History Places: A Case Study for Relational Database and Information Retrieval System Design

DAVID G. HENDRY
University of Washington

This article presents a project-based case study that was developed for students with diverse backgrounds and varied inclinations for engaging technical topics. The project, called History Places, requires that student teams develop a vision for a kind of digital library, propose a conceptual model, and use the model to derive a logical model and information retrieval specification. From these two design representations, students implement a data-driven Web site that enables users to browse content and search by exact and best-match queries. The project brief contains a set of general requirements that promote creative solutions, while also bounding the complexity of the solution space. The article includes teaching notes and a conceptual model, expressed as an enhanced entity-relationship model in UML. The model, consisting of approximately ten entities, contains binary, unary, ternary, and specialization/generalization relationships. The article concludes with some reflections based on the experiences of using this project in six classes over four years.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design; H.3.5 [**Information Storage and Retrieval**]: Online Information Services; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries; K.3.2 [**Computers and Education**]: Computer and Information Science Education

General Terms: Design, Documentation

Additional Key Words and Phrases: Case-based learning, design, lucene, postgresql, database management systems, information retrieval, informatics, education, UML, conceptual modeling

ACM Reference Format:

Hendry, D. G. 2007. History Places: A case study for relational database and information retrieval system design. ACM J. Educ. Resour. Comput. 7, 1, Article 3 (March 2007), 20 pages. DOI = 10.1145/1227846.1227849 http://doi.acm.org/10.1145.1227846.1227849

1. INTRODUCTION

Student projects for engaging information system design often concern business domains where the goods or services of physical enterprises are modeled

Author's address: D. G. Hendry, The Information School, University of Washington, Box 352840, Seattle, WA 98195-2840; email: dhendry@u.washington.edu

with the aim of improving operational effectiveness and decision support. Traditional examples include scheduling and reservation services, inventory control, and order processing. These are common, operationally rich, and economically important domains. In recent years, interesting case studies have been published in journals and proceedings (e.g., Cappel [2002], Myers [2003], Parker [2003], Chau et al. [2003], and Wedel et al. [2004]) and, of course, are found in textbooks (e.g. Date [1995], Belew [2001], Connolly and Begg [2005], and Kifer et al. [2006]). With pedagogically-informed simplifications, these are ideal domains for learning conceptual, logical, and physical database design and for learning to implement information systems with the relational model. They are domains, however, that are particularly suited to students in business programs because the focus is on modeling the enterprise, or to students in computer science because these domains all engage common issues associated with modeling, data structures and algorithms, operational performance and scale, and so on. The business transaction is the key unit of analysis which leads to system requirements that entail common technical issues and solutions.

History Places, on the other hand, is a project-based case that requires students to implement a working system that engages a different domain. Rather than asking students to model aspects of an enterprise, History Places asks students to model the information structure of collections of media items, such as photographs and digital recordings, and to enable the operation of user-interface elements for interacting with media (e.g., browsing directories, search by attribute, search by free-text, etc.). The key unit of analysis is the information task, that is, the tasks by which people contribute and discover media items. History Places has been designed for undergraduate students of Information Science who study systems from human, technological, and information orientations. It motivates students to engage technological topics in relational databases, information retrieval, and Web technology.

The root concept for History Places is that recording and exploring the history of a place is an intrinsically rewarding activity. This, of course, begs the question, "What is a place", an interesting philosophical question [Smith et al. 1998]. In fact, it is this very question that must be answered by students through their design and implementation work. In any case, the aim of History Places is to enable people to contribute, organize, and discover representations of places that vary over time. Weblogs [Nardi 2004], Wikis [WikiSym 2006], portals [Smith 2004], and collaborative information collections [Hendry et al. 2006; Hendry and Carlyle 2006] are fashionable examples of systems that are similar in spirit to History Places.

History Places is a kind of small-scale, grassroots *digital library*—a term with contentious and evolving meanings [Levy and Marshall 1995; Goncalves et al. 2004]. It is a digital library in the sense that History Places allows people to select, describe, and organize items which, in turn, can be browsed and searched. In turn, a group of people with common interests can collectively participate in these and similar information tasks. History Places is neither a platform for building digital libraries nor, of course, is it intended to be used as an institutional repository.

The root concept for History Places can be elaborated with four general activities related to the handling and access of media in which people 1) upload media, 2) browse by category or time, 3) search by exact-match queries, and 4) search by best-match queries. These general activities, in turn, can be further decomposed into many different sets of functional requirements, and, as we shall see, this framework can be reified to support the development of many different kinds of applications.

With this framework, History Places seeks to strike a balance. On the one hand, in order to constrain the technological problems and focus on specific learning objectives, it seeks to prescribe the functional requirements to be implemented. On the other hand, it seeks to give students latitude for creatively exploring the problem space, identifying the key problems and developing specific solutions. History Places, in short, is a project-based case that requires students to find a problem, describe it, and create a solution for it but within a carefully bounded problem space.

2. LEARNING ENVIRONMENT

2.1 The Information School Perspective

The History Places project has been designed for students earning a B.S. in Informatics at The Information School, University of Washington. This 2-year degree program accepts students in their junior year. Students take courses in technical and human-centered topics. All courses are firmly grounded in an information perspective where students study the cognitive, social, design, and technical dimensions of information systems.

This information perspective, with its roots in librarianship and information science, focuses on:

(1) theories and practices of information collection and organization;
(2) social factors that influence information flow within groups, organizations, and societies;
(3) cognitive factors that influence how people search, browse, discover, and make sense of information;
(4) human-centered design methods for identifying needs and designing usable, useful, and enjoyable systems.

