An Educational Perspective on Database Management Systems and Object-Oriented Methodology: A 12 Year Journey[*]

Shahram Ghandeharizadeh

Department of Computer Science
University of Southern California
Los Angeles, California 90089

ABSTRACT

Relational database management systems are an essential component of many data intensive applications. At USC, a course entitled "File and Database Management" introduces students to fundamental concepts in relational databases. Students are introduced to conceptual, logical and physical organization of data, use of both formal and commercial query languages, e.g., SQL, indexing techniques for efficient retrieval of data, the concept of a transaction, concurrency control and crash recovery techniques. This paper summarizes our experiences with this course and the challenges of educating students on use of object-oriented concepts and their mapping to tables.

Categories and Subject Descriptors

H.2.0 [**Information Storage And Retrieval**]: General

General Terms

Design, Human Factors

Keywords

Education, Object-Oriented, Database Management

1. INTRODUCTION

This paper is written based on twelve years of teaching an introductory course on database management systems (CSCI 485), an elective course for students majoring in Computer Sciences at the University of Southern California. While its attendance is dominated by undergraduates (typically Juniors and Seniors), there are some first year graduate students who enroll in this course. Throughout the years, the course has been kept up-to-date by changing its assigned projects on a regular basis (approximately every two years). A recent change exercised students' ability to use the C$^{\#}$ [5] programming language to develop the min-kernel of a relational database management system, consisting of a file system, a buffer pool manager, and abstraction of records on disk pages.

One objective of CSCI 485 is *to teach students how to map hierarchical object-oriented representation of data to flat structures, e.g., tables, that reside on a magnetic disk drive.* Typically, students appear to have an informal understanding of this concept prior to enrolling in the course. This is based on their prior experiences, e.g., web servers[1], conversations with peers, etc. They ask questions that demonstrates their curiosity about this concept. These questions range from "Why do Web Servers use relational databases?" to "How can a program store objects in a relational database management system?". It is important to answer these questions effectively at the beginning in order to be able to proceed with other database concepts, e.g., normal forms, transactions and atomicity, etc.

This paper starts with a brief description of object-oriented concepts and how they relate to database management systems. Next, Section 3 describes several fundamental student mis-perceptions and a projects utilized to address these. Section 4 concludes with a brief description of a set of tools that would enhance our educational activities.

2. OBJECT-ORIENTED METHODOLOGY AND DATA MODELS

An object-oriented framework advocates concepts such as abstraction, polymorphism, encapsulation, inheritance, etc. These concepts appear in different data models used to describe a database. These data models are categorized into conceptual, logical, and physical. A conceptual model utilizes tools such as UML [6] and E-R [1] to abstract the real world entities that are represented by a database. The E-R model, for example, consists of entity-sets that are reducible to class definitions in either Java [3] or C$^{\#}$. It includes constructs such as ISA which enables a user to specialize entity sets. This is comparable to inheritance. It is fair to conclude

[*]Our educational activities were supported in part by an unrestricted cash gift from Microsoft Research.

[1]With Hollywood's entertainment industry, management of multimedia content has emerged as a leading industry in Los Angeles. These industries hire students for their projects.

a significant amount of overlap between object-oriented constructs and those of a conceptual data model. (Several textbooks [7, 4, 2] detail the E-R model.) It is important to note the existence of object-oriented data model which extends the E-R model with encapsulation, methods, and a unique object identity.

A logical data model is a mathematical formalism for describing data. The relational data model and its algebra have enjoyed wide popularity during the past few decades. This data model employs a collection of tables to represent both data and their relationship. A physical data model addresses the organization of data across mass storage devices such as a magnetic disk drive. It includes an implementation with the primary objective to provide both (1) the functionality desired by a data intensive application, and (2) high performance. Relational database management systems (RDBMSs) have matured during the past two decades with product offerings from many vendor.

3. COMMON MIS-PERCEPTIONS

The life cycle of a data intensive application typically starts with a high level description of data using E-R model. Next, this description is reduced to its relational equivalent. This step is termed logical data design. This relational description is subsequently stored in a RDBMS (termed physical data design). One reason for using RDBMSs is their high performance for storage and retrieval of data. Typically, the functionality of the application that employs this RDBMS is written using an object-oriented programming language, e.g., C^{++}, Java, $C^{\#}$. This implementation maps rows of tables into one or more instances of class definitions. A common mis-perception is that of conceptual data design, i.e., the high level description, is not important. To the contrary, the conceptual data design is essential because it governs both the organization of data and the class definitions that implement the required functionality. It is interesting to note that products such as Rational Rose and Visio can produce both a relational database schema and its class definition (e.g., in Java) once a system designer provides a high level description of an application. However, how the instances of a class are populated with data is an implementation specific task left to a programmer. This might require the programmer to perform joins of one or more tables in order to construct an object instance required by a specific functionality. Once again, using the high level description of data (E-R), the programmer is able to make clever decisions on how this task is performed.

