CSC 195 2014S, Class 12: Relational Databases

Overview

- Preliminaries.
  - Admin.
  - Questions.
- Basic concepts: Databases, Database management system, Relational database.
- A sample relational database.
- Core database operations.
- Database design.
- An exercise in database design.

Preliminaries

Admin

- If you have not registered for 213, make sure to do so by the 4 pm today.
- This week, we look broadly at relational databases. Next week, we'll move on to SQL.
- Next homework: Finish today's database design. Try to express in SQL.
- It may be fun to skim your classmates' homework, but not in class.

Questions

Basic concepts: Databases, Database management system, Relational database

What is a database?

- "A base of data"?
- A collection of values
- What distinguishes this from lists, arrays, etc?
  - Perhaps organization
  - Databases often store compound data (multiple fields per entry)
  - Designed to support fast and complex searching - Some separation of interface from implementation
  - We often think of lists, arrays, etc. as being in memory, while we assume databases are external to our program and need to be accessed remotely.
  - Databases are persistent
  - Databases often have multiple collections

- Early computing days - Careful arrangement of data on disk or memory to achieve goals
  - Database management systems - Programs that make it easier to make databases and achieve the goals
- Relational databases - Collections of tables ; table is a relation
  - Develop/design by E.F. Codd
- There is a growing movement to explore and use other kinds of databases (e.g., think about the world as a collection of objects)
- SQL is the standard language for working with relational databases
  - Slightly different design for each, particularly in the additional programming part
  - Not imperative, not object-oriented, not functional; it's declarative
  - Declare the character of the data you want, the DBMS figures out the algorithm for getting it.

A sample relational database

Suppose we want to represent information on courses at Grinnell College

- Table for courses
  - Unique identifier: 21512
  - Department (abbreviation): "CSC"
  - Course number: 195
  - Section: "01"
  - Instructor: 32
- Table for instructors
  - ID: 32
  - LName; Rebelsky
  - FName: Samuel
  - Email: rebelsky@grinnell.edu
  - Instructor Office Number: x4410
  - ...
- Table for students
  - ID: 45232
  - LName: Smith
  - FName: Smith
  - Email: SmithSmith@grinnell.microsoft.com
  - BoxNumber:
  - ...

How do we keep track of what courses a student is in?

- Option 1: Have a list of courses in each student record
- Option 2: Have a list of students in each course record
- Option 3: Have a table of course/student pairs
- Option 4: Have fields for each course in a student record
  - Semester 1, year 1, course 1:
  - Semester 1, year 1, course 2:
  - Semester 1, year 1, course 3:
  - Semester 1, year 1, course 4:
  - Semester 1, year 1, course 5:
  - Semester 1, year 1, course 6:
  - Semester 2, year 1, course 1:
  - Semester 2, year 1, course 2:
  - Semester 2, year 1, course 3:
  - Semester 2, year 1, course 4:

Option 3 is preferred * Course id: 21512 * Student id: 43452

SELECT (Student.LName,Student.FName) from (Courses,Students,Option3) where (Courses.DEPT="CSC" and Courses.Number=195, and ...)

Key idea: Tables that join information together

Core database operations

- Extract information
  - Pick one or more rows of a table based on some criteria
  - Extract information from those rows
  - Combine duplicate information
  - Join Combine tables into an implicit table
  - Given a table, extract one column
  - Rearrange results based on some criterion (sort)
  - Pretty print using style sheet
- Add/change information in the database
  - Delete one or more rows/entry (and maybe associated information) REMOVE where Student.Attitude = Snarky REMOVE where Student.ThinksThat = "PHP means Ponies Help People"
  - Change one or more rows/entry (e.g., time changes for a course) UPDATE (Student.Grade = Pass) where CourseId=21512
  - Add a new row/entry
  - Add a new table
  - Add a new column from a table - Painful; good early design helps
  - Remove a column from a table
  - Remove a table
  - Give hints to the DBMS - E.g., "I’m going to query this database by *this field* a lot."

- Other administrative tasks
  - Create a new database
  - Remove an existing database
  - Grant and remove privileges - E.g., User 3123 can read this database, but not modify it, Zoe can read and modify the database, Ajuna and Toby can read, modify, and grant privs to other users.
  - Add and remove users
  - Export / import
  - Backup!
  - Combine two databases

Database design

Database design

- What tables you have
- What types the fields in each table have
- Relationships between tables (which are themselves tables)
- What things you identify as keys (likely to be used for queries)
- ...

What makes a good database?

- Performance for expected queries
  - Note: We should have use cases when we design a database.
- Don't repeat data
- Naming that will make sense to the user/client

An exercise in database design

- What tables would you want for CDThing?
- Homework