Students also take technologically-oriented courses in networking, information system analysis, information visualization, and database management and information retrieval. To enter the program, students must successfully complete a sequence of three programming courses in Java: Computer Programming 1 and 2, and Data Structures and Algorithms. Some students look at these courses as perfunctory: they must be passed in order to focus on such topics as user-centered design, information management, and so on. Other students, in contrast, are strongly motivated by technology and come to the courses with extensive outside technical experience, especially in Web development, and they are highly motivated to learn technological topics and build functioning systems. In short, students have

diverse technical backgrounds and inclinations to engage with technological topics.

A second challenge is coverage. Given a curriculum with a single required course on information storage and access, what should be covered? A decision was made to design a course, Database Management and Information Retrieval, which covers the foundations of the relational model and best-match information retrieval model. In this 10-week course, approximately 6 weeks is devoted to database management and 3 weeks to information retrieval. For database management topics, the course uses Connolly & Begg [2004, chap. 1–14] and for information retrieval topics, the course uses Belew [2001, chap. 1–3] and two supplemental papers, one on the general architecture of the Google search engine [Dean et al. 2003] and one on the evaluation of IR systems [Baeza-Yates and Ribeiro 1999]. Covering both of these topics in a core course requires disciplined prioritization of the topics covered and careful attention to how the transition between the two areas of study is managed.

The course consists of 2 weekly lectures (80 minutes each) and a weekly lab (120 minutes). The lectures present conceptual material in an interactive format, and the labs develop skills for particular technologies, including an interactive development environment [NetBeans 2006], Java Server Pages [2006] for creating web applications, JDBC [Hamilton et al. 1997] for database connectivity, PostgreSQL [2006] for relational database support, and Lucene [Hatcher and Gospodnetic 2004] for information retrieval support. The skills, and program code that are developed in the labs can be readily transferred to the History Places project, which is described in the next section. Individual assignments build additional skills in SQL, entity-relationship modeling, and information retrieval algorithms.

2.2 Learning Objectives

History Places addresses two pedagogical goals. First, it aims to actively engage students in the principles of database management and information retrieval and to equip students with an appreciation for the relative merits of these two approaches to information storage and access. Second, it aims to formulate a problem that will be motivating to students with diverse backgrounds and varied inclinations toward technology but who are all vested in the study, design, and use of information systems. The specific learning objectives of these goals are the following.

(1) Create conceptual models for small problems that require unary, binary, and ternary relationships and generalization/specialization relationships. Students learn to represent conceptual models using UML. The key objective is for students to appreciate the value of separating domain models from logical models that target a particular implementation platform. This is a crucial concept that assumes even greater importance as new options for target implementations grow to include XML repositories, content management systems, object-oriented systems, information retrieval systems, and, of course, relational databases [Vakali et al. 2005].

(2) Transform a conceptual model into a logical model targeted to a relational database implementation. Students learn to map a conceptual model into a logical model that can in turn be readily transformed into a relational database schema. At this point, the central concepts of the relational model—domains, keys, integrity constraints, and so on—are engaged.
(3) Transform a conceptual model into a specification for implementing a best-match retrieval engine. Students learn to transform elements of the conceptual model into an information retrieval specification and implement a best-match retrieval component using Lucene [Hatcher and Gospodnetic 2004]. The key skills developed include identifying the documents and metadata to be indexed, specifying appropriate indexing policies, specifying query types, and specifying how the document collection will be kept consistent with information in the database.
(4) Specify the functional and user-interface requirements with Use Case and Wireframe models. To fully describe the functional requirements, students are asked to create use cases. To specify the user interface and to clarify the use case, students create wire-frame models of the user interface. These methods are presented in simplified form as part of the design process. To signal to students that effort should be directed toward the data modeling and implementation work, the user interface is not graded.

These learning objectives are prerequisites for classes on advanced topics in database management and information retrieval and for a self-directed capstone experience.

3. HISTORY PLACES PROJECT BRIEF

Students are given a project brief containing the following information.

3.1 Project Objective

The objective of this project is to develop a detailed specification and a working prototype of History Places. To support this 10-week process, you will prepare a set of deliverables.

3.2 Root Concept

History Places is an information system that enables people to submit photographs and other media such as audio files of places so that people can perceive how a building, view, landscape feature, or artifact has changed over time. Being able to perceive and discuss change is a fundamental aspect of human discourse. Thus, a system that enables these tasks can be used for many purposes. That, at least, is the theory.

3.3 Short Scenarios

Consider these short scenarios that reify the root concept.

3.3.1 *Environmental Health.* An environmental group aims to show how the natural landscape in the western United States has changed. Thus,

they decide on a set of sites and ask people to submit photographs of those sites.

3.3.2 *Construction Site Progress.* An architect wishes to capture the progress of a construction project. Thus, she takes pictures of the building site over a period of a year to document how it developed.

3.3.3 *Community History.* An elementary school teacher wishes to develop his students' interest in history. Thus, he gives cameras to his students and asks them to go into the community with their parents to take pictures. The system is seeded with a set of historic photographs. Students begin by browsing these photographs and selecting one or two locations for their own current shots.

Before reading further, please identify a similar scenario that might benefit from a History Places orientation. What features of your scenario make it consistent with these examples?

3.4 User Tasks in History Places

Consider the following tasks that people might perform in History Places. In these tasks, SMALL-CAPS-TEXT represents an entity/noun and underlined-text represents an operation/verb.

(1) A USER can submit a PHOTOGRAPH or AUDIO RECORDING for a PLACE.
(2) A USER can browse a DIRECTORY of PLACES of arbitrary depth and breadth.
(3) A USER can find a particular PLACE or PHOTOGRAPH by a KEYWORD.
(4) A USER can find PHOTOGRAPHS that are similar to another PHOTOGRAPH.
(5) A USER can browse a timeline and discover PLACES or PHOTOGRAPHS.
(6) A USER can vote on the quality of a PLACE or PHOTOGRAPH.
(7) A USER can discover the most popular PLACES or PHOTOGRAPHS.
(8) A USER can find similar USERS.
(9) A USER can comment on a PLACE or PHOTOGRAPH.
(10) A USER'S BEHAVIOR is tracked and recorded for future analysis.

