

Enhancing Student Learning In Database Courses With Large Data Sets

Venkat N Gudivada¹, Jagadeesh Nandigam², and Yonglei Tao³

Abstract - Rapidly increasing storage device capacities at ever decreasing costs have resulted in mushrooming of publicly available large data sets on the Web. In this paper, we describe a novel approach to teaching relational database course by using such data repositories. We demonstrate our approach using the Amazon.com product database, though the approach is generic and is applicable to other data repositories. The Amazon database is supposedly the largest product database ever in existence. We have used the Amazon Web Services API and .NET/C# application to extract a subset of the product database to enhance student learning in a relational database course. This realistic data served various activities of the course and provided a rich backdrop to demonstrate more interesting features of SQL and Oracle cost-based query optimization. Central to the course is a semester-long team project. We discuss the details of data extraction from Amazon.com, conceptual and logical data modeling, logical and physical database design, database creation and data loading, database querying, and database application development.

Index Terms - Amazon e-commerce service, Enhancing student learning, Large data sets, Relational database course.

INTRODUCTION

Majority of the software applications used in the commercial world today are database-driven. They critically depend on fairly large relational databases. Recently, this trend has been further accelerated due to ubiquitous leveraging of data mining techniques to enhance business profitability. Large databases are interesting and challenging from both intellectual and practical perspective. Issues related to the complexity of the domain, missing and duplicate identifiers, logically inconsistent data, unnamed and badly named data, performance and scalability pose challenges. Therefore, computer science graduates are expected to possess the requisite practical skills beyond the relational database theory so that they are prepared to design and develop large and practical databases. One can leverage the publicly available large data sets and complex programs on the Web to impart practical skills to the next generation of software practitioners [3, 14, 17, 18].

The database courses in the academia have yet to leverage large data sets in imparting practical skills to the students.

Almost all the database textbooks used for teaching relational database courses employ simple and rather trivial databases consisting of at most a handful of tables. Though this approach has merit in that the database structure is simple for the students to understand, however, it has serious shortcomings. This approach lacks interesting and complex database structure to demonstrate practical aspects of data modeling, SQL features, and the interplay between the database design and query performance.

In this paper, we describe a novel approach to enhancing student learning in a relational database course by leveraging a subset of the product data of Amazon.com. The data is made available to the public via Amazon E-Commerce Service (ECS). We discuss the details of data extraction from Amazon.com, conceptual data modeling, logical and physical database design, database creation and data loading, database querying, database application development, assessment of students' learning, and the lessons learned.

The remainder of the paper is structured as follows. First, we briefly describe a few publicly available large data sets on the Web. The process that we have used to extract a subset of the product data from Amazon.com is described next. Our approach to enhancing student learning in a relational database course using Amazon data is described next. The last section concludes the paper.

LARGE DATA SETS ON THE WEB

Tremendous improvements in secondary storage device capacities and costs have resulted in the mushrooming of a number of publicly available large data sets on the Web. Examples of such data sets include:

- Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system provides access to financial information filed by publicly-held companies in the United States [13].
- Microarray Expression repository is a database of genetic and molecular biology data for the model higher plant *Arabidopsis thaliana* [15].
- Databases of characteristics of each potential gene in the *M. Tuberculosis* [10] and *E. Coli* bacteria [9].
- PROSITE is a database which describes protein domains, families and functional sites, as well as associated patterns and profiles to identify them [6].

¹ Venkat N Gudivada, Engineering and Computer Science, Marshall University, Huntington, WV 25755, gudivada@marshall.edu

² Jagadeesh Nandigam, School of Computing and Information Systems, Grand Valley State University, Allendale, MI 49401, nandigaj@gvsu.edu

³ Yonglei Tao, School of Computing and Information Systems, Grand Valley State University, Allendale, MI 49401, taoy@gvsu.edu

- UniProtKB/Swiss-Prot database is a curated protein sequence database which provides a high level of annotation [16].
- Protein information resource (PIR) is a single, centralized, authoritative resource for protein sequences and functional information [12].
- RCSB Protein Data Bank (PDB) is an information portal to biological macromolecular structures [8].
- Sun-Earth-Connections mission data hosted by NASA Coordinated Data Analysis Web (CDAWeb) system [2].
- The National Long Term Care Survey (NLTC) data [4].
- Movies data [5].

This list is by no means exhaustive. Some domain knowledge is essential to fully leverage the data sets. Moreover, the reader should recognize that these sites in future may change their policies related to the amount of data and the type of data that is made available to the public for free.

