DBDesigner: A Tool for Object-Oriented Database Applications

Shuguang Hong
Georgia State University

Joshua Duhl and Craig Harris
Ontos, Inc.

DBDesigner is a graphical front end tool for ONTOS, a commercial object-oriented database system. Through its graphical user interface, DBDesigner allows an application developer to design, browse, and modify a database schema, and to generate C++ header files from the schema. For the purpose of testing and prototyping, DBDesigner permits the user to create, modify, and delete objects in a database. DBDesigner can be a stand alone tool for the design of object-oriented databases or as a library module that can be linked into an application program. The unique approach of DBDesigner is to integrate these activities into a working environment for the development of object-oriented database applications.

Introduction

Because of the large scale and complex structures of applications built on the top of object-oriented databases (OODBs), the user, an application developer, not only have to manage the database schema and a large number of objects generated by application programs, but also need routinely to browse, modify, test, and debug the database. To help the user to manage the complexity and to improve the productivity, computer-aided tools for the development of OODB applications are required. The design and implementation of such a tool must consider the following major issues.

(a) Schema and object management. A mechanism must be developed to help the user store, retrieve, and modify a large number of classes, objects, and relationships.

(b) Data representation. An appropriate visual representation for the database schema and objects must be selected carefully to match the needs of the user.

(c) Prototyping. Interactively, the user should be able to design and modify a database schema and to test the schema by creating and modifying objects. In addition, programming code should be generated from the schema that can be directly incorporated into application programs.

(d) Database query. Query capability must be provided to the user for extracting information from a database in an easy and natural manner.

(e) Customization. The system can be tailored to the needs of different users with respect to methodology, environment, and design goals (Reiner, 1988). For example, the user should be able to customize the visual display of the
DBDesigner is a graphical tool for the development of OODB applications based on ONTOS, a commercial object-oriented database system (Andrews, Harris and Sinkel, 1989; ONTOS, 1989, 1990). DBDesigner provides the user with a visual representation of database schema and objects. A working environment is created that allows the user to design, browse, modify, and test database schema and objects and to generate C++ header files from the database. The tool also can be used as a library module and linked into an application program. Figure 1 shows a typical application development cycle using DBDesigner.

In the rest of the paper, the section Related Work provides a short survey about related work. Then, the graphical user interface of DBDesigner is introduced, and followed by a section devoted to the design and implementation issues. Lastly, the conclusion is given. To aid discussion, the terminology used in this paper is compared to other terms that are commonly referred to in other literature.

Class: Type

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Procedure</th>
<th>Object</th>
<th>Class Generalization/Specialization, IS-A, or Super-Hierarchy: type (-Class) / Sub-type (-Class ) Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute, Instance Variable</td>
<td></td>
<td>Operation, Method</td>
<td>Instance, Instance Object</td>
<td></td>
</tr>
</tbody>
</table>

### Related Work

Research prototypes based on graphical user interface for database systems have been reported in recent years, such as ADAM (Ellis and Demurjian, 1991), OdeView (Agrawal et al., 1990), RIDL (Troger, 1989), SNAP (Bryce and Hull, 1986), SIG (Maier et al., 1986), Gambit (Braegger et al., 1985), ISIS (Goldman et al., 1985), DBDT (Maryanski et al., 1985), LID (Fogg, 1984), SKI (King and Melville, 1984), and DDEW (Reiner, 1984). Since these systems were built mainly for research purpose, each of these prototypes focused on one or a few of the major design issues listed in Section 1. For example, LID concentrated on a graphical navigation methodology for database browsing. Although SNAP provided utilities for database browsing, schema designing, and simple query specification, it did not support prototyping and code generation. By contrast, SIG and OdeView permitted the user to construct a graphical display for databases, but other issues have been paid a little attention. ADAM allowed the user to design classes and relationships, and to specify database update propagations for which C++ code can be generated. But, database browsing and prototyping was not supported by ADAM. As a commercial product for solving real world problems, DBDesigner has addressed all major issues, such as schema design and modification, database browsing and prototyping, code generation, and tool customization.

The assumption of the user type is other issue that differentiates DBDesigner from most of the prototypes discussed above. For example, tools such as (Agrawal et al., 1990; Troger, 1989; Bryce and Hull, 1986; Braegger et al., 1985; Goldman, 1985; Fogg, 1984) assumed that the users are naive users. Based on such an assumption, those tools have emphasized “ease of use.” Because the users of DBDesigner are application developers, the design of DBDesigner must offer and maintain a balance between ease of use features and the degree of sophistication.

DBDesigner permits the user to customize some screen layouts. But, unlike those systems that are for constructing graphical user interfaces, such as ET++ (Weinand et al., 1988), InterViews (Linton et al., 1989), MERCY (Stary and Messner, 1991), Ingrid (Guimaraes, 1991), DBDesigner is a graphical database design tool. Hence, it has a different set of objectives and does not provide the similar capabilities as that of those systems.

Tools have been developed for aiding object-oriented programming, such as ObjectWorks (1990), GraphTrace (Kleyn and Gingrich, 1988), and OOPE (Borras et al., 1990). Those tools provided object-oriented programming.
Related Content

Assessing Business Process Modeling Languages Using a Generic Quality Framework
Anna Gunhild Nysetvold and John Krogstie (2006). *Advanced Topics in Database Research, Volume 5* (pp. 79-93).
[www.igi-global.com/chapter/assessing-business-process-modeling-languages/4387?camid=4v1a](www.igi-global.com/chapter/assessing-business-process-modeling-languages/4387?camid=4v1a)

Human Factors Studies of Database Query Languages: SQL as a Metric
[www.igi-global.com/article/human-factors-studies-database-query/51074?camid=4v1a](www.igi-global.com/article/human-factors-studies-database-query/51074?camid=4v1a)

A Case Study of One IT Regional Library Consortium: VALE - Virtual Academic Library Environment
Virginia A. Taylor and Caroline M. Coughlin (2006). *Cases on Database Technologies and Applications* (pp. 244-266).
[www.igi-global.com/chapter/case-study-one-regional-library/6215?camid=4v1a](www.igi-global.com/chapter/case-study-one-regional-library/6215?camid=4v1a)

Deriving Spatial Integrity Constraints from Geographic Application Schemas
[www.igi-global.com/chapter/deriving-spatial-integrity-constraints-geographic/11142?camid=4v1a](www.igi-global.com/chapter/deriving-spatial-integrity-constraints-geographic/11142?camid=4v1a)