INFO-445: Advanced Database Design, Management, and Maintenance

Spring 2007
B.S. Informatics
Information School
University of Washington

Advanced perspectives on DBMS theory, architecture and implementation. Conceptual, logical and physical modeling. Index structures, query optimization and performance tuning, relational algebra, transaction processing and concurrency control. Operational databases, decision support systems and data warehousing. Projects in database implementation and integration. Social implications of large distributed database systems. *Prerequisite*: INFO 340, CSE 373.

Course website & Listserv

http://courses.washington.edu/info445/

info445a_sp07@u.washington.edu
(Archive: https://mailman1.u.washington.edu/mailman/listinfo/info445a_sp07)
Registered students are subscribed automatically using their UW mail account.

Credit Hours

5 (3 lecture hours; 2 lab hours; 10 outside hours)

Meeting times

Lecture	Monday/Wednesday 130 – 250, MGH 254
Lab	Thursday 330 – 520, MGH 430

Instructor

David Hendry, Assistant Professor
330J Mary Gates Hall
dhendry@u.washington.edu | http://faculty.washington.edu/dhendry

Office hours: Monday, 330 – 430 or by appointment.

Teaching assistant

There will be no teaching assistant for this class

Student services

Dowell Eugenio, Student Services Administrator
470E Mary Gates Hall
deugen3@u.washington.edu

Please note: If you have any concerns about a course or the TA, please see the TA about these issues as soon as possible. If you are not comfortable talking with the TA or not satisfied with the response that you receive, you may contact the instructor of the course. If you are still not satisfied with the response that you receive, you may contact Joseph Janes, the Associate Dean for Academics in 370 Mary Gates Hall, by phone at : (206) 616-0987, or by e-mail at jwj@u.washington.edu. You may also contact the Graduate School at G-1 Communications Building, by phone at (206) 543-5900, or by e-mail at efeetham@u.washington.edu

Overview

The big question we shall ask in the course is: What are the merits of different kinds of data models for representing, storing, and accessing information? From this top-level question come subsequent questions, such as: What the central problems that all database models must address? Why have object-oriented databases emerged? How are they better (or worse) than the relational model? How do Object-Relational models seek to integrate both the "object" and "relational" approaches to represent, store, and access data? What is required of a database so that XML documents can be stored, updated, and access efficiently? How does XML integrate with relational databases? How does an application designer decide on a particular data model? How are fundamental database issues, such as data integrity, query processing, concurrency, efficient update and access, addressed with object-oriented and XML approaches? By answering these questions, you will expand your knowledge for databases and be equipped to tackle more complex data modeling problems and to learn more about databases on your own.

We shall seek to answer these questions by first deepening our knowledge for the relational model by covering the following advanced topics: 1) Rules for transforming conceptual models to relational schemas (conceptual and logical database design); 2) Performance monitoring and tuning (physical database design); 3) Transaction management; and 4) Query Processing. These topics shall be considered within a database design methodology, which covers conceptual, logical, and physical database design through a serious of stepwise refinements. Then, in the second part of the course, we shall examine three alternative data models: Object-Oriented Databases, Object-Relational Databases, and XML/DB integration. Throughout the course we shall study the concepts that underpin of these models and then explore the practical aspects of their use through weekly activities. Finally, you will have an opportunity to investigate a data model (or system) of your choice and top teach the class about what you've learned.

Textbooks and readings

The textbook for this is course is:

Connolly, T. M. & Begg, C. E. (2003). *Database Systems: A Practical Approach to Design, Implementation, and Management* (4th Edition) New York: Addison-Wesley Publishing. [ISBN: 0321210255]

We shall read chapters largely from the second-half of the textbook – I assume that you are comfortable with the material in chapters 1 – 6, 9, and 11 – 13 (these chapters were covered in INFO-340, the prerequisite to this class). Finally, the textbook will be supplemented with several readings, which will be posted on the course website.

Learning

Aims

The general aims of this course are to:

1. Expand and deepen knowledge for the relational database model
2. Develop knowledge for alternative database models, including object-oriented, object-relational, and semi-structured information (XML) approaches.

Objectives

On the successful completion of this course, you should be able to:

1. Apply a stepwise refinement methodology for transforming a complex conceptual model into a database schema
2. Describe how a database performance can be monitored and improved
3. Discuss and apply approaches for improving a database's performance, including denormalization, use of indexes, and data partitioning
4. Discuss concurrency control and describe major kinds of concurrency problems, and discuss database mechanisms for achieving Atomicity, Consistency, Isolation, and Durability
5. Employ the SQL commands BEGIN, COMMIT, and ROLLBACK to maintain consistency between the database and model of the world
6. Describe the major features of Object-Oriented Database Management Systems and discuss when OODBMS are appropriate
7. Describe the Object-Relational model and the problems it seeks to address
8. Discuss why an Object-Oriented or Object-Relational database might be preferred over a relational database
9. Create and query simple XML documents
10. Discuss the major approaches for integrating XML and design database applications that use both the relational and semi-structured data models.

