The object-oriented system discussed above have tended to be software systems. Databases differ very significantly from system software.

A database comprises data records which persist a long time after the process that created them has disappeared. Databases also require consistency of data and ability to recover to a known consistent state after failure. The user application view of the logical database needs to be made effectively independent of the physical database implementation. Databases need to be available concurrently to a number of users. Other data models already have a large user base and any new data model such as the object-oriented database needs to co-exist with the previous models.

Database vendors are taking up these problems and are beginning to advance solutions to some of them.

As the object-oriented technology has not matured yet, a number of issues need to be satisfactorily resolved before a truly object-oriented database system can be developed. Some of these controversial issues are listed in 5.2.1 to 5.2.5.

5.2.1 Data type or data class.

Data types define structure of the data, such as whether it is a string of characters or a number. They are used as constructors of more complicated data types in programming languages. Details of data type definition may be used anywhere in the program. Data types do not provide any organisation of procedures.

Classes define the structure as well as the behaviour of their instances. Methods that implement behaviour are intimately bound to the class. The details of structure definitions are restricted to those methods that belong to the class. In addition, the class provides a mechanism to organise or group those methods which are common to a class.

Programmers (the C++ user community) favour the data type approach since data types can be validated at compile-time, while database designers (the SMALLTALK-80 user community) prefer data class as it is more of a run-time notion. This give rise to the problem that these two approaches are incompatible.

5.2.2 Persistence of data

In order for data to be available for the user after the execution time, data needs to survive the execution of a process in a database application.

When a program is executed processes within the program create data which will be destroyed on completion of the process. In a database it is often a requirement that the data persists (is captured) on completion of the process.

In an object-oriented system, an object is treated as a black box, the contents (including data) invisible and not directly accessible. Ensuring relevant data persist after the process that created it has been destroyed, therefore poses a challenge. Not only objects have to be stored, but a "garbage collection function" needs to be devised to purge the object store of inaccessible (redundant) objects

5.2.3 Database management systems and data retrieval

A database management function helps to decouple the application logic (user's view) from the physical implementation (disk storage, memory) of the database in a given system.

However managing large databases involves the use of mechanisms such as index management, data clustering, data buffering and so on to improve data retrieval time.

Using these mechanism locks the application logic into the physical implementation making it difficult to upgrade or modify applications at a later date.

With object-oriented databases, these mechanisms are not possible and hence the application logic and the physical implementation can be kept separate. But, then it becomes difficult to improve response times, as alternative mechanisms are not as yet developed.

5.2.4 Lack of formal model

No formal model based on rigorous algebra exist for object-oriented databases unlike the case with the relational database model. Identifying what is and what is not an object-oriented database is not easy and products claiming to be object-oriented databases may appear on the market even when they do not offer all the basic features of the approach.

5.2.5 Users resistance to new technology

As users already have a vast investment in implemented technology, understandably they are reluctant to move on to new technology before they feel the return on the investment has been justified for their existing implementations. Thus there is pressure on object-oriented databases to have an SQL (Structured Query Language) interface. SQL is not the ideal database language for an object-oriented database.

5.2.6 Heterogeneity

In a heterogeneous environment not all databases use the same technology i.e. not all are object-oriented databases or relational DBs. Some transactions may require data from a number of databases to be combined to form a 'user view' of the data. Users require this facility but solutions are not easy in such cases.

The heterogeneity issue will also be relevant to a distributed database environment.

6 Standards

6.1 International

In the international standardisation arena, object-oriented database management system are gaining increasing importance. Initially the scope of SQL3, the most recent version of SQL to be developed, will be expanded to include object-oriented features.

6.2 National standardisation activity

A number of groups, in USA and Europe, have attempted to define a reference model for object-oriented databases.

6.2.1 USA

The ANSI subcommittee X3/SPARC databases systems study group has released a common reference model for object-oriented databases.

6.2.2 Others

The Database Administrator Working Group (DBAWG), based in UK, has recently completed a report titled "Object-oriented database systems". This has been published by Open University.

7 Conclusions

The benefits of using an object-oriented approach has been recognised for a long time. When an object-oriented system is combined with a database that can store objects then benefits of the object-oriented technique can be better exploited.

A number of technical issues still remain to be resolved. Standardisation effort is ongoing to define a reference model, and object-oriented database languages.

Printed copies can be ordered from:

ECMA
114 Rue du Rhône
CH-1204 Geneva
Switzerland

Fax: +41 22 849.60.01
Internet: documents@ecma.ch

A description of this Standard can be found on the ECMA Web site, www.ecma.ch. From there, files T059-DOC.EXE (MSWord, self-expanding) and T059-PDF.PDF (Acrobat PDF) can be freely downloaded.

The ECMA web site gives full information on ECMA, ECMA activities, ECMA Standards and Technical Reports.

ECMA

114 Rue du Rhône
CH-1204 Geneva
Switzerland

This Technical Report ECMA TR/59 is available free of charge in printed form and as a file.

See inside cover page for instructions