Before reading further, select a scenario from this not or one of your own and consider which tasks you think are most relevant. Are any tasks missing? Do some tasks for your scenario need to be modified?

Please note that it will not be possible to implement complete solutions for all these tasks. You will implement partial solutions for only the most important tasks, and it will be up to you, in consultation with the teaching assistant and instructor, to decide what they are.

3.5 General Requirements

Your solution to the History Places project should satisfy the following general requirements.

(1) Develop a general abstraction for the concept of PLACE. It should be possible to represent a hierarchy of PLACES of arbitrary depth (e.g., World : South Pacific : Fiji : Viti Levu : Suva : Central Square).

(2) Develop an abstraction that allows you to represent both static media (e.g., PICTURES) and dynamic media (e.g., AUDIO RECORDINGS).
(3) Develop an abstraction that allows you to track which users submitted MEDIA to PLACES. This abstraction will be important for giving credit to frequent users and for identifying users who may be abusing the system.
(4) Develop an abstraction for users that includes a profile, with such information as age, gender, interests, favorite places, and so on.
(5) You should impose no editorial control on your system (i.e., submissions are not to be reviewed by an editor).
(6) Assume that image, audio, and other such files are stored in an external storage infrastructure and these files have unique identifiers that can be accessed by hypertext links. Thus, a file upload function is not required.
(7) Do not implement a secure user authentication scheme. You may encode unencrypted user identifiers on HTTP requests.
(8) You should not develop an extensive collection of media; collect enough items to illustrate the key features of your system.
(9) A high-quality visual design is not required. The emphasis of this project should be on interesting and powerful functions. The user interface needs only to reveal these functions and be neat and clear.

As you proceed through the project, you should repeatedly return to this list of general requirements and use them to guide your decision-making.

4. TEACHING NOTES

4.1 Project Logistics and Process

In addition to the conceptual description and requirements of History Places, the project brief contains guidelines for team organization, expected deliverables, and the development process.

4.1.1 *Team Organization.* Teams of three are selected by the instructor on the basis of a skills inventory, which assesses students' skills for programming, Web development, and SQL. Results from these assessments can be used in two basic approaches. First, teams can be formed so that each team has students with relatively strong and weak technical skills. Under this approach, all teams have about equal technical capacity. Alternatively, teams can be formed so that each team has students with relatively similar technical skills. Under this approach, some teams are relatively strong, others relatively weak. While each approach has strengths and weakness, I have found the second approach, on the whole, to, be more effective for learning. (This issue is discussed further in Section 5.2.)

4.1.2 *Deliverables.* History Places runs over a 10-week quarter. To help students manage the development process, as shown in Table I, a report outline is given and each deliverable is described in reasonable detail. Again, for the same reason, the quality of the user interface is not graded; this

Table I. Table of Contents for History Places Report

1. Title page (1 page)
2. Executive summary (1)
3. Table of contents (1)
4. Introduction (∼3)
    4.1 Vision
    4.2 Functional Requirements
5. Design
    5.1 Conceptual Model (∼2)
    5.2 Logical Model (∼2)
    5.3 Information Retrieval Module (∼2)
    5.4 System Architecture (∼2)
6. Functions (−18 pages; −2 pages/function)
    For each function,
        Responsible person
        A use case
        A wire-frame sketch
        Feature and limitations
7. Conclusion (∼1)
    Status and next steps
8. References
9. Acknowledgements
10. Appendices
    A: Team Reflection on process (−4)
    B: Additional wire-frame sketches
    C: If needed, other appendices

pedagogical move is intended to help students focus on the technical aspects of this project.

Further, students are instructed to design and implement the following 9 functions. In consultation with the instructor and teaching assistant, students are able to change these functions if their specific vision of the project warrants.

Functions #1

1A. Create a place
1B. Upload an image
1C. Create a user

Functions #2

2A. Browse a hierarchy of places
2B. Browse photographs on a timeline
2C. Browse a list of users

Function #3

3A. Search places/photographs by keyword
3B. Search photographs by keyword and date
3C. Given a user, find similar users

To manage the process, students are instructed to create a simplified use case and wire-frame sketch of the user interface for each of these functions, and present a brief discussion of the strengths and weaknesses of their implementations. As well, students must make their code available at their History Places Web site.

Students submit two draft reports to allow for incremental feedback, and one final report. Students also give a demonstration of the final system to the teaching assistant and instructor and may give a demonstration to their student peers at the final lab session. Attendance at the final lab session is optional, providing a venue for enthusiastic students to discuss and reflect on their work and to demonstrate their intrinsic interest in the material.

4.2 Labs and Skills Development

Weekly 2-hour lab exercises support the development of the necessary skills for implementing History Places. While the labs do not result in specific deliverables for History Places, they do develop skills that are directly transferable to the project and often result in code examples that can be readily adapted to the project. Skills for entity-relationship modeling are developed in class and assessed through individual assignments.

The labs are now briefly described. Lab 1 introduces students to the software development environment [NetBeans 2006] by implementing a domain-specific search interface that calls on the Google search service [2006]. Students learn to parse HTTP requests and return HTTP requests in the form of HTML pages. Lab 2 introduces students to SQL, to submitting scripts at the command line, and to using an interactive SQL client. In Labs 3–4, students learn to use JDBC to query databases and produce dynamic Web pages. In Lab 5, students engage some basic concepts of teamwork, and we discuss the challenges that teams are encountering with History Places. At this point in the quarter, students tend to be concerned about their progress on History Places and begin to recognize that the project is more difficult than it first appeared. In Labs 5–10, students work on their projects and, as a class, we discuss common problems an approaches to their solution. Finally, Labs 7–8 introduce students to the Java API for the Lucene search engine.

