An object relational database for hypermedia systems

Fayed F. M. Ghaleb a, Amani A. Saad b and Osama E. Sheta c

a Dept., of Mathematics, Faculty of Science, Ain Shams University, Cairo, Egypt E-mail: fmghaleb@yahoo.com
b Computer Science and Automatic Control Dept., Faculty of Eng., Alexandria University, Alexandria 21544, Egypt E-mail: amanisaad1@yahoo.com
c Department of Mathematics, Faculty of Science, Zagazig University, Zagazig, Egypt E-mail: osama_sheta75@hotmail.com

This paper presents an object oriented hypermedia database for designing and building an online magazine, which is treated as an enhanced hypermedia application. The main contributions are a direct result of taking advantage of the combination of Object-Oriented Hypermedia Design Modeling (OOHDM) and Object-Relational Database Management Systems (ORDBMS). The Unified Modeling Language (UML) is used to represent the model as it is now the standard language for modeling Object-Oriented (OO) systems. The class diagram representing the static part of the model is illustrated. On the other hand, the dynamic part of the model (system functionality) is illustrated by providing a set of use cases supported by the model. Furthermore, a mapping from the conceptual model to object-relational schemas in ORACLE8i is presented. This mapping has been targeted to utilize most of the object-relational features within ORACLE8i. As a result, the proposed Online Magazine provides a framework for creating, storing, processing, organizing, selecting, complex queries and distributing a digital journal. This journal is basically multimedia. Besides text, there are images, video, and audio data.

1. Introduction

It is a well-known problem that developers of hypermedia applications need assistance for modeling and maintaining application-specific hypermedia structures. In the past, various hypermedia engines have been proposed to support these tasks. Until now, hypermedia engines either have provided a fixed hypermedia data model and left extensions to the hypermedia application or they left the modeling of the hypermedia data completely to the application developer and only provided storage functionality which had to be plugged into the data model by the application developer. In this paper we propose an extensible object-oriented hypermedia engine which supports the specification of application semantics as application classes within the hypermedia engine, and supports complex operations maintaining application-specific as well as application-independent constraints.

In the Online magazine hypermedia engine, the storage layer and the application layer of a hypermedia system are implemented using the object-relational database management system ORACLE8i. This proposed engine provides the re-use of database functionality as well as the flexibility necessary to support the efficient development of different kinds of hypermedia applications. Conceptually, our approach presents a hypermedia engine for online magazines or digital libraries much more powerful than previous approaches [1,2].
Consequently, we can state that the online magazine gives a new quality to hypermedia application development.

Very little work has been done in the usage of Object-Oriented Database Management Systems (OODBMS) for hypermedia applications. The use of an object-oriented data model for a hypermedia system is an excellent marriage between the rigorous infrastructure and modeling abilities of object-orientation and the flexibility and navigational access to information characterized by hypermedia.

This paper presents an object oriented hypermedia database model for designing an online magazine, which is treated as an enhanced hypermedia application. The UML based model presented may be implemented using an OODBMS or an ORDBMS.

The organization of the paper is as follows: **Section two** presents hypermedia systems and the object-oriented hypermedia design model OOHDM. **Section three** discusses Database Modeling in UML. In **Section four**, the proposed conceptual model for the design of an Online Magazine is Illustrated UML, as well as the navigational design model using OOHDM. **Section five** presents modeling the Online Magazine functionality using UML by presenting some actors, use cases, and displaying some sequence diagrams for this model. **Section six** introduces the implementation issues for this model. Finally, **section seven** gives a summary and some potential future work.

## 2. Hypermedia systems

### 2.1. Hypermedia fundamental concepts

Hypermedia is an acronym which combines the words “hypertext” and “multimedia”. Hypermedia is a simple and natural extension of Multimedia and Hypertext: Multimedia provides a richness in data types that facilitates flexibility in expressing information, while hypertext provides a control structure that supports an elegant way of navigating through this data in a content-based manner [3].

Hypermedia is a way of representing and accessing information. It views the information space as a graph whose nodes store information and whose arcs (links) represent semantic relationships between nodes. The links are usually associated with words or image icons within the document that describe the meaning of the links. A set of nodes and the corresponding links makes up a hypergraph (or hyperstructure) which can be represented as a network. This is shown in fig. 1.

