Relational Databases

Motivation
Relational Model
MySQL
CIS 170 Projects
Reading: Ch 65 of NTO

Databases

- Informally we can think of a database as a large collection of data
- But a database is more than "a really big file"
  - the *Yersinia pestis* genome was used to demonstrate string search
  - It's a big file (11MB) but is not a "database"
  - In CS jargon it's known as a "flat file"
- So what makes a large collection of data a "database"?
  - A database has additional information used to organize and retrieve the data
  - I.e. a database has a set of index structures
- Examples:
  - PubMed, Genbank, and other databases at NCBI -- index by sequence ID
  - Google -- index by word
  - Customer records at banks, phone companies, etc -- index by customer ID

History

- The idea of automatically organizing and accessing data predates the modern computer era
- Herman Hollerith designed a machine for the 1890 census
  - Operators punched holes in cards to record data
  - A "tabulating machine" counted entries in various categories
  - Founded CTR, Inc ("computing-tabulating-recording")
- Thomas J. Watson joined in 1914
  - Renamed the company IBM in 1924

Databases Today

- Businesses and other organizations have an insatiable appetite for information
- Research on databases is a major area of computer science
- Companies that produce database management systems are among the biggest software companies
  - Oracle, Informix, Sybase, ...
  - Database products are a large part of the business for Microsoft, IBM, ...
- Courses at UO
  - For CIS majors:
    - CIS 451 Database Processing
    - CIS 452 Database Issues
  - For CIT (Computer Information Technology) minor:
    - CIT 381 Database Systems

www.computerhistory.org
Database Management Systems

- A database management system (DBMS) is a set of applications for organizing and accessing information
  - client programs provide the interface to the system
  - clients can connect to servers via the internet
  - similar to the way browsers connect to web servers
- Users access information by submitting a query
- An administrator creates user accounts and manages the system

Types of Databases

- There are three main ways of organizing databases
  - Record oriented
    - assign each entry a key, build an index to hold keys
  - Object databases
    - long-term storage for objects (Arrays, Hashes, ...)
    - an object can have a reference to other objects in the database
  - Relational databases
    - organize data as a collection of tables
    - (the rest of this lecture explains how/why)
- NTO says the three types of databases are network, hierarchical, relational
  - the first two faded away in the early 1980s...

Relations

- In math we think of a “relation” as an expression that compares numbers
  - \( a = b \)
  - \( a < 5 \)
- Relations, like functions, can be displayed on a graph

Table

- One way to define a function is to describe it as a set, e.g.
  \[ f(x) = x/2 \]
  \[ f = \{ (x, y) : y = x/2 \} \]
- Here \((x,y)\) means “the pair of numbers \(x\) and \(y\)”
- The pairs are often shown in a table:

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
</tr>
<tr>
<td>(\text{etc})</td>
<td></td>
</tr>
</tbody>
</table>

A continuous function has an infinite number of rows...
Tables (cont’d)

- The same idea applies to relations, e.g. for $y < x / 2$

$$r = \{(x, y) : y < x/2\}$$

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>0.49</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Tables for relations might also have an infinite number of rows.

Types of Columns

- One thing that distinguishes a function from a relation is that for any one input value there is at most one output value
  - input column: domain
  - output column: range

- Relations do not put any restrictions on the roles for columns

function: $y = x / 2$

relation: $y < x / 2$

Another Example

- We saw an example of a function specified as a collection of possible input/output values in an earlier lecture

$$\delta: \begin{align*}
\delta(0\epsilon,N) &= 5\epsilon \\
\delta(5\epsilon,N) &= 10\epsilon \\
\delta(5\epsilon,D) &= 15\epsilon \\
\delta(10\epsilon,N) &= 15\epsilon \\
\delta(10\epsilon,D) &= 20\epsilon \\
\text{etc}
\end{align*}$$

Another Example (cont’d)

- This function can also be displayed as a table:

<table>
<thead>
<tr>
<th>current state</th>
<th>input symbol</th>
<th>next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0\epsilon</td>
<td>N</td>
<td>5\epsilon</td>
</tr>
<tr>
<td>0\epsilon</td>
<td>D</td>
<td>10\epsilon</td>
</tr>
<tr>
<td>0\epsilon</td>
<td>Q</td>
<td>25\epsilon</td>
</tr>
<tr>
<td>5\epsilon</td>
<td>N</td>
<td>10\epsilon</td>
</tr>
<tr>
<td>5\epsilon</td>
<td>D</td>
<td>15\epsilon</td>
</tr>
<tr>
<td>10\epsilon</td>
<td>N</td>
<td>15\epsilon</td>
</tr>
<tr>
<td>10\epsilon</td>
<td>D</td>
<td>20\epsilon</td>
</tr>
</tbody>
</table>

etc
Relational Algebra

- By thinking of relations as tables it is possible to define a relational algebra as a set of operations on tables.
- Algebra for numbers:
  - letters \( x, y, z \), etc represent numbers
  - operations (+, etc) combine numbers to produce new numbers
  - properties (associative, commutative, etc) allow us to manipulate symbols
    \[ x \cdot (y + z) = x \cdot y + x \cdot z \]

