ISSN 2047-3338

New Approach for UML Based Modeling of Relational Databases

Muhammad Rashid Naeem[1], Weihua Zhu[2] and Adeel Akbar Memon[3]
[1,2,3]Software Engineering College, Chongqing University, Chongqing, P.R. China
[1]rashidnaeem717@yahoo.com

***Abstract*– In this paper, we propose new approach for UML based modeling of relational databases. Furthermore, we use UML Class Diagram and Sequence Diagram to model book data archive and SQL based operations on that data. However this approach is quite in general and can be used for other relational databases.**

***Index Terms* –Class Diagram, DML, Relational Databases, Sequence Diagram and UML**

I. INTRODUCTION

NOWADAYS Object Oriented Approach is considered to be more reliable and efficient approach for developing real world applications (for details see [1]-[3]). Mostly these applications based upon huge amount of data and data manipulations. In the same time it is also preferred to use Relational databases as backend data storage [4]. In case of complex systems, it is very difficult for developers and end users to understand the Relational Database in consistent manner.

This paper is organized as following:

In section 2, we will discuss briefly about background of Relational Databases and Object Oriented Database. Section 3 is about UML Modeling. In section 4, we use our approach for modeling book data archive using UML Diagrams.

II. BACKGROUND OF DATABASES

Databases allow us to store data in the form of tables and relations. These tables and relations ensure consistency of data. It prevents from data duplication by means of unique key or primary key. Databases kept data in a format making it easier to find or retrieve when it is needed [5]-[7]. It also provides us DML (Data Manipulation Language). With the help of databases, we can control access of data and ensure security of data [8]-[10]. Databases allow multiuser access.

Today, there are many advance database tools, which helps us to manage databases like SQL Management studio, SQL Workbench or Oracle Apex etc. Using these tools we can measure the performance, usability, resource usage. We can also save data as distributed storage, located at different places. Database also comes with data recovery in case of data loss which makes is most persistent storage for data users.

A) Rational Database

Relational Database based upon Relational Theory. Relational Theory satisfies relational data model [11]-[13]. According to relational database model relations are tables and these tables consist of rows and columns. We can perform different types of operations on these relational tables. We use SQL to perform operations on these tables. In relational Database we can quickly compare information in the tables because of information divided into different columns. SQL is also a standard for database interoperability. SQL is the base for all database applications available today from Access to Oracle.

B) Object Oriented Databases

Object Technology was first introduced in late 1960s. In 1980s it became more popular because it is more flexible and the way it deals with data, changes the way of building software applications. Instead of relational tables the object oriented techniques defined data in the form of Classes. An object is instance of a Class that contains all information about that Class. Object oriented based on concept of inheritance. It means we can use already available resource instead of creating new ones. Object oriented maps the way as we see the real word. After the birth of Object Oriented methodologies and capabilities of database integration with Object Oriented Programming Language, this led to the development of Object Oriented Databases (OODB) and

Object Oriented Database Management Systems (OODBMS) [14]-[16].

C) Object Oriented vs. Relational Databases and Missing Links

The basic difference between two types of database is classes and tables. Besides this from Fig 1, we can see the difference between two databases models as objects and tuples, attributes and columns, methods and stored procedures etc.

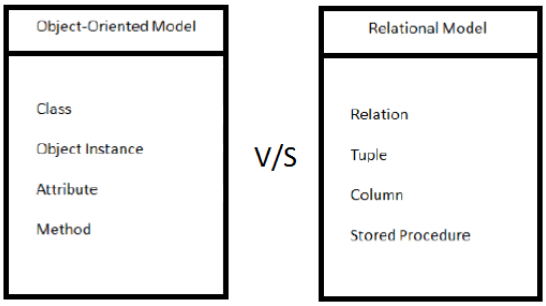

Fig. 1: Comparison b/w Object Oriented & Relational Databases

Object oriented database provides appropriate solution for many kinds of databases. It models the database more closely to the real world. Object Oriented Provide extensions for database types. For example in relational database there are specific datatypes which we have to use to safe data like integer, string etc. In Object Oriented Database Management Systems we can make our own datatypes and Object Oriented Databases can handle complex datatypes like picture, media files etc. Performance is another factor which gives Object Oriented Database advantage over Relational Database. According to many online benchmarks, it is suggested that Object Oriented Databases makes more improvements then Relational Databases. Besides this, programmers like to use Object Oriented Databases because mostly the programming languages they use for developing software applications are based on Object Oriented Paradigm, so it's easy for them to handle Object Oriented Databases.