*Fig. 1. A Hyperstructure of a Hyperdocument.*
Node and node contents are independent objects, the association being made through a general referencing mechanism which allows the referencing of an entire document. As hypermedia systems are often used to structure information contained in pre-existing documents, the same document can be referenced by different nodes. Thus, the hypermedia model allows the sharing of node contents. When a hyperlink is activated (usually by double clicking the mouse button), the system extracts its destination node and correspondingly loads the document as the current document. Users navigate around the hyperspace by moving from link to link, or by specifying direct links [4].

Hypermedia systems have appeared as an evolution of traditional database and aspire to prove much more flexible and efficient for information storage and retrieval. The key difference between a hypermedia system and a traditional database system is not the broader spectrum of information that the former can accommodate but rather the fact that traditional database systems contain information that is structured, while hypermedia systems contain interconnected information [5].

A hypermedia system must serve two major tasks:

1. like a traditional Database Management System (DBMS), it must render the whole or an appropriate part of the stored information ready to be examined and processed by end users, and
2. Unlike a traditional DBMS, it must render the whole or an appropriate part of the connections between the stored information ready to be examined and processed by end users.

2.2. Object-oriented hypermedia design model

The Object-Oriented Hypermedia Design Model (OOHDM) is a model-based approach for building large hypermedia applications. It has been used to design different kinds of applications such as: web sites and information systems, interactive kiosks, multimedia presentations, etc.

OOHDM comprises four different activities namely, Conceptual Design, Navigational Design, Abstract Interface Design and Implementation. They are performed in a mix of incremental, iterative and prototype-based development styles. During each activity a set of object-oriented models describing particular design concerns are built or enriched from previous iterations [6].

Table 1 briefly describes OOHDM activities, products, formalisms, mechanisms, design concerns.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Products</th>
<th>Formalisms</th>
<th>Mechanisms</th>
<th>Design concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Modeling</td>
<td>Classes, subsystems, relationships, attribute, perspectives</td>
<td>Object-oriented modeling constructs (classes, relationships, use cases)</td>
<td>Classification, aggregation, generalization and specialization</td>
<td>Model the semantics of the application domain</td>
</tr>
<tr>
<td>Navigational design</td>
<td>Nodes, links, access structures, navigational contexts, navigational transformations</td>
<td>Object-oriented views object-oriented state charts context classes user centered scenarios design patterns</td>
<td>Classification, aggregation, generalization and specialization</td>
<td>User’s profile and task. Emphasis on cognitive aspects</td>
</tr>
<tr>
<td>Abstract interface design</td>
<td>Abstract interface objects, responses to external events, interface transformations</td>
<td>Abstract data views configuration diagrams ADV-charts design patterns</td>
<td>Mapping between navigation and perceptible objects. Composition and generalization/specialization</td>
<td>Model perceptible objects, implementing chosen metaphors, describe interface for navigational objects define lay-out of interface objects</td>
</tr>
<tr>
<td>implementation</td>
<td>Running application</td>
<td>Those supported by the target environment</td>
<td>Those provided by the target environment</td>
<td>Performance completeness</td>
</tr>
</tbody>
</table>
This paper is concerned with the conceptual design and the navigational design of a hypermedia system. The model is presented using UML and the implementation of this model is using ORACLE8i.

### 3. Database modeling in UML

Conceptual modeling is an important phase in designing a successful database application. The Entity-Relationship ER model is a popular high-level conceptual data model. This model and its variations are frequently used for the conceptual design of "traditional" database applications. Object modeling methodologies, such as UML (Unified Modeling Language) and OMT (Object Modeling Technique) are becoming increasingly popular. Although these methodologies were developed mainly for software design, a major part of software design involves designing the database that will be accessed by the software modules. Hence, an important part of these methodologies-namely, the class diagrams are similar to ER diagrams in many ways.

The UML distinguishes between the notions of diagram and model. A model contains all of the underlying elements of information about a system under consideration and does so independent of how these elements are usually presented. A diagram is a particular visualization of certain kinds of elements from the model and generally exposes only a subset of these elements detailed information. A given model element might exist on multiple diagrams, but there is one definition of that element in the underlying model. The notation and semantics for the following kinds of diagrams are supported in UML:

- Class diagram
- Use-case diagram
- Interaction diagram
- Sequence diagram
- Collaboration diagram
- State diagram
- Component diagram
- Deployment diagram

In this work, only a subset of the diagrams supported in UML is used. The Class Diagram is used to model the static aspects of the system. These are the important abstractions of the system (Classes) and how they relate to each other (Relationships). The Use Case Diagram is used to model the functionality of the system. It describes what the system can do at a very high level. A Use Case is a general pattern or strategy of using the system that may represent many related yet distinctly different scenarios. Thus each scenario represents a specific thread through a use case. The Sequence Diagram is an interaction diagram that is used to represent a scenario that is a dynamic aspect of the system [7].

