Software Werkstatt

Bridge

- Relational and Object-oriented Model -

Inhaltsverzeichnis

1Database Models....3
1.1.1Advantages of ER Model....3
1.1.2Advantages of the OO Model....3
1.1.3When to Use which Model....4
1.2Object-oriented Model for Bridge....4

1 Database Models

Building an integrated metadata system requires storing a large amount of metadata. Typically there are not so many instances of metadata objects but many different object types. That requires an appropriate database technology to store and maintain metadata.

In general there are to alternative abstract database models available: the entity-relationship model (ER model) and the object-oriented model (OO model). The ER model is widely accepted and provides good data exchange features via standards as ODBC[1]. The OO model is becoming more and more interesting from the modelling and implementation point of view but there not so many applications based on object-oriented database models.

1.1.1 Advantages of ER Model

There are some important advantages of relational databases:

- Relational databases have become a type of standard and can be handled by most of IT experts.
- Relational databases are very stable because they exist and most of RDBMS are used in many practical applications since several years.
- Relational databases provide quick access to huge relations.
- Queries on relational databases can be defined in a common query language (SQL) that can be used for most of RDBMS.

Another more philosophical advantage is that the ER model allows reflecting a simple problem in a quite simple way.

1.1.2 Advantages of the OO Model

OO databases have their own and specific advantages as well:

- The OO model is richer in specifications, i.e. by using modelling facilities as complex attributes, inheritance, relationships you can express more in the OO model than in the ER model. One practical consequence is that there are no physical objects to introduce for reflecting M:N relationships.
- The OO model supports much more consistency rules on the database level (i.e. automatically). Those rules must not be implemented logically on the application level.
- The OO model is much more efficient traversing links along a sequence of linked object instances.
- The OO query language provides more powerful query facilities than SQL does.

The OO model requires an object-oriented view to the real world objects, i.e. the modelling is close to reality. Objects become more independent and are easier to handle if there are many object types and many relationships between objects.

[1] Open Database Connectivity

1.1.3 When to Use which Model

The theoretical answer is quite simple. The more complex a problem is the more appropriate an OO model is. The more instances are to be stored the more appropriate the ER model is. But this answer is to simple in many cases and it depends also on the complexity of queries, parsing of links and the facility of handling OO systems.

In many cases the last factor is the most important because it does not make sense to build an OO database and having no persons to develop or maintain it. This was also the reason to decide to use an ER model for building an integrated metadata system within the IMIM project, even though the problem is quite complex and has fewer instances. Most object types have less than thousand instances but there are more than 120 object types and more than 300 relationships between objects types.

Multilingual and version support causes additional complications and is difficult to handle in the ER model because of complications in the database model.

1.2 Object-oriented Model for Bridge

Many attempts to build complex metadata systems have been failed. Also the EMMA approach in the IMIM project could not be developed to an integrated metadata system. It was simply to complex for being implemented as an ER model. Then the EMMA model was enhanced to the Bridge model that was closer to reality but even more complex. A relational presentation of the Bridge model required about 300 logical relations and several hundred relationships between those relations.

A first limited thesaurus application with less than 5 relations made clear that relational databases are very slowly parsing relationships. At least there was no capacity available to take such a complex task and implement the system in a relational database. Companies asked for doing that estimated at least 120 MM for providing a system with the described features.

After deciding to implement Bridge in an object oriented database the total development capacity for implementing Bridge 2.0 with more than 300 forms on the graphical user interface was 36 MM. In contrast to relational database implementations the model has proved to be very flexible. Model extensions (expanding existing or adding new attributes) have been made within minutes without side effects to the applications (no database reorganisation, no recompilation).

The ODABA2 OODBMS that had been chosen for implementation provides general solutions for multilingual support and version control as well as a number of data exchange facilities to communicate with ER database systems, documents or ASCII files.

On the other hand it became clear that the system is not so fast processing large number of object instances. On a Windows 2000 server with 1 GHz and 256 MB RAM it took two seconds to read 2000 classification items, i.e. 1000 items per second. However, reading a classification item includes reading at least six object instances and parsing four relationships. Expressed in relational terms the system was reading in one second 2400 tuples and parsing 400 times 4 indexes (each with more than 100000 entries). Anyhow, speed is still critical and has to be improved for further Bridge versions.

Moreover, it turned out that the complexity of the Bridge data model makes it difficult for programmers in national statistical offices to understand the technical model in a short time (it is of course quite difficult to understand a data model consisting of 300 relations and hundredths of relationships within a few days). Thus, a high level interface becomes necessary to enable programmers outside the Bridge developer team to build its own Bridge applications.