Academic accommodations

To request academic accommodations due to a disability, please contact Disabled Student Services: 448 Schmitz, 206-543-8924 (V/TTY). If you have a letter from DSS indicating that you have a disability which requires academic accommodations, please present the letter to me so we can discuss the accommodations you might need in the class.

Academic accommodations due to disability will not be made unless the student has a letter from DSS specifying the type and nature of accommodations needed.

For additional information, see *Statements to Ensure Equal Opportunity and Reasonable Accommodation*, downloaded March 5, 2003, http://www.washington.edu/admin/eoo/eoost.html

Academic honesty

The essence of academic life revolves around respect not only for the ideas of others, but also their rights to those ideas and their promulgation. It is therefore essential that all of us engaged in the life of the mind take the utmost care that the ideas and expressions of ideas of other people always be appropriately handled, and, where necessary, cited. For writing assignments, when ideas or materials of others are used, they must be cited. The format is not that important–as long as the source material can be located and the citation verified, it's OK. What is important is that the material be cited. In any situation, if you have a question, please feel free to ask. Such attention to ideas and acknowledgment of their sources is central not only to academic life, but life in general.

Please acquaint yourself with the University of Washington's resources on academic honesty:
http://depts.washington.edu/grading/issue1/honesty.htm

Students are encouraged to take drafts of their writing assignments to the Writing Center for assistance with using citations ethically and effectively. Information on scheduling an appointment can be found at:
http://www.uwtc.washington.edu/resources/eiwc/

Copyright

All of the expressions of ideas in this class that are fixed in any tangible medium such as digital and physical documents are protected by copyright law as embodied in title 17 of the United States Code. These expressions include the work product of both: (1) your student colleagues (e.g., any assignments published here in the course environment or statements committed to text in a discussion forum); and, (2) your instructors (e.g., the syllabus, assignments, reading lists, and lectures). Within the constraints of "fair use", you may copy these copyrighted expressions for your personal intellectual use in support of your education here in the iSchool. Such fair use by you does not include further distribution by any means of copying, performance or presentation beyond the circle of your close acquaintances, student colleagues in this class and your family. If you have any questions regarding whether a use to which you wish to put one of these expressions violates the creator's copyright interests, please feel free to ask the instructor for guidance.

Privacy

To support an academic environment of rigorous discussion and open expression of personal thoughts and feelings, we, as members of the academic community, must be committed to the inviolate right of privacy of our student and instructor colleagues. As a result, we must forego sharing personally identifiable information about any member of our community including information about the ideas they express, their families, life styles and their political and social affiliations. If you have any questions regarding whether a disclosure you wish to make regarding anyone in this course or in the iSchool community violates that person's privacy interests, please feel free to ask the instructor for guidance.

Knowing violations of these principles of academic conduct, privacy or copyright may result in University disciplinary action under the Student Code of Conduct.

Student Code of Conduct

Good student conduct is important for maintaining a healthy course environment. Please familiarize yourself with the University of Washington's Student Code of Conduct at:
http://www.washington.edu/students/handbook/conduct.html

Assessment

Assessment	**Worth**
Eight weekly activities	60%
Project – Information Tasks Framework (IFT)	30%
Participation	10%

Weekly Activities

The weekly activities will be divided into two parts. Part I of all activities is due at the start of Monday's class on week X (shown below) and part II is due at the start of Monday's class on week X+1.

You may complete the eight activities alone or in groups of two. If you work in groups of two you are expected to work synthetically – that is, each member of the group fully works on the whole assignment. Throughout the quarter, you may work with as many different partners as you like.

Activity	**Week**
A1. Entity-Relationship modeling and normalization	#2
A2. Performance monitoring and tuning	#3
A3. Transaction management	#4
A4. Object-oriented database management systems	#5
A5. Object-relational databases	#7
A6. XML/SQL integration	#8

Project – Information Tasks Framework	**Due**	**Week**
Proposal	Apr 19	#4
Draft report	May 9	#7
Presentation	TBD	#9 – #10
Final Report	1 June	#10

You may complete the project alone or in groups of 1 – 4. The choice of groups is up to you.