### 4. The proposed UML model for an online magazine

#### 4.1. Conceptual design

The objective of this section is to propose a conceptual model that presents an Object-Oriented model for an Online Magazine. Furthermore, this conceptual model is designed using the UML. This conceptual model for an Online Magazine is independent of any specific implementation. Thus the model can be implemented as a relational database, an object-relational database, or an object-oriented database. However, in this work the model is implemented as an object-relational database.

These concepts are divided into subsystems that are presented by different Class Diagrams. A specific Class Diagram illustrates different relationships between a specific class and others.

As with other object-oriented methods, the Class Diagram is core to a UML model. A Class Diagram shows the important abstractions in a system and how they relate to each other. The primary elements found on a class diagram are class icons and relationships icons. Where the class is represented in the UML as a solid, outline rectangle with one, two, or three compartments. The class described in the previous tables can be represented using the UML class icons. The first compartment is for the name of the class and is required. The second and third compartments are optional and may be used to list the attributes and operations defined by the class. In the following subsections, all the concepts that have been discussed are modeled as class icons and relationships [8].
4.1.1. Modeling journal and front-page

Classes: This subsystem consists of class Journal and class Front-Page. Each class is presented with its attributes and operations or (methods).

1. Class Journal:
Description: this class presents a journal issue.
Class name: Journal
Attributes: Issue No.: a unique number for each issue.
Issue Date: date.
Methods: Create_Journal ()
Display_Journal ()

2. Class Front-Page:
Description: this class presents the front-page of a journal that contains more than one title.
Class name: Front-Page
Attributes: Title: shorthand textual description of content
Keywords: set of important concepts found in the content
Front_REF: A reverse reference to an object of class Journal
Methods: Get_Front_Page(): return all titles and keywords
Add_Front_Page(): add new object
Modify_Front_Page(): modifies some attributes
Delete_Front_Page(): remove object

4.1.2. Modeling Front-Page, Front-Detail, Front-Photo, and Front-Video

Classes: This subsystem consists of classes Front-Page, Front-Detail, Front-Photo, and Front-Video:

1. Class Front-Page:
Description: this class presents the front-page of a journal that contains more than one title
Class name: Front-Page
Attributes: Title: shorthand textual description of content
Keywords: set of important concepts found in the content
Front_REF: A reverse reference to an object of class Journal
Methods: Get_Front_Page(): return all titles and keywords
Add_Front_Page(): add new object
Modify_Front_Page(): modifies some attributes
Delete_Front_Page(): remove object

2. Class Front-Detail:
Description: this class presents the detail of a title which belongs to a front-page object.
Class name: Front-Detail
Attributes: Text: this is the text data of each title
Text_REF: composite object from Front-Page
Methods: Get_Front_Detail(): return the text by detail
Add_Text(): add new object
Modify_Text(): modify attribute
Delete_Text(): remove object

3. Class Front-Photo:
Description: this class presents the photo of title which belongs to a front-page object.
Class Name: Front-Photo
Attributes: Image Name: name of this image
Image Caption: shorthand textual caption of image
Image: the digital data of this image
Image_REF: composite object from class Front-Page

Fig. 2. Journal and front-page class diagram.
Methods: Get_Front_Photo (): return the photo
Add_Image (): add new object
Modify_Image (): modify attribute
Delete_Image (): remove object

4. Class Front-Video:
Description: this class presents the video of title which belongs to a front-page object.

Class Name: Front-Video
Attributes: Video Name: name of this video
Video Caption: shorthand textual caption of video
Video: the digital data of this video
Video_REF: composite object from class Front-Page

Methods: Get_Front_Video (): return the video
Add_Video (): add new object
Modify_Video (): modify attribute
Delete_video (): remove object

Relationships: There is an aggregation relationship between Front-Page and the three classes Front-detail, Front-Photo, and Front-Video. Where the relation between Front-Page with Front-Detail and, Front-Video is one-to-one and the relation between Front-Page with Front-Photo is one-to-many, in which Front-Photo fully-participates. This relationship is illustrated in fig. 3.