AMAZON E-COMMERCE SERVICE

Amazon Web Services (AWS) is a framework for providing access to Amazon.com's various resources. More specifically, AWS is a set of services which enables the creation of web sites and applications that use or incorporate Amazon's e-commerce or other externalized software or data. For example, Amazon E-Commerce Service (Amazon ECS) exposes Amazon's product data and e-commerce functionality.

Product data can be retrieved from ECS using two types of requests: SOAP and REST. The first type, Simple Object Access Protocol (SOAP), is used to request data programmatically. To use the SOAP method, one needs to examine ECS Web Services Description Language (WSDL) document – a standard (XML) format for specifying a Web service request – and appropriately specify a product data request. On the other hand, Representational State Transfer (REST) type request is initiated from a Web browser. An URL extended with key-value pairs is used to retrieve and display product data in a Web browser. REST requests are typically used for initial exploration of the product data.

Amazon ECS supports two types of product retrieval operations: *search* and *lookup* [1]. A search is a request that returns information matching the specified criteria. A lookup, on the other hand, is a request for a specific object or set of objects, specified by a unique identifier(s).

Search for products begins with the specification of *locale* – a country or geographic region. This is necessary since all products are not available for sale in every geographic region. Amazon product data is partitioned based on *search indexes*. The latter can be thought of as high-level classes such as Books, Music, DVD, Software, and Office Products. There are about thirty-five search indexes. Search indexes are structured as hierarchically organized *categories*. For example, Computers & Internet is a category under the search index Books. Descendants of Computers & Internet category include subcategories such as Software Development, and Computer Programming. The latter may have its own immediate

descendant categories such as Java, C++, and Python. Though this is hierarchical taxonomy, often books are listed under multiple categories.

One can control how much and what kinds of data are returned in a response by specifying the response group parameters in a request to ECS. A SOAP request may indicate values for one or more *response groups*. For example, you can provide the values *large*, *images*, *offers*, and *reviews* for response groups to request additional details about books including cover images, available offers, and reviews. There are about 25 response groups. Not all response groups are applicable to every product.

The response to an ECS request is always returned as an XML-formatted stream of text. The results are delivered one page at a time. A page contains information about ten books. There should be at least one second elapsed time between successive requests. All ECS requests need to contain an access key, which can be obtained from Amazon.com upon free registration.

TEACHING DATABASE COURSE WITH AMAZON.COM DATA

We developed a .NET/C# application to make SOAP requests to Amazon ECS. This application also parses the results to generate a delimited flat file. Each line in this flat file contains various pieces of information about a book. Fields are delimited by % character, and multiple values of a field (for example, authors) are delimited by colon (:) character. The application was made available to the students. One student team focused on the Computers & Internet category of Books search index, and retrieved over 30,000 books. This was done incrementally by first retrieving the books whose author's last name starts with *A*, followed by books whose author's last name starts with *B*, and so on.

We have used Amazon ECS data to enhance student learning in a relational database course. Central to the database course is a team project spanning a four-month period. Typically, two to three students comprise a team. Four-student teams are permitted only under exceptional situations. Though majority of the teams selected a subset of the Amazon product database for their project, some teams selected projects based on other data repositories. Teams progressed through the various stages of the project in sync with the discussion of the corresponding topics in the classroom lectures. The course topics progressed from conceptual data modeling, logical database design, database normalization, physical database design, writing SQL queries, triggers and database stored procedures, tuning SQL queries, to database application development.

I. Developing a conceptual data model

The following are some of the attributes of books retrieved from Amazon: ASIN (Amazon Standard Identification Number, which is actually a 10-digit ISBN); EAN, an extended ISBN (13 digits); title; author(s) names and author description; language; publisher; edition; number of pages; publication date; list price; Amazon price; sales rank; media type (hardcover, paperback, textbook binding, unknown

binding, digital, CD-ROM, and DVD); UPC; dimensions (height, length, and height); weight; availability; images of front and back covers; categories (a book may belong to multiple categories); subjects (or keywords, several subjects may be assigned to a book); similar books (ASINs of related books); and reviews.

There are different types of reviews for a book: customer reviews, spotlight reviews, and editorial reviews. For each customer review, we retrieved review date, review summary, review text, contact info of the reviewer, name/id of the person providing the review, total number of people who voted for the review, and how many people voted favorably. Spotlight reviews are those customer reviews which have been voted favorably by a large number of people. Top two customer reviews according to the above business rule are shown as spotlight reviews. Editorial reviews, on the other hand, have only two pieces of information: who provided the review (Amazon.com, From the Preface, Publishers Weekly, etc.), and review text.