The project brief for this project will be handed out at the start of week #2 and refined – as a class – during the lab on the second week. You will have significant room to pursue your own interests within the scope of the project.

Participation in classes/labs

It is important to the instructor and teaching assistant that you help make INFO-455 fun, interesting, and challenging. With spirit and a professional manner, we can create a supportive and rewarding learning environment.

Among the things you can do are:

1. Treat all with respect – be constructive in all discussions
2. Come to class prepared – read carefully and be ready for discussion
3. Be an active listener – be attentive, be engaged, use in-class technology with discretion
4. Ask challenging questions, participate in discussion
5. Comment, build on, or clarify what others have done or said
6. Help your classmates use development tools and technologies – build infrastructure
7. Post useful or interesting information to the class discussion list
8. Visit the instructor during office hours to chat, to ask questions, or to give feedback.

Please write a 2 or 3 paragraph personal statement on how you contributed to the class (optional). If you submit a statement, it is due May 30 at the start of class.

Grading criteria

Work in this course will be graded to criteria. In other words, you won't be graded on a curve. Each deliverable is designed to test your achievement against one or more of the learning objectives. Different assignments emphasize different learning objectives. The meanings of grades are described below.

General grading information for the University of Washington is available at:

- http://www.washington.edu/students/gencat/front/Grading_Sys.html

The iSchool has adopted its own criteria for grading graduate courses. The grading criteria used by the iSchool are available at:

- http://depts.washington.edu/grading/practices/guidelin.htm

Grade	Performance Quality*
3.9 - 4.0	Superior performance in all aspects of the course with work exemplifying the highest quality. Unquestionably prepared for subsequent courses in field.
3.5 - 3.8	Superior performance in most aspects of the course; high quality work in the remainder. Unquestionably prepared for subsequent courses in field.
3.2 - 3.4	High quality performance in all or most aspects of the course. Very good chance of success in subsequent courses in field.
2.9 - 3.1	High quality performance in some of the course; satisfactory performance in the remainder. Good chance of success in subsequent courses in field.
2.5 - 2.8	Satisfactory performance in the course. Evidence of sufficient learning to succeed in subsequent courses in field.
2.2 - 2.4	Satisfactory performance in most of the course, with the remainder being somewhat substandard. Evidence of sufficient learning to succeed in subsequent courses in field with effort.
1.9 - 2.1	Evidence of some learning but generally marginal performance. Marginal chance of success in subsequent courses in field.

*Taken from Faculty Resource on Grading, downloaded March 5, 2003, http://depts.washington.edu/grading/practices/guidelin.htm

Standard cover sheet

To protect your privacy when exercises are returned and to facilitate communication, submitted work must have a cover sheet. The cover sheet must include the following information and be formatted nicely:

- Course name,
- Quarter, program, department, and university
- Assignment name
- Your name and e-mail address
- A date
- A web site address (if relevant).

Staple the exercise pages to the cover sheet.

Late policy

1. If you will miss the deadline, you should inform the instructor as soon as you can, indicating when you will submit the work. The instructor will try to accommodate your needs. You should use this clause only for extraordinary personal reasons.
2. It is at the instructor's discretion to accept late work or assign late penalties (see 1 above). For any late assignment, 10% will be taken off your work per day. After five days, your work will not be accepted.
3. Late work must be handed to the instructor or teaching assistant in person. You may also be able to hand work in at the front desk of the Information School and at student services but this cannot be guaranteed.

Work that is handed in late is penalized for two reasons. First, to be fair, all students should be given the same time limits. Second, if you spend too much time on one assignment, it is quite likely that you will have insufficient time to spend on subsequent assignments.

Right to revise

The instructor reserves the right to revise this syllabus.

Re-grading policy

To have work re-graded, you must submit a Re-grade Request within five days of when your work was returned. The request must be a single page, printed on paper or sent by e-mail. It should contain the following information:

- Re-grade Request
- The information contained on the standard cover sheet
- An explanation for why you believe you deserve a higher grade.

The instructor, possibly in collaboration with the teaching assistant, will consider your request. If the instructor is convinced by your argument, your work will be re-graded. If not, the instructor will send you e-mail explaining why. No re-grades will be considered for late work.