4.3 Conceptual Model

Figure 1 shows a target conceptual model for History Places. At (1), a ternary relationship is used to model the idea that Users submit MediaItems to Places. At (2), a Place is modeled as a named entity with zero or more subplaces. The recursive relationship is shown as an aggregation type to indicate a strong whole-part coupling between a parent and its children. At (3), a MediaItem is modeled as a general entity that must be specialized as either an AudioFile or ImageFile. At (4), binary relationships are used to model the idea of visiting places and creating new places. The Visits relationship is many-to-many, whereas the Creates relationship is one-to-many. To allow for the case where a Place is created by the system, the participation constraint is shown as optional. At (5), two binary relationships are shown to model the View and Vote

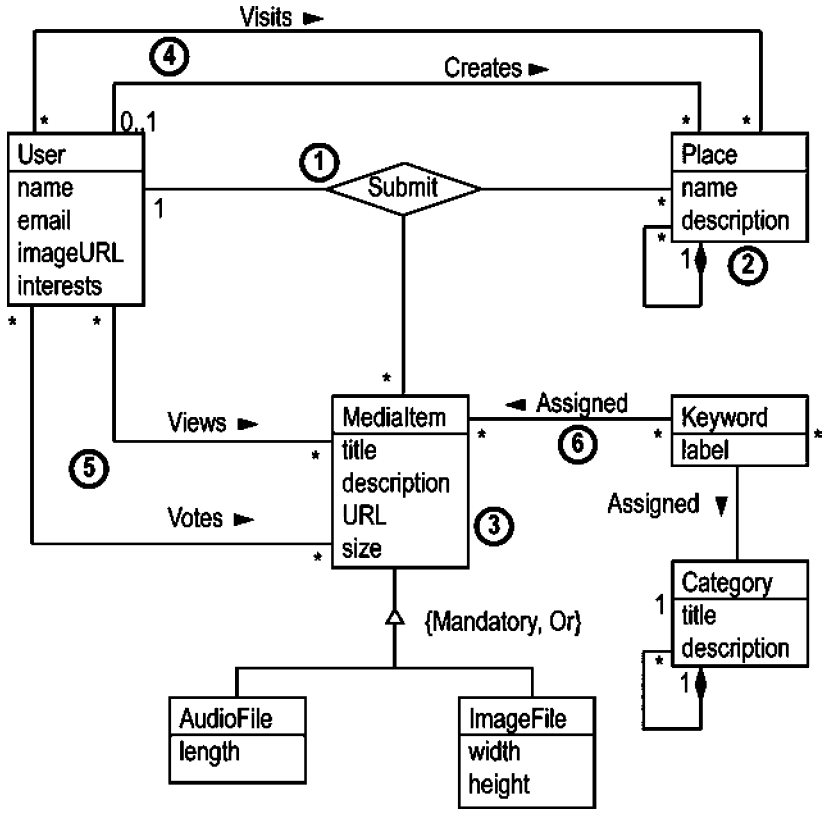

Fig. 1. A reference-conceptual model for History Places.

operations. Finally, at (6), entities are shown to model the assignment of keywords to MediaItems from a controlled vocabulary which is structured in a tree using the same pattern as used to model Places.

Of course, this conceptual model is one of many plausible solutions. Foremost, its role is to guide the instructor when giving students guidance as solutions unfold. One common limitation of students' initial models is that a one-to-many binary relationship is often used to model the relationship between Places and MediaItems. With this approach, information about who submitted items is not modeled. Another common limitation is that the entity Place is modeled as a postal address with country, state/province, city, street, and zip/postal code. This approach limits how places are described and the granularity of the description. As a final example, students often initially propose a model with a specific type of media, often images, and do not consider the issues of modeling both static and dynamic media.

To begin a discussion of these issues with students, it can be helpful to ask questions about the expected usage of a proposed data model such as the following.

(1) Can you write a query to determine to what Places a user has contributed MediaItems?

(2) How would you represent an isolated place in the Rocky Mountains with your abstraction of Place?
(3) Can you represent a full range of different kinds of media; do different media require different attributes?

As well as these common limitations, teams often propose interesting elaborations beyond the reference model shown in Figure 1. For example, a Project entity is occasionally proposed to provide scope to a set of Users, MediaItems, and Places. At (5), the feedback relationships, View and Vote, can be represented more generally as a Feedback entity with specialized subclasses. A final example is that specialized entities can be derived from Place to model different ways of specifying the physical location of places (e.g., postal address or longitude-latitude coordinates). History Places is a relatively narrow but interesting domain for conceptual modeling as these example limitations and elaborations illustrate.

4.4 Logical Model

As shown in Figure 2, the logical model is a relatively straightforward translation of the conceptual model. One interesting issue is deciding how to treat the ternary relationship. At (1), a general solution is shown where the relationship is modeled as an entity to allow for a user or group of users to submit an item to a place. Further, the item might simultaneously reside in several places. This is clearly more expressive than required, but it is also perhaps more natural because of the simple mapping between complex relationship and entity. An alternative solution is to relate the entity MediaItem to User and Place with 2 one-to-many relationships as shown at (7) at the bottom of Figure 2. At (2), the unary relationship is modeled using a single entity and with a foreign key to the entity's primary key. At (3), the superclass and its subclasses are modeled in three tables with keys from a common domain allowing the tables to be joined. At (4) and (5), the many-to-many relationships are reduced to 2 one-to-many relationships with corresponding weak entities. The other features of the model are relatively straightforward.