On Other hand, Relational Database has standards while Object Oriented Database has lack of standards and there is no proper model for Object Oriented Databases. Because of extreme functionality provided by Object Oriented Databases it makes system more complex than Relational Databases. Relational Databases are already in the market. They have more popularity in business applications then Object Oriented Databases. According to survey conducted By DB-Engines.com Relational Database Management Systems Dominates the market [17]. About 77% Business applications are based on Relational Databases. As compared to this, Object Oriented Databases have only 11% popularity. Besides this, Relational Database is better to reduce data redundancy. Object Oriented Databases are not suitable for query based applications and Most of all they lack security.

III. UML MODELING

UML (Unified Modeling Language) is general purpose modeling language that provides graphic notations and diagrams to create Visual Models for Object Oriented Software Systems. UML consists of Use Case, Sequence, State, Activity, Class, Object, Component and Deployment Diagrams. UML Diagrams helps to manage complexity of systems and architectural problems. The difficulty in software development arises when there is need for communication of the desired structure and behavior of a system between analysts, architects, developers, stakeholders and users. UML helps software developers to define a way for handling business processes within software system [18]-[20].

UML does not support modeling for Relational Database and Responsibility Driven Analysis, because Relational Database consists of tuples, columns and relations but not Classes or Objects. Therefore for developers, it is difficult to use relational databases with object oriented software systems but on other hand, using UML extensions, it is possible to model Relational Database and Relational Database Operations i.e. SQL at some extant.

In next section, we are going propose an approach that uses UML Class Diagram and Sequence Diagram to model Relational Database and SQL Operations.

IV. NEW APPROACH TO MODEL RELATIONAL DATABASES

A) UML Class Diagram & Relational Database

Class Diagram is static structure diagram that describes the structure of system by means of Classes, data members and relationship among them [21]. As we know, Relational Databases consists of tables and rows where each row consists of items. If we use Class as a relational table and each object of class as a row where data member or attribute of a class as item within that row and using UML extensions like stereotypes and OCL, it could be a nice way to mapped UML Class Diagram as logical ER-Diagram of Relational Database. Fig. 2 demonstrates the Class Diagram for Books data archive.

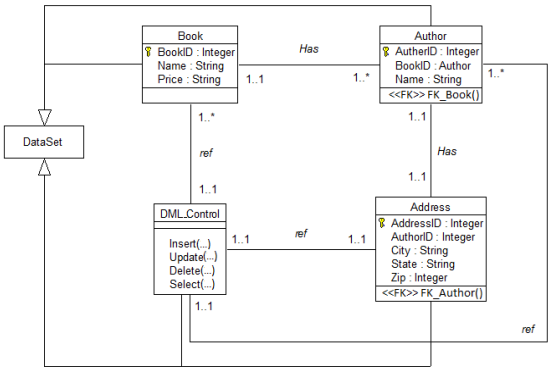

Fig. 2: Class Diagram for Book Data Archive

In this diagram we model three database tables Book, Author and Address where attributes modeled as fields with datatypes integer and String where DML_Control is modeled as SQL operations Handler. Here Dataset works as data storage for all tables in Book data archive.

B) DML Operations & UML Sequence Diagrams

DML (Data Manipulation Language) is used to apply operations on relational databases. DML is a part of SQL (Structured Query Language). DML consists of INSERT, UPDATE, DELETE and SELECT operations. We know that in RDMS, there is no concept of objects and SQL operations are separated from tables but in Object Oriented concepts, information of class encapsulates within object. So we use anonymous object as a row and attributes as item of row in our Model. We are going to use UML Sequence Diagrams to modeling DML operations.

UML Sequence Diagram & INSERT Operation

Sequence diagram are used to model message passing between objects. It is a construct of a Message Sequence Chart [22]-[24]. In our model we use Sequence Diagram to pass commands and expressions between Classes to model SQL operations.