Guidelines on using e-mail

When communicating with the instructor or teaching assistant, please follow these guidelines:

- You are welcome to give feedback to the instructor and teaching assistant about the course, to ask a question about an assignment, to share an interesting article or resource, to report that you will be absent from a class/lab, to request additional time for an assignment (because of significant health, personal, or educational matter), or similar communication ;
- Whenever appropriate, please copy the class listserv with your question or comment;
- E-mail concerning assignments might not be replied to if it is sent within 36hr of the assignment due date;
- If your e-mail concerns your grade, please follow the re-grading policy (see above);
- E-mail that is sent on Friday afternoon or over the weekend is not replied to until Monday or Tuesday of the following week;
- If you don’t receive a reply within 2 days or so, please resend your e-mail or ask about it during class or lab.

Class Schedule

Week 1: Overview

Read	Chapters 11, 12 & 13 (Review)
L1	Greetings
L2	ER modeling vs. normalization
Lab	Review of tools

Week 2: Conceptual and Logical Database Design

Read	Chapters 15 & 16
L1	Methodology – Stepwise refinement
L2	Deriving relations from ER models
Lab	Project work – Goals & Approaches

Week 3: Physical Database Design

Read	Chapters 17 & 18
L1	Query processing & indexes
L2	Monitoring and tuning
Lab	Comparative performance analysis

Week 4: Transaction Management

Read	Chapter 20
L1	Concurrency control
L2	Transaction models
Lab	Transaction design and simulation

Week 5: Object-Oriented Databases

Read	Chapters 25 & 26 (27 optional)
L1	Object-oriented concepts and databases
L2	Persistent programming languages
Lab	Zope

Week 6: Guest Talks

Read	TBA
L1	TBA
L2	TBA
Lab	Project work

Week 7: Object-Relational Databases

Read	Chapter 28
L1	Third Manifesto
L2	PostgreSQL extensions
Lab	Project Work

Week 8: XML/DB Integration

Read	Chapter 30
L1	Introduction to XML
L2	Querying XML
Lab	Project Work

Week 9: XML/DB Integration

Read	Two research papers (on website): ● Beyer et al., 2006 ● Pal et al., 2005
L1	Hybrid approaches
L2	Application development
Lab	Project work/presentations

Week 10: Socio-Technical Issues

Read	Chapter 19
L1	Memorial Day Holiday
L2	Security, system failure & risk
Lab	Project work/presentations

Spring 2007
A1: Entity-Relationship modeling and normalization
Worth: 10%

Learning objective

Apply a stepwise refinement methodology for transforming a complex conceptual model into a database schema.

Part I

1. Briefly describe the key differences between **conceptual**, **logical**, and **physical** database design.

2. Do you think that a database design methodology will lead to better databases? Please explain your answer.

3. Precisely explain why there are two relationships between Staff and Supervisor in Figure 15.6.

4. How would you change the ER model in Figure 15.8 to accommodate more complex families – for example, families with biological and step mothers and fathers? Draw a new model in UML and briefly describe the changes you've made.

5. a) Precisely define **functional dependency**. Precisely define **3NF** and explain why 3NF is desirable. b) Consider the notion of a "social bookmark", that is, a "bookmark" that can be publicly seen by anyone. Identify the functional dependencies between the data items in the table below. c) Propose a schema in which all relations are in 3NF.

Table: Bookmark

ATTRIBUTE	NOTES
url	The url of the bookmark
title	The title of the url
reading_level	Reading difficulty (1-5), estimated by algorithm
uname	Name of the user who submitted the url
email	The email of the user
age	The age of the user
comment	Comment
date	Date that the url was submitted
tag	A set of tags describing the url

Part II

1. Consider the relationships between musical genres. Assume that we have a set of musical genres, {G1, G2, … Gn}, and assume that a genre X can be influenced by one or more other genres and, in turn, genre X will influence one or more genres. So, for example, G1, G2 and G3 might influence X and, in turn, X might influence G4 and G5.

   a) Propose an ER model for this scenario.

   b) Create an SQL script for implementing a schema for this model in PostgreSQL. (Load some sample data into the database tables.)

   c) Write SQL queries for showing the ancestors, descendents, and siblings of a genre.

d) Implement a prototype UI in either PHP or JSP that enables people to navigate backward and forward in time.

2. Consider a **directory** of items with the following information and links for navigating:

Page:
Topic Title: <string>
Topic Path: <grandparent>, <parent>, … <current_topic>
Subtopics: <link_1>, <link_2>, … <link_n>
See Also: <link_1>, <link_2>, … <link_n>
Items: <Item_1>, <Item_2>, … <Item_n>

When you click on a link you move to a different page of the directory. As you can see, each page shows a Topic Title; a Topic Path, which shows the location of the page in the directory; a list of subtopics; and a list of See Also links that cut across the hierarchical structure of the directory. Finally, each page contains a list of items. Note that this structure is quite similar to www.dmoz.org.

a) Propose an ER model for this scenario.

b) Discuss the similarities between this problem, I.4, and II.1.

c) Propose a single conceptual model that could be used to implement solutions to this problem, I.4, and II.1. Reflect on your solution – will it work? Please comment.