4.5 Information Retrieval Model

The project brief for History Places asks students to implement best-match query functions with the Lucene search engine [2005]. Students are asked to work with a separate API because best-match retrieval algorithms are often not available in relational databases today. Second, this requirement develops students' skills for integrating software components and managing the associated complexity.

Recall that the design brief contains the following suggested queries: 1) search places or photographs by keyword, 2) search photographs by keyword and date, and 3) given a user, find similar users. Unlike database design, prescriptive methods for guiding the design of information retrieval systems are not in common use. To develop a specification that can be used to implement solutions, the following questions must be addressed.

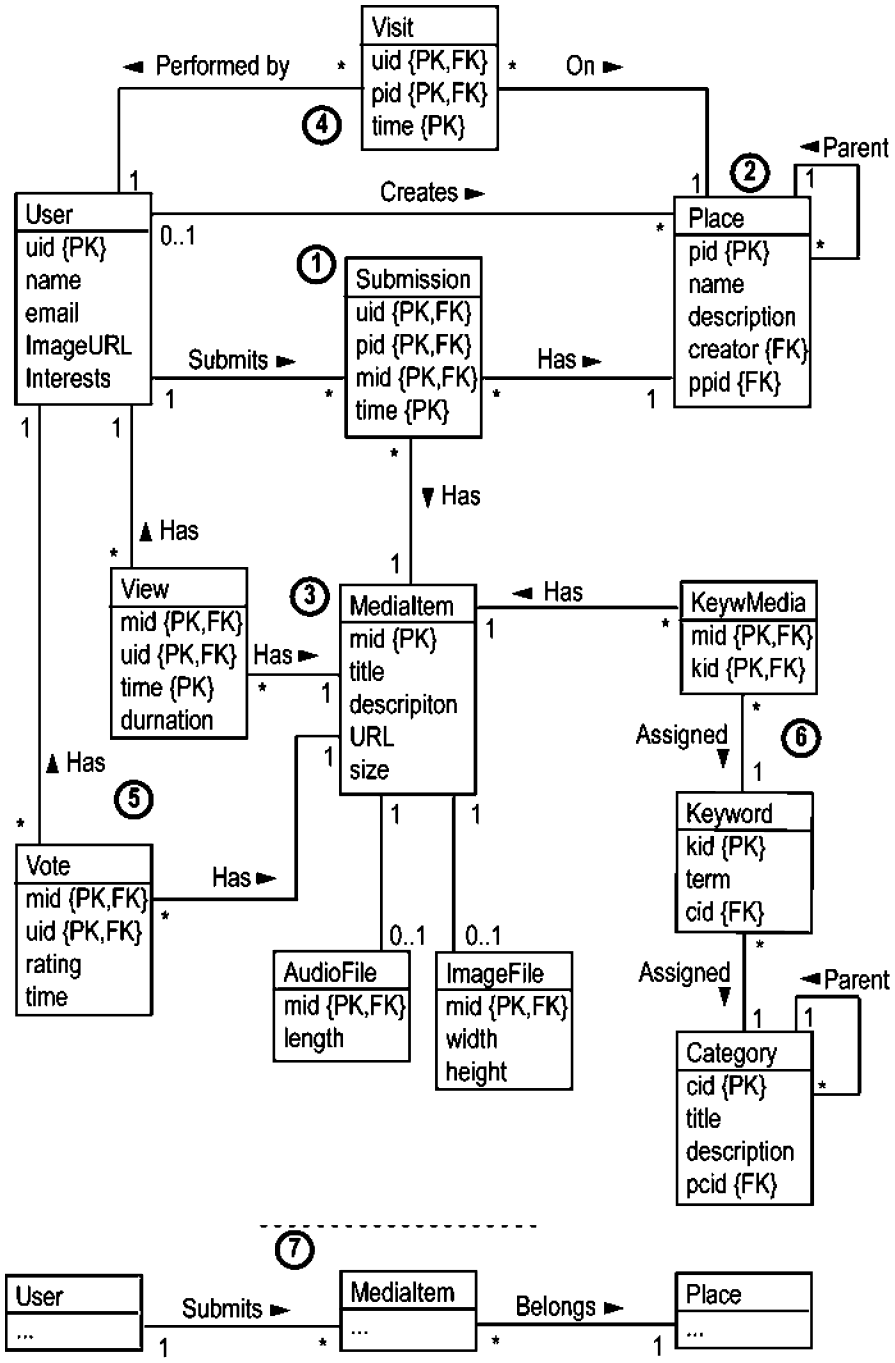

Fig. 2. A reference-logical model for History Places.

4.5.1 *What are the Documents To Be Retrieved?* In Lucene, documents, represented as a list of named fields, are the unit of storage and retrieval. Thus, the central problem is determining how to map entities in the logical model to the document abstraction that is provided by Lucene. One solution is to specify a separate document for each entity that is to be searched. In each case, the document is defined by two fields: 1) EntityId, which maps the document to a record in the corresponding relation and 2) BagOfWords, which holds a list of words that characterize the document. For example, a User document might be specified with two fields, UserID for identifying the user database entity and BagOfWords for storing the user's interests. Here, the field BagOfWords might consist of a list of words entered by the user when prompted to declare his or her interests at the time of registration. Alternatively, this field might be initialized with words through a more complex algorithm, for example, by adding all words contained in the image descriptions that are submitted by the user. Analogous documents can be readily created for the database entities Places and MediaItems.

4.5.2 *How Should the Documents Be Indexed?* Once documents are defined, it becomes necessary to specify how the terms within the fields should be indexed. The major parameters are 1) term identification, 2) folding case, 3) stemming, and 4) stop-word removal. When making choices for these parameters, good judgment is required. For example, it is probably inappropriate to remove stop-words (i.e., words that occur with great frequency such as the, on, etc.) of place names.