INSERT Operation is use to add data within table so we use going to model following INSERT query. Insert operation is simple so it can be modeled easily

INSERT INTO Book (BookID, Name, Price) VALUES (1, 'xyz', 10);

The Sequence Diagram shown in Fig. 3 illustrates that how we can use Sequence Diagram to model INSERT operation of SQL for Book table. In this Sequence Diagram, DML Control acts as DML operations handler. It demonstrates that DML is referring to the table Book and then uses Book object as anonymous row which adds new data inside Dataset. Because INSERT operation is simple so it can be modeled using Usecase Diagram and Activity Diagram too.

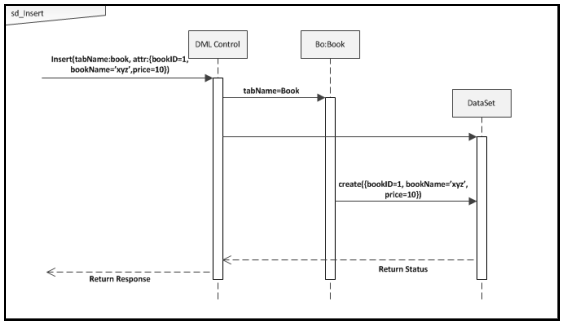

Fig. 3: Sequence Diagram for Insert Query

UML Sequence Diagram & UPDATE Operation

UPDATE Operation looks simple like INSERT but it has WHERE clause which makes it little complex than INSERT Operation. UPDATE Operation of SQL appears as:

UPDATE Book SET Name='abc', Price='20' WHERE BookID=1;

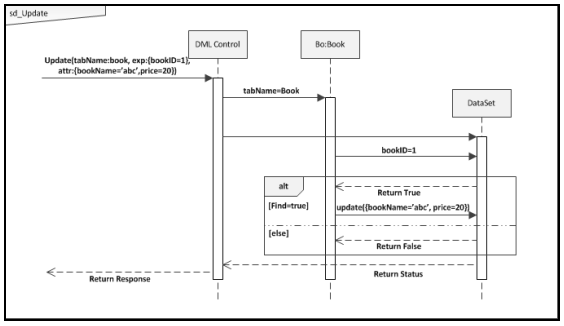

Fig. 4: Sequence Diagram for Update Query

The Sequence Diagram shown in Fig. 4 illustrates that how we can use Sequence Diagram to model UPDATE operation of SQL for Book table. Book object first passes check command to Dataset to find a row. If it finds row inside Dataset, Book object will updates row with new values. Activity diagram can be used also to model UPDATE functionality of SQL.

UML Sequence Diagram & DELETE Operation

DELETE Operation can be modeled similar way as UPDATE Operation but the difference is, in UPDATE Operation we are updating values of row and in DELETE Operation we are removing the row. DELETE Operation of SQL appears as:

DELETE FROM BOOK WHERE BookID=1;

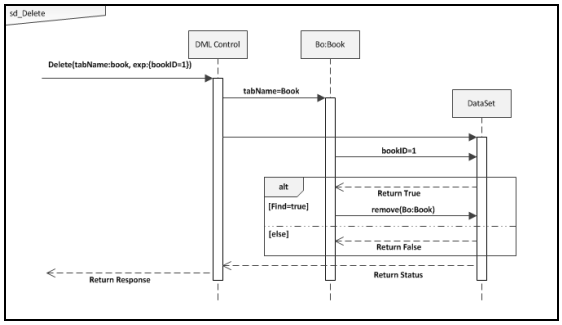

Fig. 5: Sequence Diagram for Delete Query

The Sequence Diagram shown in Fig. 5 illustrates DELETE Operation of SQL. DELETE Operation can also modeled using Activity Diagram.

UML Sequence Diagram & SELECT Operation

SELECT Operation is one of the most important operations of SQL for Relational Database. It can be simple and complex too. It depends on type of SELECT we are using. We will start from simple SELECT to complex SELECT.