Deliverable

Please hand in your solutions to Part I at the beginning of Monday's class (week #2).

Please hand in your solutions to Part II at the beginning of Monday's class (week #3). You do not need to provide code (part II.d) but please include a link to your prototype.

Spring 2007
A2: Performance Monitoring and Tuning (04 APR 07)
Worth: 10%

Note: Consider doing this assignment in groups of two!

Learning objectives (from Syllabus)

1. Describe how a database performance can be monitored and improved;

2. Discuss and apply approaches for improving a database's performance, including denormalization, use of indexes, and data partitioning.

Part I

1. In a create table statement it may be desirable to specific a primary key plus one or more candidate keys. Please explain why. How is this achieved in PostgreSQL?

2. Briefly describe three main factors that are used to measure "efficiency". How does one determine which factor is the most important?

3. Concerning physical storage, what is a "page"? How does a page relate to a relation, to a tuple, and to an attribute in a tuple?

4. Briefly discuss some of the considerations of using indexes to improve the performance of a database. Do indexes always improve performance? How does one decide?

5. What is denormalization? When is it appropriate to consider denormalization? What are the downsides and upsides?

Part II

6. i) Consider the tagging system in A1.q5. Propose a data model or revise/use your own for this system (*Hint*: You can do this with five tables, User, Bookmark, Tag, Word, and Item). For this model, describe 2 – 4 major transactions and analyze the costs associated with these transactions.

   Consider tasks such as: a) A bulk insert of new items; b) A bulk insert of new users and their data (create by, say, another system); c) Many hundreds of users adding bookmarks daily; and d) Many thousands of users accessing their bookmarks daily.

   ii) Considering Q6, suppose you wanted to generate a page that contained the following:
   a. All tags used by the a user;
   b. The last 10 bookmarks added by the user (for each bookmark also show all other users that have also bookmarked it).

   Given this page design, identify the transactions that are involved and propose a set of indexes. Carefully explain and justify your answer.

7. The course website contains materials needed for this question, including a data file, a psql script for building a table (called Word), and php script for running queries against the table.

   The data file consists of approximately 230,000 different words. Each word has been assigned a unique id and a random number between 1 and 10.

   Using these materials, perform an investigation of how indexes impact performance. To do this, I suggest you:

   a) Develop a set of hypotheses that you want to test (informed by your reading of the textbook and the PostgreSQL documentation)

   b) Design an "experiment" for testing the hypotheses. To do this, I suggest you create a chart such as this, where the cases reflect different configurations of indices (no index, one index, two indexes, etc.) and the queries reflect different kinds of queries, such as inserts, queries for single tuples, and so on.

	Case #1	**Case #2 ...**
Query 1	t1	t2 ...
Query 2	t3	t4 ...
...	...	...

   Be sure to investigate the affects of different kinds of queries, including inserts, queries that use functions as IN, BETWEEN, LIKE, and so on influence performance on particular indexes? (You may also want to explore the impact of indexes on joins, though this may require more work ...)

   Please describe your investigation, including hypotheses, experimental design, findings, and discussion. In your discussion please answer at least two questions: a) What have you learned about indexes; and b) What are the limitations of your experiment?

Deliverable

Please hand in your solutions to Part I at the beginning of Monday's class (week #3) and the solutions to Part II at the beginning of Monday's class (week #4).

Spring 2007
A3: Transactions (11 APR 07)
Worth: 10%

Note: Consider doing this assignment in groups of two!

Learning objectives (from Syllabus)

1. Discuss concurrency control and describe major kinds of concurrency problems, and discuss database mechanisms for achieving Atomicity, Consistency, Isolation, and Durability;

2. Employ the SQL commands BEGIN, COMMIT, and ROLLBACK to maintain consistency between the database and model of the world.

Part I

1. What is a transaction? In what way is a transaction a special kind of program? What is a unit of work?

2. Describe the lost update problem.

3. Briefly explain ACID.

4. When should a transaction abort?

5. Describe a basic approach to recoverability?

Part II

6. Consider the ITF project and identify an "interesting" unit of work that consists of multiple updates to the model.
    a. Describe this unit of work.
    b. Discuss the issue of "consistency" with respect to this unit of work.
    c. Implement a working solution in SQL using BEGIN, COMMIT, and ROLLBACK appropriately.
    d. Reflect on your solution to the problem, discussing salient issues or remaining questions.