We have used the DB Designer 4 (<http://fabforce.net/>), an open source data modeling tool, for developing the conceptual data model. If you use Eclipse for application development, there are several tools available for conceptual data modeling as Eclipse plug-ins at <http://www.eclipseplugincentral.com>. There are a few issues that need resolution before the conceptual data model construction can begin. First, naming conventions need to be established for entities, attributes, relationships, and constraints. These naming conventions are quite essential in constructing practical databases. In enterprise database systems, number of database tables in the range of 1000 to 1500 tables is quite common. Some data modeling tools allow you to specify two names: conceptual or business name, and a technical name. In the conceptual data modeling stage, business names are used in communicating with the business analysts and domain experts. Technical names are used in the logical and physical database design phases, and to communicate with the database administrators. Technical names are constrained by the restrictions of the target database systems (e.g., Oracle, DB/2). Another important task is to establish domains for the attributes.

We have used acronyms for the table and attribute names consistently. The underscore (_) character is used as a visual separator between acronyms. For example, the central entity Book is named as BOOK. The entity that stores book identifiers such as ASIN and EAN is named BOOK_ID. As another example, attribute *price* is abbreviated as PRC and is used consistently throughout.

Conceptual data modeling aspect of the project was the most interesting and challenging one. It is an iterative process. Instructors' involvement in this phase is very critical since the data model is of substantial size in terms of entities, attributes, and relationships. Student teams typically begin conceptual data modeling by placing most of the attributes as attributes of a central entity, the Book. However, as they evolve the data model, they realize that Author is another full-fledged entity, as well as Review, Publisher, Book Identifier, Book Category, Book Subject, and Book Image. Students recognize that the

above is stemming from two different considerations. First is the conformance to the first normal form of the relational data model – attributes are atomic values. The second is related to performance issues. Some teams started off with making an assumption that no book will have more than five authors, and added Author1, Author2, ..., Author5 as attributes of the Book entity. As they looked at additional data from Amazon, it was immediately evident to them that such an assumption is not appropriate. This observation drove home the point that they need to consult business analysts in validating their assumptions.

Though the first normal form requirement of the relational data model was relatively easier for students to incorporate into the conceptual data modeling, designing the conceptual model to address performance issues is not easy. It requires looking ahead and practical experience. However, some may argue that conceptual data modeling should be free from performance considerations. Changing the data model after the applications have been built is an expensive process. The following exemplifies another situation where the instructor needs to help the teams by injecting his insight and experience. Migrating the Author1, Author2, ..., Author5 attributes into an entity of their own (called Author) entails several benefits. The central entity Book becomes more *agile* to address the performance requirements. When a book search is initiated based on author's name, applications are dealing with a smaller table – Author. This approach also permits adding an attribute to the Author table to store author's biography without performance penalty.

There are several other situations in the conceptual modeling process that provided right contexts to illustrate ways to address performance issues early on. One example is the need to provide two entities rather than one to store review information. Another example is to store book images in a separate entity rather than storing them in the BOOK entity. A third example is the need for surrogate keys for entities rather than using their natural identifiers. For example, for the Author entity, an internally generated object identifier (OID) is better suited as a (surrogate) key than the natural identifier author name. There is also the need to create additional entities to store data which is not part of the flat file extract from Amazon. For example, entities are needed to support *suggestive selling* and order processing applications that are yet to be designed and developed. For space reasons, we don't further elaborate the effectiveness of the iterative evolution of the data model, and the rewards of thinking ahead.

II. Logical database design

Conceptual database design is independent of any issues related to the database system that will be used to implement the conceptual data model. The logical data model is a natural progression in this step and aims at designing a set of relational database tables that are free from insertion, deletion, and update anomalies. The course instructors play the role of a business analyst (BA) and a database administrator (DBA) for the student team projects. The first step in the logical database design is to insure that we have captured all the entities and

their attributes to support the business requirements. The BA provided a list of business requirements in the form of use cases for building an online boutique specializing in technical books. Student teams provided a walk-through of the data model to the BA. The primary goal of this walk-through is to insure that the conceptual data model is complete (has the necessary data elements to support all business processes) and consistent (in naming conventions).