UML Sequence Diagram with Simple SELECT Operation:

Simple Select can be written as:

SELECT * FROM Book WHERE BookName='abc'

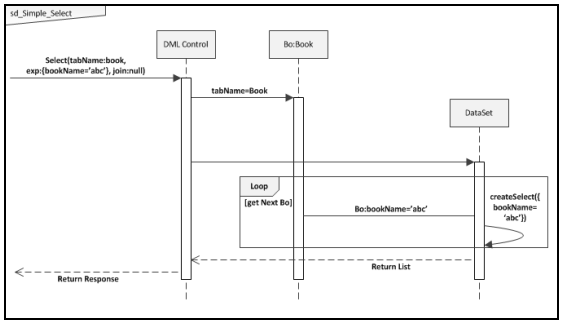

Fig. 6: Sequence Diagram for Simple Select Query

The Sequence Diagram shown in Fig. 6 illustrates how we can model Sequence Diagram for simple SELECT operation. The Loop gets each Book row inside Dataset and creates a list of rows that contains 'abc' as bookName. When loop finishes, it returns list to the DML Control as response.

UML Sequence Diagram & Sub query:

When a query calls another query from WHERE clause than it is called sub query or nested query. We can write sub query as:

SELECT * FROM Book WHERE Price= (SELECT MAX () Price FROM Book);

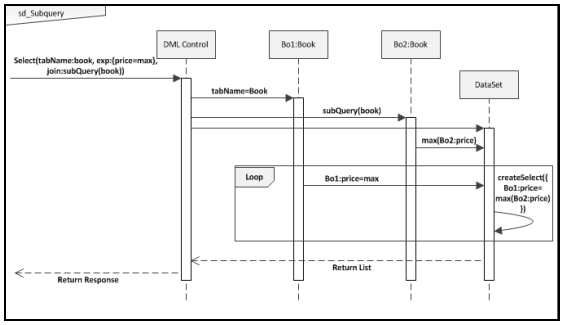

Fig. 7: Sequence Diagram for Select Sub Query

Fig. 7 illustrates how to model sub query using Sequence Diagram. In this Sequence Diagram, Bo1:BOOK acts as first query row and Bo2:Book as second query. Second query first request for max price from Dataset then using Loop Bo1:Book query create list on the basis of max price. Using this technique we can easily model sub query in UML.

UML Sequence Diagram & Joins:

The complexity of SELECT operation more arises when it contain Joins. We use joins when we want to select data by comparing values with other tables.

INNER JOIN Operation:

The INNER JOIN operation also describe as intersection of two tables. We select rows by comparing key values. INNER JOIN of SQL can be written as:

SELECT * FROM Book INNER JOIN Author ON Book.BookID=Author.BookID;

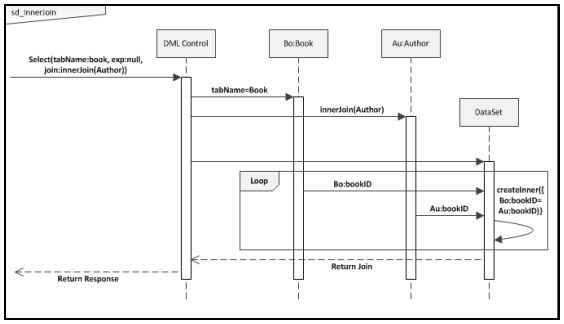

Fig. 8: Sequence Diagram for Inner Join Query

The Sequence Diagram shown in Fig. 8 illustrates how to model INNER JOIN Operations using Sequence Diagram. Inside the Loop Book and Author row fields are being matching in Dataset. Dataset creates list where bookID matched and returns joined list to DML Control.

CROSS JOIN Operation:

CROSS JOIN returns all records after matching each row of first table with each row of other table. Cross Join can be written as:

SELECT * FROM Book CROSS JOIN Author;

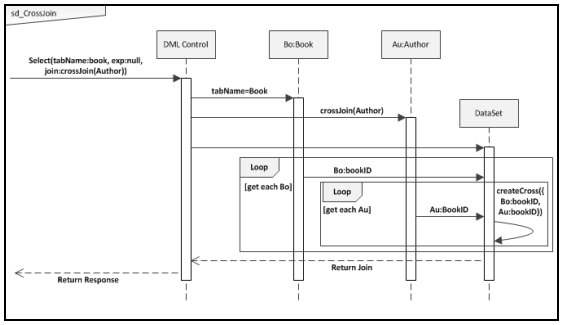

Fig. 9: Sequence Diagram for Cross Join Query

Fig. 9 illustrates how we can model Cross Join Operations using Sequence Diagram. In this Diagram, there are two loops. Outer loop select a row from Book and inner matches it with each row of Author and creating cross join in Dataset list.