Deliverable

Please hand in your solutions to Part I at the beginning of Monday's class (week #4) and the solutions to Part II at the beginning of Monday's class (week #5).

Spring 2007

A4: Object Database Management Systems (18 APR 07)

Worth: 10%

Learning objectives (from Syllabus)

1. Describe the major features of Object-Oriented Database Management Systems and discuss when OODBMS are appropriate.

Part I

1. Based on your reading, do believe that ITF project is better suited for implementation in an object-oriented database than a relational database? Please carefully explain your answer using the key concepts of object-oriented databases.

2. Define **impedance mismatch** and discuss a place within the ITF project where this problem obviously crops up. How might we better deal with it?

3. Briefly discuss the difference between **behavioral** and **structural** modeling.

4. What is a persistent **programming language**?

5. When data is stored as a tree with, for example, the tables itf.Item and itf.Link, we often need to recursively visit some part of the tree. Consider the ITF framework and this problem, when answering the following questions:
   a. Describe a user-interface scenario that requires a recursive traversal of the itf.Link table;
   b. Outline an API that could be called to implement a solution to your scenario;
   c. Briefly discuss the strengths and weaknesses of the API.

   <u>Note</u>: Please frame this problem in as general a way as you can.

Part II

6. Using PL/psql, implement a solution to question #5. Please include the code for your solution and briefly discuss your approach and what you learned.

   <u>Note</u>: Please seek to develop a general solution that can be readily used by other programmers for a family of tasks.

7. Use the zope framework (see zop.org) to implement a solution to question #5. Does its object orientation help? Please discuss. (Note: I've not worked with zope and I'm unsure what the learning curve is – if you find it too steep please send e-mail to the listserv.)

Deliverable

Please hand in your solutions to Part I at the beginning of Monday's class (week #5) and the solutions to Part II at the beginning of Monday's class (**<u>week #7</u>**).

Spring 2007
A5: Object- Database Management Systems (18 APR 07)
Worth: 10%

Learning objectives (from Syllabus)

1. Describe the Object-Relational model and the problems it seeks to address

2. Discuss why an Object-Oriented or Object-Relational database might be preferred over a relational database.

Part I

1. Please discuss the major features of the Object-Relational Model.

2. Please discuss the features of PostgreSQL that enable the Object-Relational Model to be implemented.

3. Inspect this article on Object-Relational Mapping: http://en.wikipedia.org/wiki/Object-relational_mapping

   Using examples from the itf framework, please discuss the point of this article.

Part II

No part II this week – please spend time working on your project.

Deliverable

Please hand in your solutions to Part I at the beginning of Monday's class (week #7).

INFO-445: Advanced Database Design, Management, and Maintenance
Spring 2007
A6: XML/DB Interation (09 May 07)
Worth: 10%

Learning objectives (from Syllabus)

1. Create and query simple XML documents.
2. Discuss the major approaches for integrating XML and design database applications that use both the relational and semi-structured data models.

Part I

1. Briefly explain the difference between structured and semi-structured data.
2. Describe two basic ways for processing XML documents – what do you see as the costs and benefits of each approach.
3. Briefly describe XSLT. Is it most useful as a client-side or server-side technology?
4. Briefly describe XPath. What is the difference between a declarative query language and a navigational language?
5. Briefly describe XML Schema. Comment on the complexity of XML Schema, giving an example. Do think the complexity is warranted? Briefly explain your answer.
6. Briefly discuss two approaches for harmonizing XML document representations and relational databases. How do these approaches compare?
7. Briefly discuss the approach that Oracle has taken for integrating XML documents and relational databases. What do you think of the approach?

Part II

1. Write a small, exploratory application, consisting of the following:
   a. An XML schema or DTD.
   b. An import function that shreds XML documents and puts them into one or more relations.
   c. An export function the queries one or more relations and outputs a XML document.
   d. An XSLT transform that presents particular elements of the document in a browser.
2. Briefly comment on each of these components. When, if ever, do you think this approach would be useful for structuring an application?

Deliverable

Please hand in your solutions to Part I at the beginning of Monday's class (week #8). Please hand in your solutions to Part II at the beginning of Monday's class (week #9).

Spring 2007
Project: Information Task Framework (DRAFT 1.0, April 03)
Worth: 30%

Aim

The aim of this project is to design a general data model and application programming interface that can be used to build a family of "Information Task" applications, which are applications that allow to submit, organize, browse, and search items of information.

