https://doi.org/10.48009/2_iis_2008_538-543

USING UML FOR OBJECT-RELATIONAL DATABASE SYSTEMS DEVELOPMENT: A FRAMEWORK

Ming Wang, California State University, ming.wang@calstatela.edu

ABSTRACT

Data model of object-relational databases (ORDBs) is a great challenge to many database application practitioners due to its complexity. Unified Modeling Language (UML) offers tremendous flexibility and rich expressivity for modeling ORDB systems. This paper proposes a framework of using UML to model ORDBs systematically. The intent of the paper is to provide a guideline for database application practitioners to develop ORDBs more efficiently.

Keywords: UML, Data Model, Database Development, System Analysis and Design

INTRODUCTION

The emergence of object-relational database (ORDB) technology into the commercial database market has caused the database professional's attention in seeking a modeling tool for it. Traditional entity-relationship diagram (ERD) for conceptual modeling of relational databases can no longer represent object-relational database features. ERD modeling is static and offers no means to depict the dynamic aspects of ORDBs and their data interaction.

UML offers great potential for ORDBs data modeling. The Version 2 of the UML defines more than a dozen of diagrams, such as use case diagram, class diagram, activity diagram, sequence diagram, state diagram and communication diagram. These diagrams are ideal to represent multiple perspectives of ORDBs by providing difference types of graphical diagrams during different stages of ORDBs development. Not only these diagrams can model both static and dynamic aspects of ORDBs, but also inheritance, encapsulation, and other object-oriented features of ORDBs. There are many techniques for transforming UML models to ORDB systems. While no one technique has emerged as a leader, it is reasonable to expect that one technique may be more suitable than others, when applied to the same problem [2].

The paper proposes a framework that applies UML components to ORDBs development workflows as defined in United Process (UP) [4]. The paper presents the techniques utilized in the framework three sections: ORDBs analysis, ORDBs design, and ORDBs implementation. Oracle ORDBMS is used as a tool to illustrate ORDBs implementation. The paper concludes with a discussion of the advantages and implications of data modeling ORDBs with UML.

UML AND ORDBs ANALYSIS

This section presents how to apply UML techniques to the ORDBs analysis workflow. UML is a use case driven modeling language. As shown in Figure 1, ORDBs analysis starts with development of use-case narratives. Based on use case narratives, use case diagrams are developed to illustrate the system functionality visually. Both sequence and class diagrams can be derived from the UML use case. The sequence diagram helps the use case generate the class diagram during the analysis workflow.

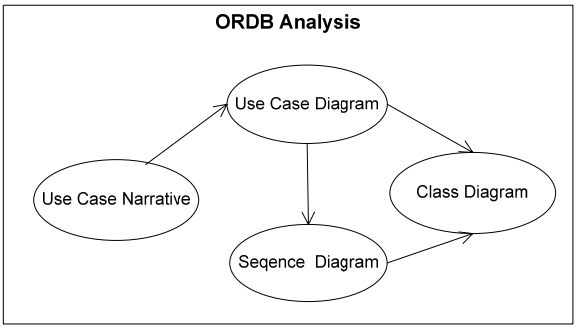

Figure 1 UML Components in the ORDBs Analysis

Use Case Narrative for Place Order

A use case narrative is a text-based description that portrays the basic functions of the system. Use case narratives catch what end-users expect ORDB systems to do from the output of system requirements. During the requirements phase, customer purchasing activities and product data are collected. The Place Order Use Case Narrative is developed based on the specification of system

functional requirements. Figure 2 shows the flow of events generated in the Place Order use case.

Use Case: Place Order ID: 2
Actor: Customer Preconditions: Customer has invoked "Browse catalog"
Flow of Events
1. Customer adds additional item to shopping cart 2. Customer selects "review order" 3. Customer receives calculation of total cost, tax and shipping charges 4. Customer selects "Place order" 5. Customer clicks "Submit Order" to update the system 6. Customer's receipt is created
Post Conditions: Customer leaves the screen or the Web site.