4.5.3 *What Is the Structure of Queries and Results?* Next, it is necessary to consider what information is available in the query and what information must be included in the results. For example, if the task is to find similar users to a given user, then the given user entity must be mapped to a query. A straightforward solution would be to create a query out of the keywords in the field Interests, and then match this query against all user documents, producing a ranking of the most similar users. For the Lucene search engine, a standard TF-IDF weighting function is used. (TF-IDF refers to term frequency-inverse document frequency and produces document weights that are a function of the frequency of a term's occurrence in the document and the relative rareness of the term across all documents, summed for all query terms in the document. Thus, documents that contain relatively more instances of rare words tend to receive the highest rankings [Belew 2001].) Continuing with the example, on the results side, it may be desirable to show result hits that consist of a thumbnail image, user name, list of words from the field Interests, and so on. Such pieces of information must be clearly identified in order to specify the database queries that need to be spawned from the initial search. This analysis of inputs and outputs leads to requirements that are addressed by the next question.

4.5.4 *How Are Data Combined From the Database and Search Engine?* Fundamentally, students must develop a solution that allows for the combination of results from two retrieval systems. Consider, for example, this search

task: List all photographs that are about mountain streams in the autumn and were taken between 1940 and 1945. This search decomposes into two separate queries: 1) a best-match query for mountain streams in the autumn and 2) an exact-match query for 1940 < date_taken < 1945. By taking the intersection of these two sets of identifiers, the final result set is obtained. Then, using the entity IDs, additional database queries must be performed to collect all the necessary data for the result presentation. The best student solutions implement this algorithm in a relatively general way so that various patterns of search dialog can be implemented.

4.5.5 *How are Data from the Database and Search Engine Kept Consistent?* This system suffers from data redundancy because similar data—records in the database and documents in the search engine—must coexist. Students must describe how the two data sets will be kept consistent. Two basic approaches exist. First, data from the database can be exported into a document collection on the file system which is then indexed. Alternatively, data from the database can be transferred into temporary documents which are indexed and then destroyed. Both approaches must be scheduled to run periodically. Incremental solutions that maintain the consistency of the two data sets as users add new media items are much more difficult to implement and outside the scope the History Places project.

4.5.6 *Using the Lucene API.* Answering the previous four questions constitutes a specification for the Information Retrieval module of History Places. The next step is to implement this specification with Lucene [Hatcher and Gospodnetic 2004]. Lucene provides a high-level API that hides most of the complexity normally associated with implementing IR applications. An outline for a basic implementation approach follows.

(1) *Representing Documents*. The Lucene class `Document` must be used to represent the database items that are to be indexed. This is done by adding one or more instances of the class `Field` to an instance of `Document` and by then assigning names and values to each `Field`.
(2) *Indexing Documents*. To index the documents, an instance of the class `IndexWriter` must be created. Then, the method `addDocument()` must be called for each document that is to be indexed. The `IndexWriter` instance must be parameterized with an instance of the class `Analyzer`, which will index the content of the document's fields according to particular policies (e.g., tokenization, stop-word removal, stemming, etc.). The output of this process is an inverted file stored on disk.
(3) *Searching Documents*. To search the documents, an instance of the class `IndexSearcher` must be created to provide search capability on the inverted file created in Step (2). Next, a user's free-text query must be represented as an instance of class `Query`. The `Query` instance must be created using the same `Analyzer` that was used to index the documents in Step (2). The class `IndexSearcher` provides a method `search()` that must be called with this query, and it returns an instance of the class Hits, which holds a list

Table II. Course engagement ratings from standard course evaluations. Median ratings are averaged across 6 classes between spring 2003 and spring 2006 (138 responses). Responses were on a 7-point scale (Always/Great/Much Higher-7, 6, 5; About Half/Average—4,3,2; Never/None/Much Lower-1).

Relative to other college courses you have taken:	
1. The intellectual challenge presented was__	$5.5(SD = 0.61)$
2. The amount of effort to succeed in this course was__	$5.5(SD = 0.60)$
3. The amount of effort you put into this course was__	$5.4(SD = 0.74)$
4. Your involvement in course (assignments, attendance, etc.)__	$5.2(SD = 0.66)$

of documents in ranked order. The class `Hits` provides access to each of the retrieved documents, allowing the user's output to be generated by iterating through this list.

To implement this approach requires about 150 lines of Java code. In summary, with the appropriate introduction, which is done in Labs 7–8, incorporating best-match search capability is feasible in the History Places project.

5. REFLECTIONS ON LEARNER EXPERIENCE

Quantitative assessments of the effectiveness of History Places have not been conducted. On standard course evaluations, students rate that the course is intellectually challenging and engaging (see Table II). While these ratings do not reveal anything specific about History Places, they do quantify general student feedback that History Places is challenging but rewarding and even fun. To get a qualitative student perspective on History Places, students are asked to write short reflective statements on what they learned, what they would do differently, and what they took away from the project. These statements can be used to consider how the History Places experience can be improved. The following discussion is based on these statements and my own reflections of the learner experience.

5.1 Innovation and Creativity

Case-based learning in information science and technology can take many forms [Carroll and Rosson 2005]. When teaching system development, project-based cases often present an extended scenario with many descriptive details, some explicit and others subtle or only implied. Through a careful examination of the narrative, students uncover these details from which functional requirements are derived and which in turn drive the design process. History Places, on the other hand, provides a pattern that outlines the basic boundaries of the problem, but students must devise their own vision and functional requirements. The rationale for this pedagogical move is to motivate students and to invite them to be curious and shape the project to their own interests.

The upside of this move is the possibility that students become vested in the material and motivated to engage in it because the problem is their own. The downside is the possibility that students choose a vision that leads to technical issues that are outside the central learning aims for the project. The downside is mitigated by providing students with examples—scenarios, tasks, and target

functions—that act as analogs in their own projects and by giving students feedback on their first project deliverable that is focused on the on the suitability of technical problems they are engaging.