Examples of such applications include: private and community blogs, portals, resource pages, directories, social tagging systems, bug trackers, message boards, help systems, wikis, forums, and many others. In the future, it seems possible that many systems will be built that draw upon elements that underlie these and similar applications. This project seeks to investigate the technical issues that underpin such a possibility.

Therefore, the BIG question we need to answer in this project is: What set of basic building blocks are required so that it is possible to implement a wide variety of different kinds of applications?

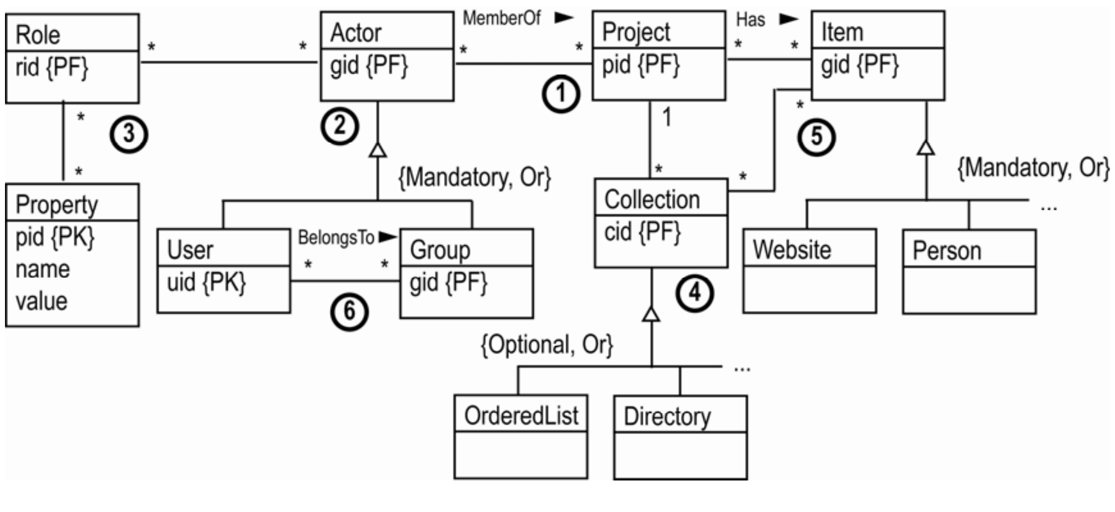
Approach

The basic approach will be to develop a three-tier application. We shall design a single conceptual model and translate the conceptual model into a relational database (PostgreSQL).

One group will build the middle tier in PHP and a second group with build it in Java. In both cases we shall aim to carefully isolate calls to the database so that I will be possible to switch out the database.

Finally, we will build the front-end in JSP and PHP.

DRAFT conceptual model



This conceptual model is an initial draft. The entity Project (1) is used to represent all the resources that are required for a project. Projects are made up of Actors (2), which, depending on their roles (3), have permissions to do certain kinds of work. Actors can either be single users or groups of users (6). In addition, a project consists of one or more collections of items, which can also have specialized sub-types (4). Finally, in this draft, collections are made up of items, which come in various types (5).

Application development

Presumably the above conceptual model can be used to build a wide variety of applications; that is, the model is quite expressive. What should we be able to easily build with this model?

1. Bibliography of websites
2. Personal and community blog
3. Glossary
4. Resource page
5. Directory
6. Wiki
7. Message Board
8. Peer-review bibliography

Are these genres doable? What's missing? Can you imagine novel hybrids or info-hydras?

Extensions

The above model gives us a base. What else could be added to it? Here are some suggestions:

1. Tagging/annotating to allow people to associate keywords, comments, votes etc. with Items;
2. Expressing hierarchical structures to allow people to create hierarchies of various things, especially collections;
3. Facet structures to allow people to create, browse, and search by facets;
4. Workflow to allow people to specific rules for how items are allowed to be selected.

Project logistics

Loosely following open source practices we shall:

1. Propose, design, and implement our own projects;
2. Peer-review each others work;
3. Test each others systems and provide feedback;
4. Seek to reuse each others work.

We shall try to allocate some time in all class and labs to work on this project and wherever possible we shall situate our conceptual learning within the context of the project.

The proposal (due, April 19th) should contain the following:

1. A description and sketch of the application(s) that you will build;

2. A conceptual model (and description) showing how you will extend the base model;

3. An implementation plan that outlines how you will implement your application, especially how you will extend the API in a general way;

4. A thorough discussion of the key issues that you will confront and how you plan to deal with them. (See ongoing issues below.)