- Algebra for relations:
  - letters represent tables
  - operations (\( \cup, \propto, \sigma, \text{etc} \)) create new tables
  - it is also possible to solve equations made up of symbols and operators
    \[(r \cup ir) \propto s = (r \propto s) \cup (ir \propto s)\]

Select Examples

<table>
<thead>
<tr>
<th>current state</th>
<th>input symbol</th>
<th>next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0¢</td>
<td>N</td>
<td>5¢</td>
</tr>
<tr>
<td>0¢</td>
<td>D</td>
<td>10¢</td>
</tr>
<tr>
<td>0¢</td>
<td>Q</td>
<td>25¢</td>
</tr>
<tr>
<td>5¢</td>
<td>N</td>
<td>10¢</td>
</tr>
<tr>
<td>5¢</td>
<td>D</td>
<td>15¢</td>
</tr>
<tr>
<td>10¢</td>
<td>N</td>
<td>15¢</td>
</tr>
<tr>
<td>10¢</td>
<td>D</td>
<td>20¢</td>
</tr>
<tr>
<td>15¢</td>
<td>N</td>
<td>20¢</td>
</tr>
<tr>
<td>15¢</td>
<td>D</td>
<td>25¢</td>
</tr>
<tr>
<td>20¢</td>
<td>N</td>
<td>25¢</td>
</tr>
<tr>
<td>20¢</td>
<td>D</td>
<td>30¢</td>
</tr>
</tbody>
</table>

Select Examples

\[ \sigma \text{ input symbol } = \text{“N”} \]

\[ \sigma \text{ next state } = \text{“25¢”} \]

Note the new tables have the same columns as the original.
Project Examples

<table>
<thead>
<tr>
<th>current state</th>
<th>input symbol</th>
<th>next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0¢</td>
<td>N</td>
<td>5¢</td>
</tr>
<tr>
<td>0¢</td>
<td>D</td>
<td>10¢</td>
</tr>
<tr>
<td>0¢</td>
<td>Q</td>
<td>25¢</td>
</tr>
<tr>
<td>5¢</td>
<td>N</td>
<td>10¢</td>
</tr>
<tr>
<td>5¢</td>
<td>D</td>
<td>15¢</td>
</tr>
<tr>
<td>10¢</td>
<td>N</td>
<td>15¢</td>
</tr>
<tr>
<td>10¢</td>
<td>D</td>
<td>20¢</td>
</tr>
<tr>
<td>15¢</td>
<td>N</td>
<td>20¢</td>
</tr>
<tr>
<td>15¢</td>
<td>D</td>
<td>25¢</td>
</tr>
<tr>
<td>20¢</td>
<td>N</td>
<td>25¢</td>
</tr>
<tr>
<td>20¢</td>
<td>D</td>
<td>25¢</td>
</tr>
</tbody>
</table>

Note duplicate rows have been removed from the output table.

A relation is a set of rows.

Relational Operator: Join

- The join operator combines two tables (symbol = “bowtie”)
- In set theory, the product of two sets is formed by making all possible combinations:
  \[ \{a, b, c\} \times \{x, y\} = \{(a, x), (a, y), (b, x), (b, y), (c, x), (c, y)\} \]
- To join two tables:
  - pick a column they have in common
  - form all possible combinations from the other columns

Structured Query Language

- A widely used language for relational databases is known as the Structured Query Language (SQL)
  - pronounced “ess queue ell”, “sequel”, or “squill”
  - grew out of an IBM research project on relational databases
  - English-like syntax
  - “high level” view of data
  - programmers/users specify what tables to construct
  - let system figure out how to combine data
- Computer languages that emphasize “what” over “how” are known as “declarative” languages
  - Ruby, Perl, Java, C, C++, .... are “imperative” languages
MySQL

- MySQL is a DBMS that uses SQL for its query language
  - open source, available from www.mysql.com
  - the "com" means it is now a business
  - the software is open, but you can buy training, support, etc
- Can download and install on Windows, OS/X, Linux, ...
  - install client if you just want to connect to servers on other systems
  - can also install server if you want to have a database on your own machine
- One of the “big four” open source projects
  - LAMP = Linux / Apache / MySQL / Perl
  - everything you need for a successful e-business

MySQL Command Line Client

- Using the MySQL client is very much like using Ruby with irb
  - start the program
  - type SQL queries
  - MySQL will execute the query and print the results on your terminal

MySQL Syntax

- MySQL commands are very “English-like”
  - All start with a verb telling the system what you want it to do
  - Examples: “show tables”, “describe X”, “select a, b, c from X”

SQL Query Examples

- The state transition function for the candy machine is in a table named delta
- To see all the records in this table:

```
mysql> select * from delta;
+---------+-------+------+
| current | input | next |
+---------+-------+------+
| 0@      | N     | 5@   |
| 0@      | D     | 10@  |
| 0@      | Q     | 25@  |
| 5@      | N     | 10@  |
| 5@      | D     | 15@  |
| 10@     | N     | 15@  |
| 10@     | D     | 20@  |
| 15@     | N     | 20@  |
| 15@     | D     | 25@  |
| 20@     | N     | 25@  |
| 20@     | D     | 30@  |
+---------+-------+------+
```