DB Designer 4 has an option to produce a report of the conceptual data model in HTML format. This report served as version zero of the logical data model. Functional dependencies between the data elements in each entity/table are collaboratively determined by the teams and the instructor. This is again an iterative process. Next, the students applied the rules of database normalization on the set of tables in the logical data model. They have also used a software tool that runs in SWI-Prolog environment to verify that the candidate keys chosen for the tables don't entail violation of first three normal forms. This tool is discussed in the lab manual that accompanies a database textbook [7]. Somewhat surprisingly, most of the tables didn't need any changes. Upon reflection, it appears that this situation is a direct consequence of the due diligence applied in the conceptual database design phase.

III. Physical database design

Physical database design turned out to be much harder for the students because it requires a detailed knowledge of file storage structures and access paths available in the Oracle 10g database system. Furthermore, in some situations, an access path which is beneficial for one query or transaction turns out to be a curse for another query. Moreover, as the types of database queries and transactions change or evolve over a period of time, the suitability of existing file storage structures and access paths needs to be assessed on a periodic basis. Students reflected on the above considerations before performing the physical database design. As a rule of thumb, physical database design started off with a primary index for the attributes that comprise table keys. Secondary indexes were also created for attributes that are required to have unique values in a table.

IV. Creating and populating the database

Database creation scripts were generated from the logical database design. The script also included specification of various types of constraints: not null, primary key, referential integrity, and check. Each team created their database by running the script. A Web-based tool, iSQL*Plus, was used for this purpose.

Database population was the most tedious and time consuming task. Referential integrity constraints were disabled before loading the data so that the data can be loaded into tables in any order. Data was loaded using a Oracle tool named SQL*Loader. This required creating several flat files (one for each table in the logical database) by extracting the appropriate fields from the large flat file generated by the .NET/C# application that interfaced with Amazon ECS.

Students have developed programs for this task using the languages such as VB.Net, and Python.

The log file from the SQL*Loader run was examined to see if any of the records in the flat files didn't go through. To the students' surprise a significant portion of the books had null values for several attributes/columns. This required reevaluating the previously specified *not null* constraints, altering the logical database schema as needed, dropping the existing tables in the database, recreating the database tables, and loading the data. Next, referential integrity constraints were turned on. Here again, students noticed a significant number of referential integrity violations. They have to devote substantial amount of time in resolving these violations. In summary, it was a revealing experience for the students that the real data can be incomplete and inconsistent.

V. Writing queries and application development

This phase of the project focused on writing database triggers, stored procedures, queries, and transactions. For example, the business logic to promote *suggestive selling* via *better together* feature found on Amazon.com Web site is an ideal candidate for implementation as database triggers. Likewise, the logic for the selection of two reviews as spot reviews from a set of customer reviews is suitable for implementation as a database stored procedure. Triggers and stored procedures were implemented in both PL/SQL (Oracle procedural extension to SQL) and Java.

Several queries were written and packaged as stored procedures to support an application for browsing and searching the book database. Transactions were also developed to implement order processing. Student teams developed these applications using JDBC, Servlets, and JSP. By engaging in the above activities, students clearly conceptualized and internalized distinctions among triggers, stored procedures, queries, and transactions. This also exposed them to a three-tier application architecture which is typically used for developing today's data-intensive Web applications.

VI. Tuning SQL queries and physical database design

Oracle uses cost based optimizer (CBO) to generate accurate execution plans for queries. CBO requires statistics about data storage and distribution. The `gather_schema_stats()` method located in Oracle `dbms_stats` package is run to generate these statistics. Oracle uses histograms to store detailed information about non-uniform data distributions. This information helps the CBO better estimate the selectivity of predicates which will result in more efficient execution plans.

Once the statistics are generated, students use the *explain plan* feature of Oracle to understand query execution plan. This enables them to clearly see where the execution plan uses *table* and *index scans*. They time query execution times with and without indexes on database tables. Depending on the number of rows in the database tables, the time it takes to execute a query with and without indexes can be quite substantial.

Students also experimented with a feature of Oracle 10g called *SQL Tuning Advisor*. Oracle 10g allows the CBO to run

in *tuning mode*, where it can gather additional information and make recommendations about how SQL statements can be tuned further. The optimizer performs statistics analysis, SQL profiling, access path analysis, and SQL structure analysis to recommend how the query execution time can be improved.

CONCLUSIONS

The database course is primarily taken by senior computer science majors. They are already well versed in two object-oriented programming languages (Java and C++). Furthermore, they have taken the following math courses: discrete mathematics, two-semester calculus sequence for engineering majors, and linear algebra. This background made the challenge of implementing the approach described in this paper in a one-semester course more manageable.

