An Object-Relationship Diagrammatic Technique for Object-Oriented Database Definitions

J. Bradley
University of Calgary

An object-relationship diagrammatic technique for communicating object-oriented database definitions is proposed and analysed. The technique permits diagrammatic display of objects in terms of system generated object identifiers, simple attributes, collection attributes, and both simple and collection reference attributes. Object-relationship diagrams are meant for final database designs with the OO approach, and are not intended to replace the entity-relationship diagrams commonly used in the early stages of data base design. They are intended mainly for ease of communication of complex database structure - particularly the inherently complex relationship aspects of object-oriented data base structure - to programmers and others involved in using OO databases.

The object-oriented (OO) approach to database management evolved from the object-oriented approach to programming (Abiteboul & Hall, 1987, Cardenas & McLeod, 1990, and Catell, 1991), and is finding use in such application areas as software engineering, document preparation and management, and in the design and production of engineering parts. Typically OO database systems involve a built-in object-oriented programming language that is procedural, as discussed in Bacilhon & Maier (1988), and Kim (1990), unlike non-procedural relational database manipulation languages like SQL, described in detail in Date (1990). Some current object oriented data base systems are Gemstone (see Bretl et al., 1988), ObjectStore, described in Object design (1990), Objectivity, described in Objectivity (1990), ONTOS, described in Ontologic (1989) and Versant, described in Versant Object Technology (1990).

A consistent feature of the object-oriented approach is that associated with every object representation, which crudely corresponds to a tuple in the relational approach (Codd, 1982 and Date, 1990), is a system generated identifier together with both lists of references, and individual references, to the object identifiers of related objects.

In an OO database, the relationships can be one-to-many, binary many-to-many, ternary many-to-many, recursive many-to-many, and ISA one-to-one relationships, as with relational databases. However, unlike relational databases, these relationships are defined by the supporting individual and lists of reference, and are all included in the conceptual database definition. Furthermore, the reference attributes supporting a relationship in the object-oriented database definition are two-way.

All this, despite the utility of the approach for many applications, does introduce a level of complexity in the database definition that is absent from the relational approach, at least as far as relationships are concerned. An undesirable consequence is that an OO database definition does not easily communicate the structure of the database. As a result, it can be difficult for a programmer attempting to manipulate an OO database to keep all the objects, their relationships, and supporting individual and lists of references in mind. Consequently, there is a need for an easily interpreted diagrammatic approach to OO database definitions that is relatively rich - from a semantic viewpoint. In other words, database definition diagrams should be able to capture a great deal of the database semantics, particularly with respect to relationships.

Currently, entity-relationship (E-R) diagrams, introduced by Chen (1976), are frequently used for communicating database structure. However, because E-R diagrams do
not explicitly incorporate the system generated identifiers and method of handling relationships that is unique to the OO approach, it can be criticized as being semantically poor - at least in the context of OO databases. The diagrammatic technique presented in this paper, or object-relationship (O-R) diagrammatic technique, is designed only for final database designs with the OO approach, and appears to satisfy the requirements of relationship-semantic richness and ease of interpretation in the OO context. This object-relationship technique has its origins in the extended Bachman diagrams that are fairly commonly used with relational databases in the final design stage. Extended Bachman diagrams, due to Bradley (1978), give explicit information about how relationships are supported, unlike E-R diagrams.

Extended Bachman diagrams were first introduced in the late 1970s with an experimental data model that attempted to combine the simplicity of relations with the owner-coupled set relationship concept of CODASYL (Bradley, 1978). The strong point of extended Bachman diagrams is that they capture more of the explicit support mechanism for relationships than most other types of database definition diagrams. Since the OO approach involves very explicit and quite complex support methods for relationships, this type of diagram, suitably modified, is a good candidate for OO database definition diagrams. The modifications are quite extensive however, so that the object-relationship diagrams presented in this paper do not look much like extended Bachman diagrams.

It is worth emphasizing that the object-relationship diagrams are not intended to replace the E-R diagrams commonly used in the initial high level design stage of OO database. Object-relationship diagrams are rather intended for ease of communicating the final OO database design, and for use by programmers when writing database manipulation programs.

Relational Version of the Project Database

The project database chosen to illustrate the ideas behind object-relationship diagrams concerns document management. A relational version of the database is presented first, as this will permit easy grasp of the semantics of the database, before the OO version need be discussed. The base tables for the database are illustrated by the extended Bachman diagram in Figure 1.

In an extended Bachman diagrams, a rectangular box is used for each relation, whether that relation represents a thing or a relationship, and the relation box is a string of further boxes that shows the primary key (underscored) and the attributes of the relation. A one-to-many (1:n) relationship is shown as an arrow from the primary key (or sometimes a candidate key) of the parent relation to the foreign key of the child relation. Many-to-many relationships are shown as composed of 1:n relationships, consistent with the way they are modeled in a relational database. Subtypes, taking part in IS-A-TYPE-OF or ISA relationships are shown as a line from the primary key of the subtype to the primary key of the supertype, with the line containing a bowl to symbolize the epsilon set inclusion symbol. As a result, E-B diagrams enable the relationship structure of a relational database to be evident at a glance, with the primary and foreign keys supporting any relationship being equally self-evident. The database in Figure 1 contains all the types of relationships that are encountered in OO databases. It is an extended and modified version of a database used by Cattell (1991) in a discussion of OO databases. It has one-to-many relationships as well as a binary many-to-many, a ternary many-to-many, a recursive many-to-many, and a subtype (ISA) relationship, this last relationship permitting utilization of the inheritance concept in OO databases.

The main relation is DOCUMENT, each tuple of which describes a document. A document can have many chapters, with each chapter represented by a tuple of the relation CHAPTER. A person can be both an author and a borrower of a document. A person can author many documents and a document can be authored by many persons. A person is represented by a tuple in PERSON and relation AUTHOANNITY enables the resulting many-to-many relationship between PERSON and DOCUMENT. A document can have copies in many libraries, and can be checked out of a library by a person. A tuple in the relation LIBRARY represents a library, and a tuple in the relation BORALIDITY represents a check-out of a document event from a library by a person. Thus BORACTIVITY enables the ternary many-to-many relationship between DOCUMENT, PERSON and LIBRARY.

A document contains references to other documents and to itself. A tuple in the relation REFERENCE describes a reference from a document (FROMDOC attribute) on a page number (PAGE attribute) to a document (TODOC attribute). Since a document can have many references there is a one-to-many relationship between DOCUMENT and REFERENCE supported by the foreign key FROMDOC. But, in addition, since a document can be referred to by many documents there is another one-to-many relationship between DOCUMENT and REFERENCE supported by the foreign key TODOC. These two 1:n relationships mean that DOCUMENT participates in a many-to-many relationship with itself, that is, in a recursive or cyclic many-to-many relationship. Thus there can be an explosion of references emanating from a single document: One document can reference a set of documents; each document of the set references a further set of documents, each of which references a further set of documents, and so on. Similarly there can be an implosion of references. One document can be referred to by a set of documents, each of which can be referred to by a further set of documents, and so on.