Many applications that use a database management system are required to hold temporal data, although few commercially available DBMSs provide temporal facilities. Furthermore, requirements specification techniques, such as structured methods and Object-Oriented Analysis, typically do not support a temporal modelling notation. This paper describes components of temporal modelling, such as granularity, events, and transaction time and introduces a temporal diagramming notation that is suitable for Object-Oriented analysis. Examples are furnished of class diagrams for time-stamped attributes and associations. A logical relational design for a class diagram is given that could be used as the basis for physical design of a relational database. The future direction is represented by temporal modelling of object reclassifications, which will require access to business model meta-data.

A common business requirement of management information applications is to retain a history of how data have changed over time. This is particularly the case with financial and management accounting applications in which a commonly found requirement is: to report current year results using the current chart of accounts; to report current year results using the previous year’s chart of accounts; and to report current year results using a future chart of accounts that will be introduced for the next financial year. Any operational computer system that requires a history of changes to be maintained for audit trail purposes, such as a medical information system, or for trend analysis, such as a history of budget forecasts in a Decision Support System, has a need for temporal modelling of data.

However, many modelling notations push the issue of time to the periphery. For example, in the Structured Systems Analysis & Design Method (SSADM), which is the most widely used system development method in the UK (see Goodland & Slater, 1995 for a description of SSADM Version 4 and Robinson & Berrisford, 1994 for an Object-Oriented approach to SSADM), the logical data structure does not usually feature time as an entity with the consequence that the need to keep history may be recorded separately as a functional requirement. This can result in a system designed on the basis of data structures that take little or no account of the need to hold temporal data. Where a need for history is recognized it might be accomplished through the introduction of naive denormalizations. For example, rather than record all the changes to a customer’s credit limit over time, an upper limit of three might be imposed, with limit1, limit2, and limit3 being held as attributes of a Customer entity type. Once a system has been built and a temporal requirement recognized belatedly then less than satisfactory work-arounds are often introduced, such as taking multiple copies of the database and using the different physical versions to produce the required temporal effects.

There has been considerable interest in temporal modelling in data models and databases and a number of surveys have been conducted of the literature, including (McKenzie, 1986; Roddick & Patrick, 1992; Snodgrass 1995). Time semantics have been incorporated in a number of conceptual modelling techniques and Theodoulidis & Loucopoulos (1991) describe a number of approaches, including: the infological data model (Langefors, 1973; Langefors & Sundgren 1975); the conceptual information model (Bubenko, 1977); the Time-extended Entity-Relationship model (Kloppegge, 1983); the Historical Database Model
(Clifford & Warren, 1983); the Entity Relationship Attribute Event model (Dubois et al., 1986); and TEMPORA, which has been developed as part of the European Union ESPRIT initiative (Theodoulidis et al., 1990; 1991). There has also been considerable work in the area of temporal query languages, such as ERT-SQL (Entity-Relationship Time Structured Query Language) (Snodgrass, 1987), including O-O extensions to query languages (see Snodgrass (1995) for a review of temporal query languages).

The interest in temporal modelling has grown throughout the 1980s and, although grounded in a semantic data modelling and database design tradition, an invitational workshop held at the University of Arizona in March 1994 reported that the participants felt that Object-Oriented data models provide the most appropriate basis for future work, while recognizing that temporal object bases are still in the early stages of development and commercial adoption (University of Arizona, 1994).

The first objective of this paper is to recommend a diagramming notation that can be used to capture temporal business requirements in an object model. Mainstream Object-Oriented analysis and design methods, such as that of Martin & Odell (1992), Coad & Yourdon (1991), and Rumbaugh et al’s OMT (1991) do not provide a notation for time-stamping and do not address directly the issue of modelling temporal aspects. Although the simplicity of the time-stamped object model makes it a useful medium for communication with business/user personnel and for the capture of requirements, the designer cannot avoid the complexity of time-stamped data structures when performing logical and physical design. Therefore, the second objective is to show how temporal requirements can be modelled in a non-temporal object model and hence form a basis for implementation in non-temporal environments. Examples of logical relational design for time-stamped data structures are also given since relational technology is still prevalent in practice, particularly in management information system applications. The third objective is to consider what contribution an Object-Oriented (O-O) approach can make to time-stamped data, particularly through the addition of encapsulated behaviour.