OUTER JOIN Operations:

OUTER JOIN returns null values with non-matching row. OUTER JOIN is further divided into three types.

LEFT OUTER JOIN Operation:

LEFT OUTER JOIN Operations returns all rows of left table and matching rows. It displays null with non-matching rows on right table.

SELECT * FROM Book LEFT OUTER JOIN Author ON Book.BookID=Author.BookID;

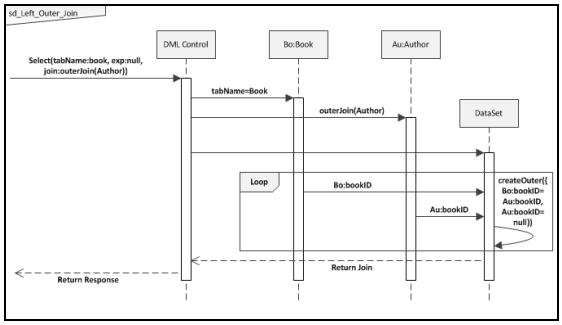

Fig. 10: Sequence Diagram for Left Outer Join Query

In Fig. 10 Sequence Diagram illustrates how to model Sequence Diagram with LEFT OUTER JOIN.

RIGHT OUTER JOIN Operation:

RIGHT OUTER JOIN is opposite of LEFT OUTER JOIN. Instead of showing right table rows as null, it shows left table rows as null. Fig. 11 shows how to model RIGHT OUTER JOIN query with sequence diagram.

SELECT * FROM Book RIGHT OUTER JOIN Author ON Book.BookID=Author.BookID;

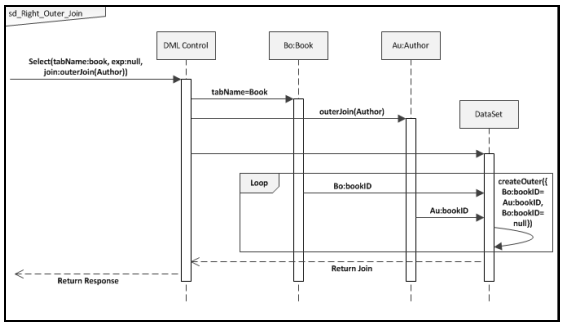

Fig. 11: Sequence Diagram for Right Outer Join Query

FULL OUTER JOIN Operation:

FULL OUTER JOIN shows all rows from left and right tables. It display null both sides with non-matching rows. Full outer join can be model with Sequence diagram as shown in Fig. 12.

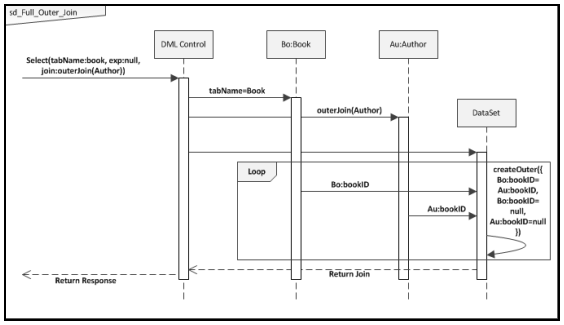

Fig. 12: Sequence Diagram for Full Outer Join Query

V. CONCLUSION

In this paper, we proposed new approach for modeling relational databases by using UML which can be useful for programmers not only for better understanding of relational databases in unified way but also facilitate its integration with Object Oriented Software Systems. Furthermore it is helpful to use UML at database domain level.

REFERENCES

[1] M. Wirsing, A. Knapp, A formal approach to object-oriented software engineering, *Theoretical Computer Science*, Volume 285, Issue 2, 28 August 2002, Pages 519–560.

[2] S. Cook, J. Daniels Designing Object Systems Object-Oriented Modeling With Syntropy Prentice-Hall, New York (1994).