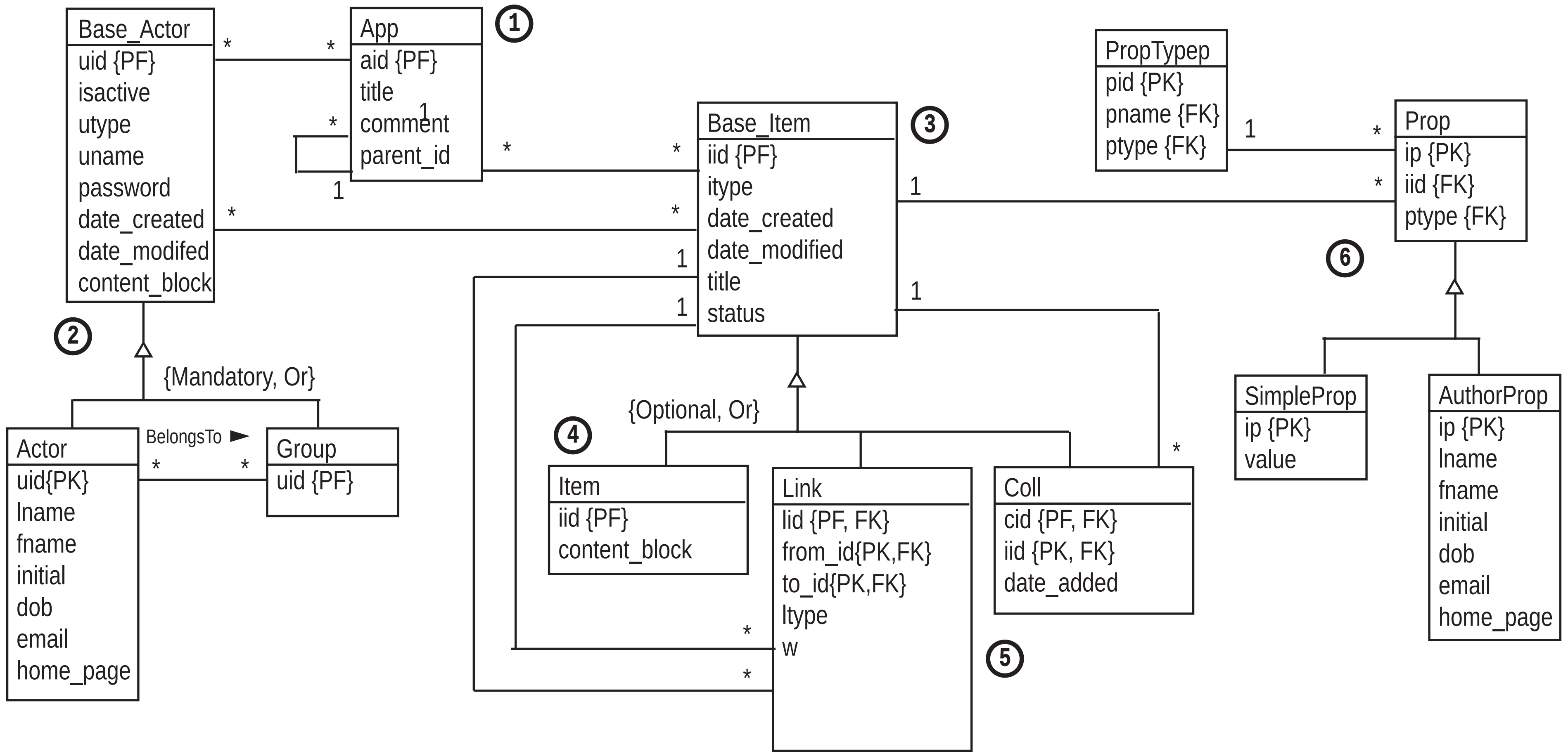
The Draft report (due, May 9th) will fully describe your project and contain an appendix where you critically reflect on your work.

As stated in the syllabus, the final report is due June 1, and presentation will be given on week 9 – 10. You may choose to work alone or in small groups.

Ongoing questions

1. **Abstraction**. How general can we make the framework (data model and API)?
2. **Data integrity**. Does the framework keep the data safe?
3. **Modularity**. Can we create a clean separation between the back-end, the middle tier, and the front end? Are we able to obtain good logical and physical data independence?
4. **Reuse**. Does the framework make application building easier?
5. **Concurrency**. Can the framework handle many concurrent updates and accesses?
6. **Scale**. How well does the framework scale?
7. **XML vs. relational model**. How does XML and the relational model compare? Do they complement each other? Where or where not?

Do you have others to add?



Conceptual Model (04/23/07)