The structure of the paper is as follows: in section 2 some basic assumptions about temporal characteristics are described; in section 3 temporal modelling requirements are introduced; in section 4 diagramming notations and expanded class diagrams are developed; section 5 considers how an O-O paradigm might contribute to temporal modelling; section 6 shows how time-stamped data structures can be implemented relationally; and section 7 introduces the temporal modelling of state transitions.

Temporal characteristics

Before looking at temporal object modelling and class diagrams it is appropriate to consider some basic assumptions concerning time.

Granularity

If we are told that “John Smith became the owner of the motor car with registration A123 XYZ on 06-Sep-1994” then it will not be possible to ascertain the ownership of the car at 3:00 pm on 06-Sep-1994 since the change in ownership has been recorded using a granularity of date. If it is a requirement to know more precisely the point in the 24 hour period that transfer of ownership took place then a finer granularity might be introduced, such as minutes: “John Smith became the owner of the motor car with registration A123 XYZ at 15:25 hours on 06-Sep-1994.” If we assume that a car must have an owner, then the fact that the previous ownership ended at 15:24 is derivable. However, this leaves an indeterminate period of one minute where the ownership, as reported in a computer system, is indeterminate. This is not because ownership is necessarily in dispute in the real world - it arises as a result of the choice of granularity of time used to record the change in ownership. Yet finer granularities of time may be introduced (milliseconds, microseconds, nanoseconds, etc.) but, assuming that time is infinitely divisible, then the indeterminacy can never be removed entirely.

Events

Changes in data are attributed to events. For example, the event “car bodywork colour changes” results in a new value for the data item colourOfBodywork; the event “product price changes” results in a new value for the data item productPrice. An event is assumed to have a duration of one unit of time of the finest granularity defined to the temporal object model. Thus, although events should not be considered to be instantaneous (of zero duration), this assumption means that they do not need to be included as permanent aspects of the temporal object model. This is obviously a simplification that might be acceptable for some data items, such as productPrice, where a discrete change might well be a reasonable assumption. For other items, such as colourOfBodywork the assumption of a discrete change might not be sufficient. There can be a significant duration in changing the colour of a car of hours or even days. During this period the car is neither one colour nor another and if this is to be captured in the object model then it will be necessary to introduce a state that allows cars to be in the state “being re-sprayed” together with two events such as “re-spray begins” and “re-spray ends.” Whether this is necessary can only be decided based upon an analysis of the requirements that the information system is to satisfy.

Frequency

In addition to the granularity of the time interval it is also useful to specify how frequently the value of a data item may change. For example, a granularity of date could be used to...
Related Content

Applying UML and XML for Designing and Interchanging Information for Data Warehouses and OLAP Applications
[www.igi-global.com/article/applying-uml-xml-designing-interchanging/3305?camid=4v1a](www.igi-global.com/article/applying-uml-xml-designing-interchanging/3305?camid=4v1a)

The Expert's Opinion: Is the Webmaster Position Becoming Obsolete?
[www.igi-global.com/article/expert-opinion-webmaster-position-becoming/51198?camid=4v1a](www.igi-global.com/article/expert-opinion-webmaster-position-becoming/51198?camid=4v1a)

SeaDataNet: Towards a Pan-European Infrastructure for Marine and Ocean Data Management
[www.igi-global.com/chapter/seadatanet/166840?camid=4v1a](www.igi-global.com/chapter/seadatanet/166840?camid=4v1a)

Keyword-Based Queries Over Web Databases
[www.igi-global.com/chapter/keyword-based-queries-over-web/9206?camid=4v1a](www.igi-global.com/chapter/keyword-based-queries-over-web/9206?camid=4v1a)