Term Projects

As a term project for this class, you may choose a term-paper project or an implementation project. You may work alone or choose to partner with another student in the class.

Timetable

The timetable for the term project is as follows.

- Find a partner (if you decide to form a team), select a tentative topic, and notify me via email no later than Tuesday October 25, 2011.
- Discuss with me your proposal during office hours or after class, finalize the selection of the topic, and agree on the material that you will read by Tuesday November 1, 2011.
- Submit your report by the last day of classes, which is Thursday December 1, 2011.
- Give a 20-minute presentation on your project on Friday December 2, 2011.

Term-Paper Projects

Guidelines

The expectation is that you will accomplish the following:

- Study a small number of papers or a doctoral thesis on a particular topic related to the principles of database systems.
- Write a report (five to ten pages) in which you will summarize and comment on the papers you read, and also perhaps fill some missing details in these papers (this will depend on the papers you choose to read; for example, conference proceedings papers tend to be terse and leave out details).
- As stated above, give a short presentation of your project on Friday December 2, 2011 at a to-be-determined time.

List of Possible Topics for a Term-Paper Project

Tractable Cases of Conjunctive Query Evaluation

Conjunctive queries (also known as select-project-join queries) are the most frequently asked queries against relational databases. Since the combined complexity of conjunctive-query evaluation is NP-complete, a major research effort has been undertaken towards identifying polynomial-time cases of the conjunctive-query evaluation problem. This area has benefitted from numerous interactions with constraint satisfaction, graph theory, logic, and other areas.

The goal of a project on this topic is to become familiar with this area by learning about some of the techniques and the results obtained thus far. One of the early influential papers is “Algorithms for Acyclic Database Schemes” by M. Yannakakis in VLDB 1981. This had led to an extensive investigation of generalizations of acyclic queries, including various notions of queries of bounded width (see, for example the web page Hypertree Decomposition at http://wwwinfo.deis.unical.it/ frank/Hypertrees/). If you decide to embark on this project,
please talk to me first so that I can provide a more detailed guide to the literature and help you selecting which papers to read.

Note: This project assumes some familiarity with graph algorithms and the basics of computational complexity.

**Query Languages with Recursion** Since relational algebra and relational calculus cannot express recursive queries (e.g., the transitive closure query), researchers have studied extensions of these formalisms with mechanisms that embody recursion. One such query language is Datalog, which will be covered in some detail in class.

The goal of a project on this topic is to become familiar with query languages richer than Datalog, such as various extensions of Datalog with negation, least fixed-point logic, and query languages with the WHILE construct. A starting point would be to read Chapters 14, 15, and 17 in the “Foundations of Databases” textbook.

**Disclosure and Confidentiality in Data Exchange** One of the most challenging issues in managing heterogeneous and distributed data is the issue of inadvertent disclosure of sensitive information when data are exchanged and shared. This is an area of growing importance in the interface of database systems research with cryptography, access control, and other areas. The goal of a project on this topic would be to become familiar with the main technical challenges in this area, the modeling of these challenges, and the proposed solutions. A good reference is the Ph.D. thesis of Gerome Miklau on “Confidentiality and Integrity in Distributed Data Exchange”, which won the 2006 SIGMOD Jim Gray Doctoral Dissertation Award.

**Declarative Query Languages for Network Protocols** In recent years, a variant of the Datalog query language has been successfully used as a declarative language for specifying network protocols. A good reference is the Ph.D. thesis of Boon Thau Loo on “The Design and Implementation of Declarative Networks”, which won the 2007 SIGMOD Jim Gray Doctoral Dissertation Award. A high-level introduction to this body of work can be found in the paper “Declarative Networks”, which appeared in the November 2009 issue of the *Communications of the ACM*. Another good introduction is the paper “Datalog and emerging applications: an interactive tutorial” by Shan Shan Huang, Todd J. Green, and Boon Thau Loo in the proceedings of the 2011 ACM SIGMOD Conference (the paper is based on a tutorial these three authors gave at the conference).

The goal of a project on this topic would be to become familiar with the main developments in this area of research by developing an understanding of the uses of Datalog and its variants in network design and specification.

Datalog has also found recent applications to program analysis, authentication and access control, and computing in a MapReduce environment. An alternative project on Datalog could center around one or more of these applications.