(I couldn’t figure out how to put a ¢ in the table so I used @ instead)
SQL Query Examples

- SQL does a lot more than basic relational algebra
- To answer the query shown below, MySQL will
  - sort the `delta` table by `input`
  - count the number of rows in each group
  - the result is a table that shows how often each symbol from the `input` column (N, D, Q) is used in the state transition function:

```
mysql> select input, count(input) as n from delta group by input;
+-------+---+
| input | n |
+-------+---+
| D     | 5 |
| N     | 5 |
| Q     | 1 |
+-------+---+
3 rows in set (0.03 sec)
```

Joins Are Done Automatically

- SQL does not have a special symbol to specify a join
- To make a table that combines information from two or more tables just write a `select` statement that lists the tables in the `from` part of the statement:

```
mysql> select taxon, genus, accession from taxonomy, dna where taxonomy.species_id = dna.species_id limit 6;
+--------+---------------+-----------+
| taxon  | genus         | accession |
+--------+---------------+-----------+
| 62977  | Acinetobacter | NC_005966 |
| 272557 | Aeropyrum     | NC_000854 |
| 176299 | Agrobacterium | NC_003062 |
| 176299 | Agrobacterium | NC_003063 |
| 176299 | Agrobacterium | NC_003064 |
| 176299 | Agrobacterium | NC_003065 |
+--------+---------------+-----------+
```

Schema

- In order to write queries with joins one needs to know how tables are related
- A database schema explains the relationship between tables
  - each box represents one table
  - lines connect boxes if the tables have rows in common

- For each row in `taxonomy` there are one or more rows in `dna`

Schema (cont’d)

- The example join uses the fact that taxonomy and DNA both have a `species_id` column

```
select taxon, genus, accession from taxonomy, dna
where taxonomy.species_id = dna.species_id
```

- Columns selected for output table
Keys

- MySQL and other database systems use index structures to make selects and joins more efficient
  - designate one or more more columns as a key
  - keys in this example are shown in gray
  - a table can have any number of keys

Advantages of Tables

- The ability to combine information from separate tables using a join is one of the advantages of a relational database
- For example, consider an e-business that keeps track of customer orders
  - Customer information could be kept with each order
    - name, address, credit card, items ordered, date, total cost, ...
  - If a customer makes several orders the same information appears over and over again
    - wastes space
    - difficult to maintain
      - e.g. what if a customer’s address changes?

Advantages of Tables (cont’d)

- If the data is stored in a relational database, the information could be split into two or more tables
  - customer info: id, name, address, etc
  - order info: customer id, order date, total, etc
- A report generator can fetch the information it needs to print statements, print business summaries, ...

Computer Science Success Story

- The idea of using relations (concepts from mathematics) to solve important problems in data management is one of the great success stories in computer science
- In the 1960s and 1970s there were many different ways of organizing data, but all had problems and were frustrating and difficult to use
- E. F. Codd, a researcher at IBM, came up with the idea of using relations to describe data
  - originally it was not clear how to implement these ideas efficiently
  - after much research and development relational databases became as efficient as other types of databases
- Codd was the 1981 winner of the ACM’s Turing Award
  - the CS equivalent of the Nobel Prize
Review

- You should be familiar with the basic definitions of relational databases
  - table, column, record, key (index)
- Given the description of the contents of one or more tables, know what the results of the basic operations would produce
  - select, project, join
- Be able to use MySQL queries to fetch information from a database
  - get a list of tables
  - get a description of a table
  - select rows from a single table
  - extra credit: write a query that does a join

MySQL Database for CIS 170

- Lab projects for this class will use a database set up on the server used by my research lab

Connecting to the Database

- There are two ways to use the database
  1. Run the MySQL command line client from your own computer
  2. Use your browser to connect to this page

Connecting to the Database (cont’d)

- prompt means MySQL is ready for a query
- (2) Use your browser to connect to this page
Some Things to Remember

- Terminate queries with semi-colons
- MySQL doesn’t care about upper/lower case
- Know when a column contains string data
  - Called varchar in MySQL
  - Use quotes around string values in queries, e.g.
    \[ SELECT * FROM delta WHERE input = "N"; \]
- Use AND, OR, etc to form complex conditions following WHERE

Important Note

- The CIS 170 database is accessible only from the uoregon domain
  - Only computers with names ending “.uoregon.edu” can access the database
- You can
  1. Connect from a computer in McKenzie 101 or other public lab
  2. Connect from the dorms or other places on the campus network
  3. If you connect via browser, use VPN (virtual private network)
     - Go first to
       \[ https://uo-vpn1-gw.uoregon.edu/webvpn.html \]
     - Then enter
       \[ http://teleost.cs.uoregon.edu/cis170 \]
       in the box that says “Enter Web Address”