Students' learning was measured based on a few assessments which we have specifically designed for the database course (based on Bloom's taxonomy). The course is also assessed based on an instrument which is used across all course offerings at the university. Overall, the students' response was very positive (relative to the same course offering using the traditional textbook format in the past). More importantly, the proposed approach provided a realistic situation to experience the difficulty of building a conceptual data model, dealing with inconsistent and incomplete data, whether or not to normalize a database design, and how to deal with performance issues. We plan on undertaking a more formal approach to assess student learning after making the following enhancements.

We plan on incorporating into the course more practice with writing hierarchical queries to explore the category hierarchy. We will also extract more data about books from Amazon (e.g., books that a given book cites, and other books that cite the given book), as well as data about other product types (e.g., electronics, toys, music). We expect the resulting database to serve other courses such as software engineering, data warehousing, data mining, and machine learning.

REFERENCES

- [1] Amazon.com. Amazon E-Commerce Service: Developer Guide. <http://developer.amazonwebservices.com>. Accessed 13 March 2007, 2007.
- [2] Coordinated Data Analysis (Workshop) Web (CDAWeb). A web-based system for viewing essentially any data produced in common data format (CDF) by sun-earth-connections mission. Data is available for anonymous ftp. http://cdaweb.gsfc.nasa.gov/istp_public/. Accessed 14 March 2007, 2007.
- [3] Cohen, J. Updating computer science education. *Commun. ACM*, 48(6):29–31, 2005.
- [4] The National Long Term Care Survey data. Data related to health and functional status of americans aged 65 and above. It also includes health expenditures, medicare service use, and the availability of personal, family, and community resources for caregiving. <http://www.nlctcs.aas.duke.edu/index.htm>. Accessed 14 March 2007, 2007.
- [5] Movies database. Data about 10,000 films, which includes information on actors, casts, directors, producers, studios, etc. <http://kdd.ics.uci.edu/databases/movies/movies.html>. Accessed 14 March 2007, 2007.
- [6] Protein database. Describes domains, families and functional sites, as well as associated patterns and profiles to identify proteins. <http://expasy.org/prosite/>. Accessed 14 March 2007, 2007.
- [7] Elmasri, R, and Navathe, S. *Fundamentals of Database Systems*. Addison-Wesley, 5th edition, 2007.
- [8] Research Collaboratory for Structural Bioinformatics (RCSB) Protein Data Bank (PDB). Provides a single, centralized, authoritative resource for protein sequences and functional information. <http://www.rcsb.org/pdb/home/home.do>. Accessed 14 March 2007, 2007.
- [9] E. Coli genes. This database provides sequence, homology, and structural information, and function (if known) of e. coli genes. <http://kdd.ics.uci.edu/databases/ecoli/ecoli.html>. Accessed 14 March 2007, 2007.
- [10] M. Tuberculosis genes. This database provides sequence, homology, and structural information, and function (if known) of m. tuberculosis genes. <http://kdd.ics.uci.edu/databases/tb/tb.html>. Accessed 14 March 2007, 2007.
- [11] The Health and Retirement Study data. Data about physical and mental health, insurance coverage, financial status, family support systems, labor market status, and retirement planning data about aging Americans. <http://hrsonline.isr.umich.edu/>. Accessed 14 March 2007, 2007.
- [12] Protein Information Resource (PIR). A single, centralized, authoritative resource for protein sequences and functional information. <http://pir.georgetown.edu/>. Accessed 14 March 2007, 2007.
- [13] U.S. Securities and Exchange Commission. Electronic Data Gathering, Analysis, and Retrieval (EDGAR) database system. <http://www.sec.gov/edgar.shtml>. Accessed 13 March 2007, 2007.
- [14] Stevenson, D, E, and Wagner, P, J. Developing real-world programming assignments for CS1. In *ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education*, pages 158-162, New York, NY, USA, 2006. ACM Press.
- [15] The Arabidopsis Information Resource (TAIR). Microarray Expression database for the plant Arabidopsis. <http://www.arabidopsis.org/index.jsp>. Accessed 13 March 2007, 2007.
- [16] The universal protein resource. A curated protein sequence database which provides a high level of annotation. <http://expasy.org/sprot/>. Accessed 14 March 2007, 2007.
- [17] Wagner, P, J. Teaching data modeling: process and patterns. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 168-172, New York, NY, USA, 2005. ACM Press.
- [18] Wick, M, R, and Wagner, P, J. Using market basket analysis to integrate and motivate topics in discrete structures. In *SIGCSE'06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 323-327, New York, NY, USA, 2006. ACM Press.