**Query Answering in Inconsistent Databases** An inconsistent database is a database that violates certain given integrity constraints, such as keys or some other functional dependencies. In recent years, much effort has been devoted to studying notions of “repairs” of inconsistent databases and also the semantics and the complexity of query answering in the context of inconsistent databases.
The goal of a project on this topic would be to become familiar with the main concepts, algorithms, and computational complexity results in this area. A starting point would be the paper “Consistent Query Answers in Inconsistent Databases” by M. Arenas, L. Bertossi, and J. Chomicki from PODS 1999, the chapter “Query Answering in Inconsistent Databases” by L. Bertossi and J. Chomicki, the Ph.D. thesis of Ariel Fuxman on “Efficient Query Processing Over Inconsistent Databases”, which won the 2008 SIGMOD Jim Gray Doctoral Dissertation Award, or the 2007 Ph.D. thesis of Slawomir Staworko on “Declarative Inconsistency Handling in Relational and Semi-Structured Databases”.

Provenance in Databases Provenance (also known as lineage) in databases is the area of research concerned with developing an understanding of how a particular piece of data has been derived in a data set. In the past decade, different types of provenance have been investigated in considerable depth. The goal of this project would be to become familiar with the main concepts and results about provenance in databases. A good starting point is the long survey on “Provenance in Databases: Why, How, and Where”, by James Cheney, Laura Chiticariu and Wang-Chiew Tan, in Foundations and Trends in Databases: Vol. 1: No 4, pp 379-474, 2009.

Implementation Projects

Guidelines The expectation is that you will accomplish the following:

- You will fully implement one of the projects described below producing working and correct code.
- You will submit your code and also write a report explaining the design choices you made and giving the user sufficient information to use your tool.
- Give a short presentation and demo of your project in class on Friday December 2, 2011. at a to-be-determined time.

List of Possible Implementation Projects

From Relational Algebra to Relational Calculus and SQL In this project, you are to implement:

1. A translation of relational algebra to relational calculus, and
2. A translation of relational algebra to SQL.

You may choose the input format (for relational algebra expressions) and the output format (for relational calculus expressions and for SQL expressions), but you should include a clear description of the format in your report - think of your report as a user’s manual for your tool.

Additional requirements:
- In this translation, the relational schema used is part of the input. So, the tool should ask first for the description of the relational schema used.
Your tool should include a parser that first checks relational algebra expressions for correctness, and rejects them if they are ill-formed (for example, the parser should reject an expressions that attempts to form the union of two relations of different arities).

You should ensure that your translation to relational calculus produces domain independent relational calculus expressions.

Your report should include a number of non-trivial test cases that include combinations of the five basic relational algebra operations.

From Relational Calculus to Relational Algebra In this project, you are to implement a translation of relational calculus to relational algebra.

You may choose the input format (for relational algebra expressions) and the output format (for relational algebra expressions), but you should include a clear description of the format in your report - think of your report as a user’s manual for your tool.

Additional requirements:

- In this translation, the relational schema used is part of the input. So, the tool should ask first for the description of the relational schema used.
- Your tool should include a parser that first checks relational calculus expressions for correctness, and rejects them if they are ill-formed.
- You should ensure that your translation to relational calculus produces domain independent relational calculus expressions.
- Your report should include a number of non-trivial test cases of relational calculus expressions. In particular, your report should include test cases of relational calculus expressions that involve at least two alternations of quantifiers (think, for example, of the relational calculus expression for the quotient as a starting point).

Generalized Quantifiers in SQL As discussed in class, SQL supports quantification using the constructs EXISTS and NOT EXISTS, but it does not directly support other quantifiers, such that the universal quantifier FOR ALL or counting quantifiers, such at “AT LEAST THREE” or “AT MOST FIVE”.

In this project, you are to augment SQL with generalized quantifiers that make programming in SQL easier and more intuitive. There has been already a fair amount of work in this direction, including an ICDE 1995 paper on “Improving SQL with Generalized Quantifiers” by P.-Y. Hsu and D.S. Parker, Jr., and a 1997 doctoral dissertation by A. Badia’s on “Quantifiers in Action: Generalized Quantification in Query, Logical and Natural Languages”.

This is an open-ended project. Should you decide to work on it, we will need to discuss and agree on the particular direction and scope that it will take.

Design your own implementation project Other implementation projects could also be considered, provided that they related to a topic covered in the course. Please submit an outline for the project you have in mind before meeting with me to discuss its suitability.