Figure 2 Use Case Narratives: Place Order

Use Case Diagram for Place Order

As specified in Figure 1, the use case diagram is derived from the use case narrative. The Place Order use case diagram in Figure 3 is derived from Figure 2 and further expanded by adding the normal flow of events specified in the above use case narratives using <<use>> and <<extend>> notations.

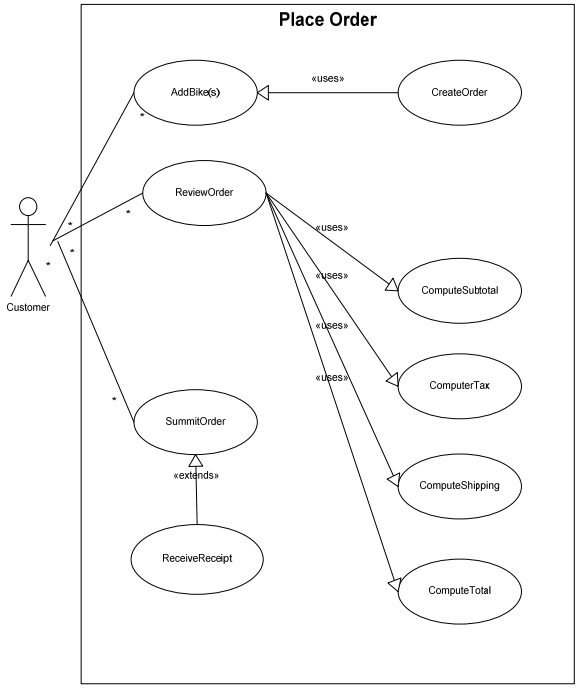

Figure 3 Use Case Diagram: Place Order

Sequence Diagram

Figure 4 shows the Place Order sequence diagram.