It is shown in figure 5(a-b) that there are other aggregation relationships in the system. For example between, Opinion, Local, Sports, Region, Economy, International and their composite objects.

4.1.3. Modeling Person, Journalist, Author, and Correspondent
Classes: This subsystem consists of classes Person, Journalist, Author, and Correspondent:
1) Class Person:
Description: this class is a super class representing the persons who are working in a journal.

Class Name: Person
Attributes: Name: shorthand textual description of content
Address: set of important concepts found in the content
Telephone: shorthand textual description of content

2) Class Journalist:
Description: this class is a subclass of Person

Class Name: Journalist
Attributes: Link1_REF: A reverse reference to an object of class Person
Link2_REF: A reverse reference to an object of class Person

Methods: None

3) Class Author:
Description: this class is a subclass of Person

Class Name: Author
Attributes: Link_REF: A reverse reference to an object of class Person

Methods: None

4) Class Correspondent:
Description: this class is a subclass of Person

Class Name: Correspondent
Attributes: Link_REF: A reverse reference to an object of class Person

Methods: None

Relationships: There is an inheritance relationship between class Person and classes Journalist, Author, and Correspondent. Where the super class is Class Person and the subclasses are Journalist class, Author class, and Correspondent class. This relationship is illustrated in fig. 4.

The complete UML Class Diagram is illustrated in Figure 5-a, 5-b. This model can be mapped into any standard logical design for any standard OODBMS.

4.2. Navigational design

In OOHDM, an application is seen as a navigational view over the conceptual model. This reflects one of the major innovations of OOHDM, which recognizes the non-conceptual objects (items) navigated by the user. These navigated items are kind of objects that are "built" from one or more conceptual objects. Navigation design is expressed in two schemas, the navigational class schema and
the navigational context schema [9].

4.2.1. The navigational class schema

The navigable objects of the hypermedia application are defined by a navigational class schema, whose classes reflect the chosen view over the application domain. There is a set of pre-defined types of navigational classes: nodes, links, and access structures.

In order to define Nodes as object-oriented views of conceptual classes defined during conceptual design, we use a query language. Nodes possess single typed attributes, link anchors, and may be atomic or composite (Notice that a node represents a web page).

Links reflect relationships intended to be explored by the final user and are also defined as views on relationships in the conceptual schema (a link represents an anchor (hyperlink) in web page). Access structures (such as indices or guided tours) are defined as classes and present alternative ways for navigation in the hypermedia application. Figure 6 contains a navigational class schema for the online
Examples of Nodes in this application, include:

The Node Class Journal
- Single attribute: issue_no and issue_date
- Anchor attribute: NXT and PRV which appear as buttons and are used to link the user to next or previous issue of the journal.
- Index attribute: CHP which represents an index of all subjects of the journal.

Also, the Node Class Front-Page:
- Single attribute: title
- Anchor attribute:
  - NXT and PRV as buttons and are used to link the user to next or previous item in the front-page.
  - Text which appears as buttons and is used to link class Front-detail.
  - Image which appears as buttons and is used to link class Front-photo
  - Video which appears as buttons and is used to link class Front-video
- Index attribute: CHP which represents an index of all subjects of the journal.

4.2.2. The navigational context schema
Navigational contexts organize the navigational space in consistent sets that can be traversed following a particular order; they should be defined in such a way as to help the user to perform his intended task. Each context contains definition and elements. Fig. 7 shows the context schema for the online magazine, where the Main Menu node represent an index that contains three other indexes Store Journal, Read Journal and Search Journal.

The context Store journal includes the definition store and the element: User name and By Subject. The element user-name represents a simple class node which is used to enter the user name and password of the administrator who can store a new journal issue. The By subject element represents a class group node which links to other contexts that represent all the subjects of the journal.

5. Modeling the online magazine functionality by UML

The OO model for an Online Magazine proposed in this paper facilitates the integration of the operations (methods in OO terminology) performed on the instance of different classes (Objects in OO terminology) with the objects themselves using the concept of encapsulation.