A second common mis-perception is to duplicate RDBMS functionality in an object-oriented programming language. A good example of this is to compute an average value of a certain property (i.e., attribute, field) across all instance of a class, e.g., average shopping bag checkout value. An RDBMS is capable of processing SQL commands that include aggregates (average, count, sum, min, max). One challenge is how to map an object-oriented aggregate to its relational representation in order to author the necessary SQL query. The presence of a high-level description and how it is mapped to its relational equivalent is an essential component of this step.

We try to tackle these challenges both during the lectures and by employing projects that require students to practice these concepts. Once students are introduced to the E-R diagram, their first homework assignment requires them to derive the E-R diagram of an application. Grading their solution is a difficult task because of the subjective nature of data modeling. Different students interpret the same problem in a different way, resulting in different E-R diagrams. With an in-class discussion of these E-R diagrams, students quickly realize the importance of abstraction. Moreover, they learn the importance of communication during this stage in order to ensure that their abstractions match the data needs of their target application. If the abstraction of an application fails to capture critical data, then the implementation effort with almost certainly be a failure.

Next, students are introduced to the relational data model and SQL. In the second part of their homework assignment, students reduce a class-provided E-R diagram of the first homework to a collection of tables. They populate these tables with synthetic data and are asked to write several applications to produce different set-oriented functionality (including aggregates). Each can be accomplished by writing a SQL command. This shows students how to think in terms of both object-oriented functionality and its mapping to flat tables.

We continue to exercise students' understanding of the object-oriented programming language by requiring them to implement a restricted file system, its buffer pool, and abstraction of records on disk pages. This project is broken into three parts: a) a simple file system using a Windows file with functionality such as CreateFile, DeleteFile, AppendPage, FreePage, etc., b) a buffer pool manager that stages disk pages into frames of a buffer pool, and c) abstraction of records across disk pages. We provide them with sample code that exercises each part of the project extensively. In the past, we have used Java as the required programming language for this effort. More recently, we have switched to $C^{\#}$. It has been fascinating to see almost all students learn an object-oriented programming language constructs on their own in a limited amount of time. I typically introduce them only to object-oriented constructs such as inheritance (and its casting idiosyncrasy), encapsulation, polymorphism and the students learn the rest on their own. Almost all student (95-99%) complete the first part of the project. Parts (b) and (c) are more challenging because they must manipulate data structures effectively while maintaining a one-to-one correspondence between the main memory representation of data and its disk resident image. A lower percentage of students are able to complete parts (b) and (c) – 60-70% is typical.

4. SUMMARY AND FUTURE EDUCATION ENHANCING TOOLS

Teaching students to conceptualize data in object-oriented terms while utilizing tools such as RDBMSs is an important challenge. Projects along with in-class discussions is one approach to address this challenge. Another approach that remains to be explored includes visualization tools that enable students to observe the object-oriented representation of an application's data and how it is mapped into tables (and the organization of these tables across mass storage devices). In order to be effective, these tools must be easy to use and manipulate in real-time. This would enable an instructor to show a conceptual abstraction of a database to students, question the design of this abstraction, change this abstraction and reason about modifications. The visualization should be able to show a mapping to a relational representation and the organization of data across the platters of one or more disks. This would help students understand the importance of physical data models and how an abstraction impacts both system functionality and performance. With rapid technological advances, this vision might be realized in the near future.

5. REFERENCES

[1] P. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[2] C. Davis, S. Jajodia, P. Ng, and R. Yeh. *Entity-Relationship Approach to Software Engineering.* North Holland, 1983.

[3] H. Deitel and P. J. Deitel. *Java How to Program, Fifth Edition.* Prantice Hall, 5th Edition, ISBN: 0131016210, 2002.

[4] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* Benjamin Cummings, 2000.

[5] C. Petzold. *Prgoramming Windows with C# (Core Reference).* Microsoft Press, ISBN: 0735613702, 2001.

[6] C. Kobryn. UML 2001: A Standardization Odyssey. *Communications of the ACM*, 42(10), October 1999.

[7] B. Thalheim. *Entity-Relationship Modeling: Foundation of Database Technology.* Springer Verlag, 2000.