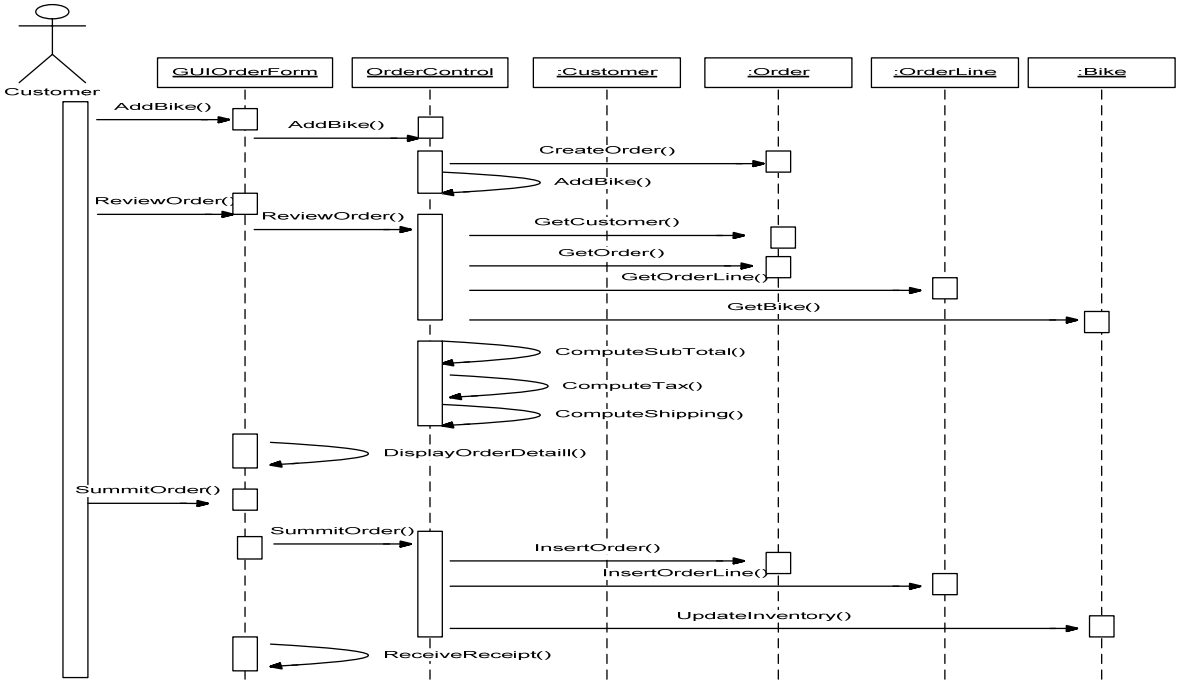

Figure 4 Sequence Diagram: Place Order

Sequence diagrams are derived from use case diagrams – resulting in one sequence diagram for each use case. Sequence diagrams illustrate the objects that participate in a use case and the messages that pass between them over time in a use case. Objects:Classes in sequence diagrams are derived from a list of nouns identified from the use case diagrams including actors

Class Diagram: Place Order

UML class diagrams can encapsulate data and their methods within a class and effectively depict the design of object-relational database features. A list of nouns identified including actors from use case diagrams are class names. The use case identifying technique is to check which use case has what classes and verify which classes are responsible for creating, retrieving, update and deleting methods. Messages in the corresponding sequence diagrams are member methods encapsulated in each of the classes. Creating a use case-sequence matrix is to generate a class diagram that contains both static and dynamic aspects of ORDBs. Without mapping from a use case-sequence matrix to a class diagram, the class diagram might miss some elements either in sequence diagrams or use case diagrams easily.

	Customer	Order	OrderLine	Bike
Create order	CRU	C	C	R
Review order	R	R	R	R
Submit order	RU	RU	RU	RU

Figure 5 Use Case-sequence Matrix for Place Order
Note: C = Create, R = Retrieve, U = Update, D = Delete

In short, the analysis workflow starts from inception phase of the UP life cycle and increments in the elaboration phase and decrements in the construction phase. In a subsequent iteration, additional use cases would be described, designed, implemented, tested and integrated with the previous development. Most of the use cases, sequence diagrams and some class diagrams are completed in the ORDB analysis workflow. The construction of one diagram depends on information provided by another diagram. The development of a new diagram often helps refine and correct the previous diagram.

Besides use case narrative, use case diagram, sequence diagram and class diagrams, activity diagrams and state charts may be needed to describe the dynamic aspects of ORDBs in the analysis workflow optionally. An activity diagrams can be developed to depict the place order case and how object (data) move in the ORDBs system. The activity diagram is needed to refine business processes and its object data flow in a logically complicated use case. The state chart is needed to analyze the state of an object only if many messages pass between certain objects in a sequence diagram. If a class has many different incoming and outgoing messages and appears in several different sequence diagrams, then we assume the class plays a critical role in the system. State chart is to be developed to show the different state of the order object.

UML AND ORDBs DESIGN

The design workflow refines and structures the static and dynamic aspects of ORDB systems from analysis workflow into class diagrams for ORDBs design. In this workflow, problems in traditional relational database such as multi-valued attributes, non-1st Normal Form and transitive dependency can be resolved by implementing ORDBMS features.

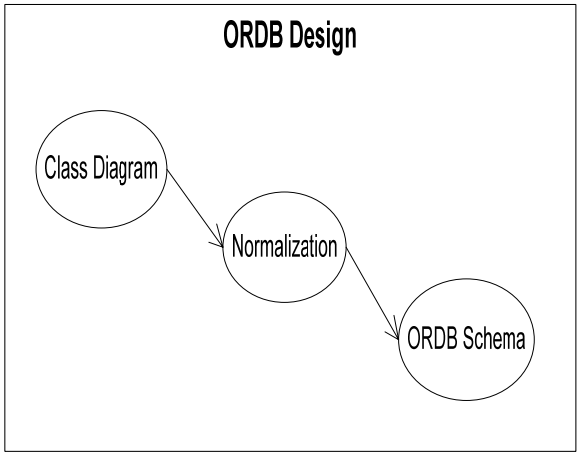

Figure 6 ORDB Design

Transforming a UML class diagram an ORDBM consists of its logical data structure and data behavior. This is not the case with the traditional Entity-Relationship (ER) model. Unlike RDB, object types in ORDB fully support encapsulation, where method definitions can be associated explicitly with object type definitions.