In the reflective statements, students often underline this aspect of the project, saying such things as "The project gives students a chance of 100% creative control over the project" and "Our project didn't just fulfill the project requirements, it had an idea behind it that required certain functionality". At the same time, students often highlight the difficulty of bounding the problem, as represented in this quotation: "There are certain guidelines and requirements, but other than that, most of the other things are a free for all. It is good on one hand, but things can also get complicated and we can lose our way."

In any case, History Places has prompted students to envision a wide range of interesting systems for information collection and presentation, including using History Places to 1) promote alumni-university relations by enabling each generation of students to collect and organize photographs of campus life for future review and reflection, 2) promote civic discourse about the development and impact of the Seattle Monorail, a large urban development project (since stopped), 3) explore how people's dress has changed over time, 4) document the changes in a landscape that occur after a natural disaster, and 5) enable homeowners to contribute and discover historic photographs of their homes and homes that are similar to theirs. The brief scenarios given in the project brief are sometimes used by teams as a concrete starting point, and some teams appreciate this option. Seeking concrete models, students will occasionally ask to see working systems that were developed by teams on previous projects. I don't facilitate these requests because their solutions may cause teams to fixate on particular solutions to general problems. Further, as a practical matter, the implemented systems are generally available only for a short time after the completion of the course.

In summary, the History Places metaphor and its root concept, that is, to record and study how a place changes over time is intrinsically rewarding, provides a rich terrain for shaping particular problems. While place is an abstract concept, it also can be reified in many different settings, each with their own nuances and points of interest. This malleability offers students a good deal of room for finding problems and establishing their own problem-setting, while also directing them to solve specific technical problems (e.g., browsing a hierarchical directory).

5.2 Engaging a Diversity of Learners

History Places presents a familiar concept, information collection and access within specific domains. History Places is an interesting problem because it draws on students' familarity with engaging information problems from varied perspectives. This is a first step for accommodating a diverse group of learners. A second step is to help students structure the process for working the project so that each member of the team is given maximam opportunities to develop their skills. In this project, the skills to be developed are focused on technology, and

two factors are particularly important for creating an effective process. First, unlike students who are studying a single technical discipline such as Computer Science or Electrical Engineering, students in Informatics differ in their inclinations to engage technical topics. Some Informatics students enjoy working on technological topics and orient their educational and work experiences towards technologically-focused positions. Other students are not particularly inclined to engage topics in technology and are oriented towards positions in information management, project management, and so on. Second, students differ in their backgrounds for technology. Some students have a large and varied reservoir of experiences to draw on and are quite confident in their abilities, especially their abilities to learn technology as needed; other students, have fewer experiences and seem disinclined to learn new technological topics on their own. These differences present a major pedagogical challenge, without a simple solution.

As described in Section 4.4.1, I have considered two basic approaches. On the basis of a skills questionnaire, you can either pick groups to equalize skill levels across all groups or you can pick groups to equalize skills within groups. Under the first approach, I have observed that teams parcel out work to economize their effort by exploiting preexisting skills; thus, for example, students strong in technology do most of the design and implementation work and students strong in writing do that part of the work. When this selection strategy has been followed, students with relatively weak skills in technology have remarked in written reflections and in informal conversations that they have been disappointed at not having more thoroughly engaged implementation work because their colleagues did that work. This is a significant problem for a project that seeks to raise all students' abilities in information system design and implementation, and I have found that adding process does not solve it (e.g., by requiring teams to designate technical leads which are held responsible for specific functional components).

Under the second approach, however, where teams are selected so that they consist of students with similar technological skills in programming, Web development, and SQL and with similar goals (e.g., taking advanced courses in databases, pursuing careers in Web development, seeking a project management position, etc.), teams seem to collaborate more strongly; excellent systems are often built by teams with modest technical skills coming into the class. While this creates an imbalance in the teams' expected capacities across the class, my experience is that it results in better overall project outcomes and individual student accomplishments compared to teams that are formed to balance relatively weaker and stronger students. The basic reason is that when students have similar skills, they seem to be more willing to divide all components of the project among each other.

Even so, problems associated with the make-up of teams are relatively common, arising from different degrees of commitment to the course, incompatible schedules, varying skills in teamwork, and no doubt others. A further complication concerns the equitably of assessment: Should lower-capacity teams be graded on the same basis as the higher-capacity teams? In principle, it should be possible to take into account the teams' initial capacities and students' learning

goals and individualize the assessment. With public calls for more personalized learning [State of Washington 2006], this will become an ever more important direction to pursue. In summary, accommodating the diversity of technological skills and interests in system development is the most difficult aspect of this project-based case.

5.3 Technological Diversity

While History Places draws upon multiple disciplinary areas, including web technology, software engineering, user-centered design, and visual design, the emphasis is information storage and access with relational and information retrieval systems. Integrating these two technical areas to create a coherent working system is a significant challenge. Adding to the complexity, students work on separate functional components, which must be combined and presented through a Web server, which is complex in its own right. Students often report that no one piece is particularly difficult but that in combination the development process can be very difficult. Indeed, students must coordinate their technical development, and to do so, they must develop skills in reading and explaining each other's code, code integration, joint debugging, learning and teaching each other how things work, and so on. The breadth of this project leads to the risk that students fail to engage the learning goals.

Two approaches are used to mitigate this risk and have been found to be relatively successful. First, students are given lab assignments where they read and borrow from code examples. These code examples reveal enabling concepts and technologies that underlie the learning objectives (e.g., HTML client-server form-handling). Thus, students learn to adapt and borrow code from these assignments. Developing these materials so that students at different levels of skill can see the analogical mappings between such skills-oriented materials and the technical problems to be solved in History Places is a significant instructional design problem.