Thus for each class of objects proposed in this model, we have already presented the set of essential methods that can be sent as messages to objects of this class. In this section, the whole system functionality is illustrated as a set of Use Cases where each Use Case represents an intended functionality of the system and is actually implemented using a Scenario which is a sequence of message calls which are sent between objects that are possibly of different classes. The following list represents the supported use Cases in the proposed model:

- Create Journal
- Display Journal
- Add Front-Page
- Add Opinion
- Add Books
- Add Person
- Add Essay
- Add Region
- Add Economy
- Add Sports
- Add Local
- Add International
- Send Letters
- Send Cartoons and Comment
- Get Front-Page
- Get Opinion
- Get Books
- Get Person
- Get Essay
- Get Region
- Get Economy
- Get Sports
- Get Local
- Get International
- Search by Title
- Search by Author
- Search by Date
- Search Complex

5.1. Use case diagram

A Use Case Diagram is a rectangular box in which each use case in the system or a subsystem is represented by an oval. A dotted
Fig. 5-a. The complete UML class diagram of online magazine.
Fig. 5-b. The complete UML class diagram of online magazine (continued).
Fig. 6. Navigation class schema for the online magazine web site.
Fig. 7. The context schema for the online magazine web site.
The line between two use cases implies that one is dependent on the other. An Actor is an external entity that may initiate a use case (e.g., the DBA) or is the recipient of the system usage (e.g., the user, a display device, etc.).

The use case diagram in fig. 8 is given as an example but it represents only a small subset of the list of the supported Use Cases in the model proposed in this paper.

5.2. Sequence diagram

A Scenario is a particular path through the system functionality. A use case may represent many related yet distinctly different scenarios. A Scenario may be represented in UML by one of the Interaction Diagram supported in it that is the Sequence Diagram.

In this subsection, an example scenario is illustrated by its sequence diagram in fig. 9. This example is for the Display Journal Use Case.

First the Reader clicks the read button, which is expressed as a message from reader to Home_Page. You can see, the focus of control for Home_Page starts where the focus of control Reader ends.

Second is that the Home_Page calls that method by sending a message display, containing a call action, to the Front_Page.

Third the Reader clicks the title button, which express as a message from reader to Front_Page then the Front_Page send a message display to Front_Detail.

Fourth the Reader clicks the back button, which express as a message from reader to Front_Detail then the Front_Detail send a message display to Front_Page.

Finally, the Reader clicks the local button, which is expressed as a message from reader to Front_Page to read local_Page.

6. Implementation issues

Object types (also known as abstract data types or ADTs) can be used as abstractions of real-world objects. Object types consist of two main components: Attributes: Contain built-in types or other object types as values. Attributes model the structure of the real world entity.

Methods: Functions or procedures written in PL/SQL or an external language like C are stored in the database. Methods implement the operations the application can perform on the real world entity.
VARRAYs and Nested Tables are two new types that allow a structured collection of data to be the type of a table column.

*REFs* (or object references) are used to store logical pointers to objects.

*LOBs* are large arrays of bytes with additional system-defined properties.

The following section elaborates on the use of the object-relational features of the underlying DBMS (Oracle8i) and the generation of ORACLE8i schemas from UML conceptual class diagram.

6.1. Mapping UML conceptual diagrams to Object-relational schemas in ORACLE8i

1. Classes map to object tables, where the type of an object table must first be defined as an object type.
2. Single-valued relationships between objects (i.e., the 1 side of a 1:1 relationship) map to: References between objects using REFS.
3. Multivalued relationships between objects (i.e., the M side of a 1:M relationship or either side of a N:M relationship) map to: Structured collections such as VARRAYES or nested tables.
4. If object reference information is explicitly maintained on both sides of a relationship, triggers can be generated to maintain inverse relationships automatically.
5. Object types can be created to represent user-defined structures, enumeration types, or other complex types that may be needed within an application. Such types can then be used to create embedded objects within relational and object tables [10].

The following subsections present examples of these mapping procedures and address some of the mapping issues of which a designer must be aware in the mapping process.
Related to our case study, we describe in what follows how the conversion is made.

6.2. Table design

6.2.1. Converting object classes to tables

Each class in the object-oriented model to be implemented is mapped into a table, where each column stores an attribute, and row corresponds to an object of that class. A distinguished attribute may be used as a database key, or an object identifier, which corresponds to an object handle, that can be generated and used as a key.

Each object class maps directly to one table. All object attributes become fields of that table. Our data modeling methodology provides strong support for the notion of object identity. Each object has a unique ID; all references to objects are made via the ID. Object identity is implicit in object diagrams and must be made explicit in tables.

6.2.2. Converting associations to tables

As in step 2 above, we map associations to distinct tables, fig. 11 shows an association relationship between class Journal and class Front-Page.