[3] R.A. Kemmerer, Integrating formal methods into the development process, *IEEE Software*, 7 (5) (1990), pp. 37–50.

[4] M. E. Sharkawi, N. A. Tazi, LNV: Relational Database Storage Structure for XML Documents, *Computer Systems and Applications*, 2005. The 3rd ACS/IEEE International Conference, 10.1109/AICCSA. 2005. 1387046.

[5] O. Etzion, B. Dahav, Patterns of self-stabilization in database consistency maintenance, *Data & Knowledge Engineering*, Volume 28, Issue 3, 15 December 1998, Pages 299–319.

[6] S Nissen-Meyer, L König, M Reiser, Maintaining database consistency in an integrated, heterogeneous HIS–RIS–PACS environment, *International Congress Series*, Volume 1256, June 2003, Pages 855–859.

[7] Y. Ishiharaa, S. Shimizub, H.Sekib, M. Itob, Refinements of Complexity Results on Type Consistency for Object-Oriented Databases, *Journal of Computer and System Sciences*, Volume 62, Issue 4, June 2001, Pages 537–564.

[8] H. Rex Hartson, Database security—System architectures, *Information Systems*, Volume 6, Issue 1, 1981, Pages 1–22.

[9] P. Lothian, P. Wenham, Database Security in a Web Environment, *Information Security Technical Report*, Volume 6, Issue 2, 1 June 2001, Pages 12–20.

[10] E. Fernández-Medina, M. Piattini, Designing secure databases, *Information and Software Technology*, Volume 47, Issue 7, 15 May 2005, Pages 463–477

[11] J. M. Medina, O. Pons, M. A. Vila, Gefred: A generalized model of Fuzzy Relational Databases, *Information Sciences*, Volume 76, Issues 1–2, 1994, Pages 87–109.

[12] Didier Dubois, Henri Prade, Semantics of quotient operators in fuzzy relational databases, *Fuzzy Sets and Systems*, Volume 78, Issue 1, 26 February 1996, Pages 89–93.

[13] J.M. Medina, M.A. Vila, J.C. Cubero, O. Pons, Towards the implementation of a generalized fuzzy relational database model, *Fuzzy Sets and Systems*, Volume 75, Issue 3, 10 November 1995, Pages 273–289.

[14] MA Garvey, MS Jackson, Introduction to object-oriented databases, *Information and Software Technology*, Volume 31, Issue 10, December 1989, Pages 521–528.

[15] K. R. Dittrich, Object-oriented database systems: The next miles of the marathon, *Information Systems*, Volume 15, Issue 1, 1990, Pages 161–167.

[16] A. Formicaa, H.D. Grogerb, M. Missikoff, Object-oriented database schema analysis and inheritance processing: A graph-theoretic approach, *Data & Knowledge Engineering*, Volume 24, Issue 2, November 1997, Pages 157–181.

[17] P. Andlinger, RDBMS dominate the database market, but NoSQL systems are catching up, http://www.db-engines.com/en/blog_post/23, 21 November 2013.

[18] K. Lano, Advanced Systems Design with Java, *UML and MDA*, Elsevier Ltd, 2005.

[19] R. Pooley, P. Wilcox, Applying UML, *Advanced Application*, Elsevier Ltd, 2004.

[20] D. Drusinsky, Modeling and Verification Using UML Statecharts, Elsevier Ltd, 2006.

[21] D. Berardia, D. Calvaneseb, G. D. Giacomoa, Reasoning on UML class diagrams, *Artificial Intelligence*, Volume 168, Issues 1–2, October 2005, Pages 70–118.

[22] A. Tsiolakis, Integrating Model Information in UML Sequence Diagrams, Electronic Notes in Theoretical Computer Science, Volume 50, Issue 3, August 2001, Pages 266–274.

[23] A. Refsdal, K. Stølen, Extending UML sequence diagrams to model trust-dependent behavior with the aim to support risk analysis, Science of Computer Programming, Volume 74, Issues 1–2, 1 December 2008, Pages 34–42.

[24] E. Song, S. Yin, I. Ray, Using UML to Model Relational Database Operations, *Computer Standards & Interfaces*, Volume 29, Issue 3, March 2007, Pages 343–354.