Second, students are informed that some otherwise important aspects of the History Places design will not be graded. For example, students are told not to implement authorization schemes, file upload and storage, and robust state management. Further, students are told that the user interface will not be graded, which is a disappointment to many teams. A consequence of these disincentives is that critical information security issues are not addressed, and many interface designs lack coherence and suffer from basic usability faults. Nevertheless, they do help students focus on the learning objectives of the course.

6. CONCLUSION

History Places is an engaging project-based case that is suitable for students with diverse backgrounds and interests in information system design. Its authentic nature derives from the root concept, that is, to implement a kind of digital library for tracking change, from the requirement that student teams establish their own visions, and from the need to combine multiple technological components into a working system. While the project is structured to develop skills in relational databases and information retrieval systems, it could be

used to engage other aspects of information systems, such as usability engineering, information architecture, content management, businesses modeling, and social computing.

ACKNOWLEDGMENTS

I am grateful to Melody Ivory-Ndiaye who encouraged me to write about History Places. Shaun Kane, Gifford Cheung, and Suzi Soroczak—superb teaching assistants—contributed many improvements to History Places. Joshua Ayson, Joel Johnson, and Scott Barker enthusiastically provided technical assistance over the past four years.

REFERENCES

BAEZA-YATES, R. AND RIBEIRO, B. 1999. *Modern Information Retrieval*. Retrieval evaluation (Chapter 3). Addison Wesley, New York, NY, 73–97.

BELEW, R. K. 2001. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge University Press, Cambridge, UK.

CAPPEL, J. J. 2002. A systems analysis and design case: ABC Church. *J. Inform. Syst. Educat. 12*, 4, 223–224.

CARROLL, J. M. AND ROSSON, M. B. 2005. A case library for teaching usability engineering: Design rationale, development, and classroom experience. ACM *J. Educat. Resour. Comput. 5*, 1, Article 3.

CHAU, M., Z. HUAG, H., AND CHEN, H. 2003. Teaching key topics in computer science and information sytems through a web search engine project *ACM J. Educat. Resour. Comput. 3*, 3, Article 2.

CONNOLLY, T. AND BEGG, C. 2005. *Database Systems: A Practical Approach to Design, Implementation, and Management*. Addison-Wesley, Reading, MA.

DATE, C. J. 1995. *Introduction to Database Systems 5th Ed.* Addison-Wesley, New York, NY.

DEAN, J., BARROSO, L. A., AND HÖLZLE, U. 2003. Web search for a planet: The Google cluster architecture. *IEEE Micro 23*, 2, 22–28.

GONCALVES, M. A., FOX, E. A., WATSON, L. T., AND KIPP, N. A. 2004. Streams, structures, spaces, scenarios, societies (5S): A formal for digital libraries. *ACM Trans. Inform. Syst. 22*, 2, 270–312.

GOOGLE. 2006. Google SOAP Search API, http://www.google.com/apis/.

HAMILTON, G., CATTELL, R., AND FISHER, M. 1997. *JDBC Database Access with Java*: *A Tutorial and Annotated Reference*. Addison-Wesley, Reading, MA.

HATCHER, E. AND GOSPODNETIC, O. 2004. Welcome to Lucene in Action, http://www.lucenebook.com.

HENDRY, D. G. AND CARLYLE, A. 2006. Hotlist or bibliography? A Case of genre on the Web. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences*.

HENDRY, D. G., JENKINS, J. R., AND MCCARTHY, J. F. 2006. Collaborative bibliography. *Inform. Process. and Management 42*, 3, 805–825.

JAVA SERVER PAGES. 2006. http://java.sun.com/products/jsp/.

KIFER, M., BERNSTEIN, A., AND LEWIS, P. M. 2006. *Database Systems: An Application-Oriented Approach 2nd ed.* Addison-Wesley, New York, NY.

LEVY, D. M. AND MARSHALL, C. C. 1995. Going digital: A look at assumptions underlying digital libraries. *Comm. ACM 38*, 4, 77–84.

MYERS, M. 2003. An IS Capstone Project: The Mywick Property Management System. *J. Inform. Syst. Educat. 13*, 235–239.

NARDI, B. A., SCHIANO, D. J., GUMBRECHT, M., AND SWARTZ, L. 2004. Why we blog. *Comm. ACM 47*, 12, 41–46.

NETBEANS. 2006. Welcome to NetBeans. http://www.netbeans.org/.

PARKER, K. R. 2003. A database design case: Teton whitewater kayak. *J. Inform. Syst. Educat. 14*, 3, 271–274.

POSTGRESQL. 2006. PostgresSQL: The world's most advanced open source database. http://www.postgresql.org/.

SMITH, J. M., LIGHT, A., AND ROBERTS, D. 1998. *Philosophy and Geography III: Philosophies of Place*. Introduction: Philosophies and geographies of place 1–19. Roman and LitterField, New York, NY.

SMITH, M. A. 2004. Portals: Toward an application framework for interoperability. *Comm. ACM*, *47*, 20, 93–97.

STATE OF WASHINGTON. 2006. Washington learns final report: World-class, learner focused, seamless education. http://www.washingtonlearns.wa.gov/.

VAKALI, A., CATANIA, B., AND MADDALENA, A. 2005. XML data stores: Emerging practices. *IEEE Internet Comput. 9*, 2, 62–69.

WEDEL, T. L., BEHNEZHAD, A. R., AND GRAY, G. L. 2004. A data modeling case: Writers Guild of America East. *J. Inform. Syst. Educat. 15*, 1, 13–17.

WIKISYM. 2006. The international symposium on Wikis: Wiki research and practice. http://www.wikisym.org/.

Received August 2006; revised January 2007; accepted February 2007