![Fig. 10. Converting class journal to an object table.](image1)

![Fig. 11. Converting associations to tables.](image2)
Create an object table for this aggregation relationship:

1- Create the types for each part

Create or replace type `front_detail_t` as object
{
    text        varchar2(1000),
    d_ref       REF front_page_t
};
/

Create or replace type `front_photo_t` as object
{
    image_name   varchar2(10),
    image_caption varchar2(300),
    image        blob,
    ph_ref       REF front_page_t
};
/

Create or replace type `front_video_t` as object
{
    video_name   varchar2(10),
    video_caption varchar2(300),
    video        blob,
    v_ref        REF front_page_t
};
/

2- Create tables for each part

Create table `front_detail` of `front_detail_t`
(scope for (d_ref) is front_page);

Create table `front_photo` of `front_photo_t`
(scope for (ph_ref) is front_page);

Create table `front_video` of `front_video_t`
(scope for (v_ref) is front_page);

---

6.2.3. Converting aggregations to tables

When we map aggregations to tables, every piece of the part must be assigned to the whole. Fig. 12 shows aggregation relationships between class Front-Page, and each of class Front-Detail, class Front-Photo, and class Front-Video.

6.2.4. Converting inheritance to tables

The inheritance relationship is a fundamental concept in any object-oriented system. Type Inheritance allows sharing similarities among types as well as extending their characteristics.

- Superclass/subclass relationships can be mapped to the use of either:
- A superclass variable that appears in subclass to establish class hierarchy relationships.
- An object type that represents the flatten class hierarchy with relational views established to simulate the subclass structure.

Oracle supports the single type inheritance model. This is closely aligned with the ANSI SQL99 standards [10].

The inheritance relationships between class Person, and subclasses Journalist, Author, correspondent is shown in fig. 4. Fig. 13 illustrates the mapping (implementation) of these relationships in ORACLE8i.
Create an object table for this inheritance relationship:

1-Create the super type class Person

Create type Person_t as object (  
  name varchar2(100),  
  address varchar2(100),  
  telephone varchar2(100),  
  photo blob,  
  email varchar2(100),  
  persone_ref journal,  
  member function addperson (),  
  member function deleteperson (),  
  member function modifyperson () ) NOT FINAL,

2-Create the subtype classes: Journalist, Author, and Correspondent

Create type journalist_t under person_t (  
  link1_ref opinion,  
  link2_ref essay );

Create type author_t under person_t (  
  link_ref Books );

Create type correspondent_t under person_t (  
  link_ref Letters );

3- Create Tables for all classes

Create table person of person_t;
Create table journalist of journalist_t;
Create table author of author_t;
Create table correspondent of correspondent_t;

Fig. 13. Converting inheritance to tables.

Type hierarchy:

In Oracle8i, the type hierarchy is created using the CREATE TYPE statement and the super type should be declared to be NOT FINAL.

A subtype can be created under a non-final type. It inherits all attributes and methods from its supertype. It can add new attributes and methods and / or override inherited methods.

7. Summary and future work

This paper presented an object oriented hypermedia database for designing and building an online magazine, which is treated as an enhanced hypermedia application. The main contributions are a direct result of taking advantage of the combination of hypermedia modeling and object oriented database management systems.

Among the main results we can cite:

The Unified Modeling Language UML is used to represent the model as it is now the standard language for modeling OO systems. The Class Diagram is used to represent the several classes in the model, as well as, the relationships between them. The functionality of the system is shown as a list of Use Cases and a subset of the Use Case diagram is presented. Moreover, an example scenario is illustrated to show how to implement one of these use cases.

The mapping defined in this research has been targeted to utilize most of the object features within the underlying DBMS. In particular:

- Object types support the creation of abstract data types that correspond to class definitions in UML and to other, user-defined complex types that are needed for attribute definitions.
- Object tables are created from object types and used to hold persistent instances with object identifiers that correspond to UML class objects.
Embedded objects are used to define complex attribute values from user defined object types.

Object references (REFs) are used to define relationships between objects.

Object type hierarchies are used to define inheritance relationship.

VARRAYS and nested tables are used to implement multivalued attributes and multivalued relationships between objects.

Triggers are used to maintain inverse relationships between objects.

Relational views instead of triggers are used to simulate the functionality of UML class hierarchies.

Indeed, the Online Magazine provides a framework for creating, storing, processing and organizing complex hypermedia application. Furthermore, complex queries can be submitted to hypermedia data supplied by the Online Magazine.

References


Received December 31, 2003
Accepted May 30, 2004