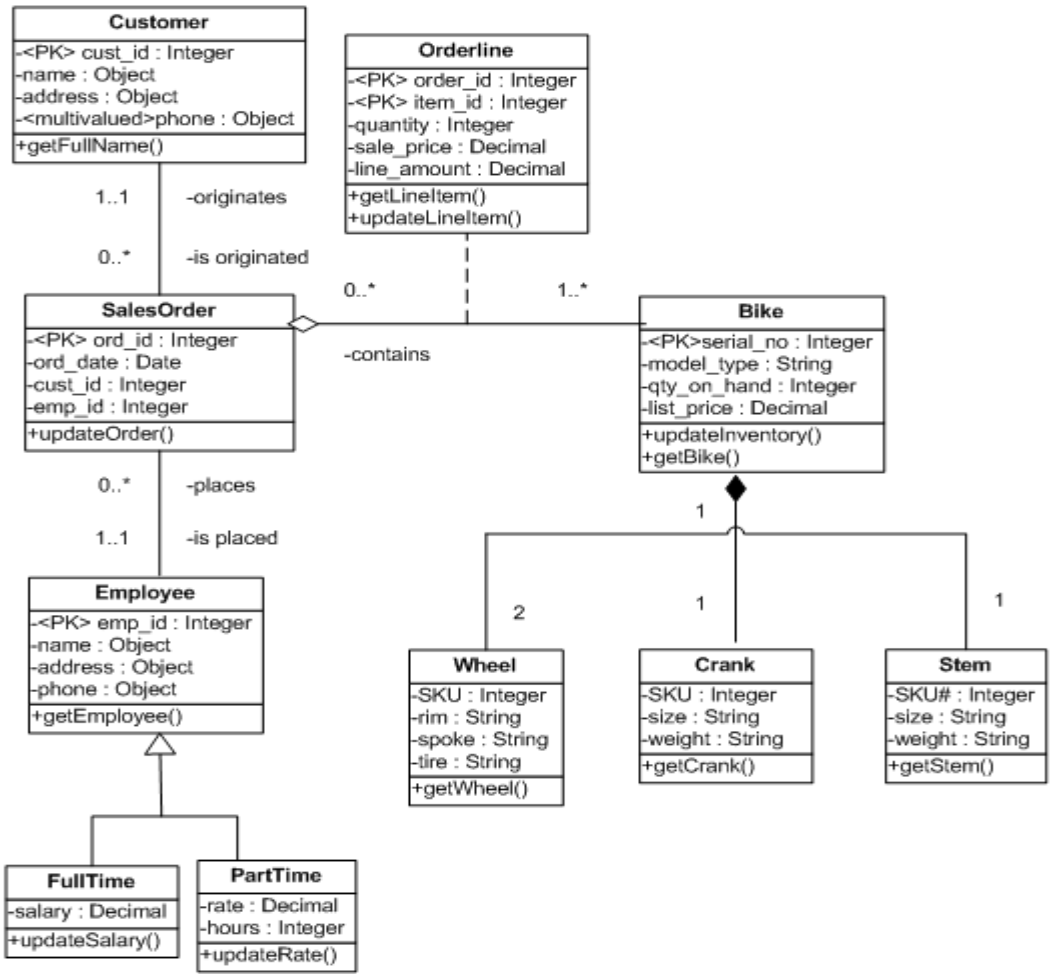

Figure 7 Class diagram for the Bike Order System [3]

Of course, the UML class diagram provides additional advanced concepts that are beyond the ERD scope. Aggregation and composition are association details that are absent from the ERD. In Figure 7, the empty diamond symbol shows the aggregation relationship between Product and Sale Order class. The solid diamond symbol shows a composition relationship between Bike and Wheel, Crank and Stem. A bike is composed with bike parts. Composition models a whole-part relationship where individual parts become elements in a bike class. Aggregation is an association that models a whole-part relationship. Composition is a stronger aggregation association.

Each of the classes is displayed as a rectangle that includes three sections: the top section gives the class name; the middle section displays the attributes of the class; and the last section displays methods that operate on the data in the object. The negative sign indicates private and positive sign indicates public. Associations between classes are indicated with roles (verbs) and multiplicity ("min..max." notation). Inheritance is indicated with an empty triangle.

A sales order is made of line items (bikes). Aggregation is indicated by a small empty diamond next to the SalesOrder class. The dotted line links to the associative class generated from the many-to-many relationship. Aggregation models a whole-part relationship where individual items become elements in a new class.

UML AND ORDBs IMPLEMENTATION

This section discusses how to implement the UML class diagram with object-relational database systems (ORDBMSs). The UML diagrams fully support Oracle ORDB features. Based on the UML class diagram in Figure 7, the following ORDB

features are implemented with Oracle 10g. The sequence diagram transforms its messages to member method headers of an object type. The activity diagram transforms its logic for member method body of an object type. The state machine chart transforms its actions to database triggers in the ORDB systems. The implementation workflow shows how the UML class diagram maps and supports major ORDB features.

- Object data type encapsulation
- Object type Inheritance
- Collection type: VARRAY for multi-value attributes
- Composition using nested table for composition

Object Data Types: Customer Class

Figure 8 shows an object instance transformed to the Oracle ORDB from the Customer class defined in Figure 7.

EMP_ID	SSN	NAME(F_NAME, L_NAME, INITIALS)	DOB	PHONES	ADDRESS(STREET, CITY, STATE, ZIP)	SALARY
1000	123456789	NAME_TY('Jim', 'Fox', 'K ')	12-MAY-60	VARRAY_PHONE_TY('(626)123-5678', '(323)343-2983', '(626)789-1234')	ADDRESS_TY('3 Lost Spring Way', 'Orlando', 'FL', 32145)	45000

Figure 8 Customer table with object types in Oracle 10g

Object Type Inheritance: Employee Classes

ORDBMSs allow users to define hierarchies of data types. With this feature, users can build subtypes in hierarchies of object types (classes) defined in Figure 7. If users create standard data types to use for all employees, then all of the employees in your database will use the same internal format. Users might want to define a FullTime employee object type and have that type inherit existing attributes from Super object type Employee. The FullTime employee can extend super type Employee with attributes to store the FullTtime employee's salary. The PartTime sub type can extend Super Type Employee with attributes to store PartTime employee's hourly rates and wages. Inheritance allows for the reuse of the super type Employee data. Figure 9 shows the result of Oracle's implementation of Employee classes in Figure 7. The full-time employee object instance "Jim Fox" is retrieved from Oracle ORDB.

EMP_ID	SSN	NAME(F_NAME, L_NAME, INITIALS)	DOB	PHONE	ADDRESS(STREET, CITY, STATE, ZIP)	SALARY
1001	123456789	NAME_TY 'Jim', 'Fox', 'K ')	12-MAY-60	VARRAY_PHONE_TY('(626)123-5678', '(323)343-2983', '(626)789-1234')	ADDRESS_TY('3 Spring Way', 'Orlando', 'FL', 32145)	45000

Figure 9 Object Type Inheritance: Employee Classes in Oracle ORDB

VARRAY for Multi-Value Attribute

VARRAY is a collection type in ORDBMSs. A VARRAY consists of a set of objects that have the same predefined data type in an array. In a relational model, multi-valued attributes are not allowed in the first normalization form. The solution to the problem is that each multiple-valued attribute is handled by forming a new table. ORDBMs allow multi-valued attributes to be represented in a database. ORDBMSs allow users to create the varying length array (VARRAY) data type to be used as a new data storage method for multi-valued attributes. Figure 8 shows the result of Oracle's implementation of the Phone object type specified in the Customer class in Figure 7.

Composition: Nested Bike Table

A nested table is a table that can be stored within another table. With a nested table, a collection of multiple columns from one table can be placed into a single column in another table. Nested tables allow user to embed multi-valued attributes into a table. Forming an object Figure 10 shows that using the nested table can implement the composition association in the Bike class of Figure 7.

SERIAL_NO	FRONT_WHEEL (SKU, RIM, SPOKE, TIRE)	REAR_WHEEL (SKU, RIM, SPOKE, TIRE)	CRANK (SKU, CRANK_SIZE, CRANK_WEIGHT)	STEM (SKU, STEM_SIZE, STEM_WEIGHT)
1000	(WHEEL_TYPE ('w7023', '4R500', '32 spokes', '700x26c'))	(WHEEL_TYPE ('w7023', '4R500', '32 spokes','700x26c'))	(CRANK_TYPE('c7023', '30X42X52', '4 pounds'))	(STEM_TYPE('s7023', 'M5254', '2 pounds'))

Figure 10 Bike object displayed in Oracle ORDBMS

CONCLUSION

One of the major criticisms of ORDBs is that its complexity results in the loss of the essential simplicity and purity of the relational database model. Using the visual UML makes the development of ORDBs easier to be comprehended by system analysts and database professionals. The proposed framework indicates that the UML can represent both static and dynamic aspects of ORDBs and the UML class diagram can depict encapsulation and inheritance features of ORBD effectively. Figure 11 summarizes the usage of UML components in the framework of ORDBs development.

Workflows	Required techniques	Additional techniques	Deliverable
Analysis	Use case narratives Use case diagram Sequence diagram	Activity diagram State machine chart	Use case narratives Use case diagram Sequence diagram Activity diagram state machine diagrams
Design	Class diagram 1. Association 2. Aggregation 3. Composition 4. Inheritance 5. Encapsulation	Normalization	ORDB conceptual design Refined Class diagram ORDB schema
Implementation	Mapping and coding class diagram to ORDBMS	Non-1st NF with varray Partial dependency with nested table Object-type within object-type	ORDB with the listed features: User-defined object types VARRAY Nested table Object type Inheritance

Figure 11 Framework of Using UML for ORDBs Development

Using UML for ORDB modeling and implementation for system development teams has great significance in the information system development. An ORDB is an extended relational database and is a vital part of an overall system whose underlying model should encompass all the different perspectives of the system. The UML class diagram graphically depicts actors and objects in the system as an effective and visual communication tool. Behavior components such as a sequence, activity, or state chart diagrams need to be compatible to class diagrams that illustrate the timing of various activities and show how messages are passed between objects on the system. In their UML component usage research, Dobing and Parsons [1] indicate that UML class diagrams were the most frequently used UML component, with 73% of respondents saying they were used in two-third or more of projects.

REFERENCES

1. Dobing, B. and Parsons, J. (2006), How UML is used, Communications of the ACM, 49(5), 109-114
2. Grant, E. S., Chennamaneni R. Reza, H. Towards analyzing UML class diagram models to object-relational database systems transformations, Proceedings of the 24th IASTED international conference on Database and applications 2006, Innsbruck, Austria, 129 - 134
3. Wang, M. 2003. "E-business application development with java technology and oracle: The Fortune Invest Inc. Case", Journal of Information Systems Education, 14(3). 293-299.
4. Wikipedia, Unified Process, http://en.wikipedia.org/wiki/Unified_Process