First, a story…

UW’s Databases

The Seattle Times

New UW payroll system behind schedule, more costly than expected

Outline

1. Administrivia
2. The Relational Data Model
3. Databases, SQL, and RA

What am I going to learn?

Course Topics
- Queries
- Database Design
- Optimization
- Transactions and Parallelism
- Semi-Structured Document Databases

Tools:
- Experimental to Enterprise Platforms
- Cloud Services (AWS, Microsoft Azure)

What am I going to learn?

After the course, you will be able to…
- Explain how a query is processed end-to-end
- Integrate a database into an application
- Effectively manage data for long-term use
- Create database constructs to provide speedups
- Make design choices when selecting tools for a project

414 Staff

Instructor: Ryan Maas (maas@cs)
- Office hours Tuesday 9:30am in CSE 358 and by appointment

TA’s
- Shana Hutchison
- Matthew Liu
- Anne Pham
- Ben Shmidt
- Ying Wang
- Cong Yan
Course Format

- Lectures: this room, please attend!
- Sections: for locations, see website
  - Bring your laptop
- 8 homework assignments
  - First assignment published on website tomorrow
- Midterm and final (in-class)
- Participation:
  - Post and answer questions (in class, piazza, etc)

Exams

- Midterm (TBD) and Final (Dec 12)
- You may bring letter-size piece of paper with notes
  - Handwritten
  - May write on both sides
  - Midterm: 1 sheet, Final: 2 sheets
- Closed book. No computers, phones, watches,...

References

- Main textbook, available at the bookstore or pdf:
- Also useful:
  - Database Management Systems (3rd Edition)

Administrivia

- Website: cs.washington.edu/414
- Grading:
  - 40% HW, 20% Midterm, 30% Final, 10% between them
  - 4 late days, 2 days max per assignment, used in 24 hour chunks
- Collaboration:
  - Work together or alone on HWs, but type your own solutions and say who you worked with
  - Labs are done individually
Communication

  - Course materials here
- Ed message board
  https://us.edstem.org/courses/136/discussion/
  - The place to ask course-related questions
  - Log in today, enable notifications
- Class mailing list
  - Very low traffic, only important announcements

Let’s get started!

Database

What is a database?

- A collection of files storing related data

Give examples of databases

- Accounts database;
- Payroll database
- UW’s students database
- Amazon’s products database
- Airline reservation database

Database Management System

What is a DBMS?

- A big program written by someone else that allows us to manage efficiently a large database and allows it to persist over long periods of time

Examples of DBMSs

- Oracle, IBM DB2, Microsoft SQL Server, Vertica, Teradata
- Open source: MySQL (Sun/Oracle), PostgreSQL, CouchDB
- Open source library: SQLite

We will focus on relational DBMSs most quarter
Think About This

How do we describe information?

Introduction

19

Think About This

How do we describe information?

19

Think About This

How do we describe information?

20

Think About This

How do we describe information?

21

3 Parts of a Data Model

The 3 parts of any data model

- Instance
  - The actual data
- Schema
  - A description of what data is being stored
- Query Language
  - How to retrieve and manipulate data

22

Data Model Zoo

There are lots of models out there!

- Relational
- Semi-structured
- Key-value pairs
- Graph
- Object-oriented
- …

23

What is the Relational Model?

A Relational Model of Data for Large Shared Data Banks

September 25, 2019

24

A Data Model is a mathematical formalism to describe data. It is how we can talk about data conceptually without having to think about implementation.
Multiple Representation
Same data can be represented in different ways
An example of Facebook friends

Data Model Zoo
There are lots of models out there!
- Relational
- Semi-structured
- Key-value pairs
- Graph
- Object-oriented

Components of the Relational Model
Payroll (Userid, Name, Job, Salary)

Components of the Relational Model
Payroll (Userid, Name, Job, Salary)

Schema, describes data

The Relational Model
Again, how we describe information?
Most common answer: The Relational Model

Ted Codd

https://db-engines.com/en/ranking
Components of the Relational Model

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

Instance of actual data

Characteristics of the Relational Model

- Originally defined with **Set semantics**
  - No duplicate tuples
- Attributes are **typed** and **static**
  - INTEGER, FLOAT, VARCHAR(n), DATETIME, ...
- Tables are **flat**

* I wish
Characteristics of the Relational Model

- Originally defined with **Set semantics**
  - No duplicate tuples
- Attributes are **typed** and **static**
  - INTEGER, FLOAT, VARCHAR(n), DATETIME, ...
- Tables are **flat**

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

Order doesn’t matter

Characteristics of the Relational Model

- Set semantics → not in most DBMS implementations
  - No duplicate tuples
- Attributes are **typed** and **static**
  - INTEGER, FLOAT, VARCHAR(n), DATETIME, ...
- Tables are **flat**

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

Violates set semantics!

Characteristics of the Relational Model

- Set semantics → not in most DBMS implementations
  - No duplicate tuples
- Attributes are **typed** and **static**
  - INTEGER, FLOAT, VARCHAR(n), DATETIME, ...
- Tables are **flat**

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

Violates set semantics!

Characteristics of the Relational Model

- Set semantics → not in most DBMS implementations
  - No duplicate tuples
- Attributes are **typed** and **static**
  - INTEGER, FLOAT, VARCHAR(n), DATETIME, ...
- Tables are **flat**

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

Violates set semantics!

Characteristics of the Relational Model

- Set semantics → not in most DBMS implementations
  - No duplicate tuples
- Attributes are **typed** and **static**
  - INTEGER, FLOAT, VARCHAR(n), DATETIME, ...
- Tables are **flat**

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

Violates set semantics!

But how is this data ACTUALLY stored?

<table>
<thead>
<tr>
<th>Payroll</th>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
<td></td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
<td></td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
<td></td>
</tr>
</tbody>
</table>
Characteristics of the Relational Model

But how is this data ACTUALLY stored?

Don’t know. Don’t care.

Physical Data Independence

September 25, 2019

Introduction

43

Structured Query Language - SQL

Alright, I have data and a schema. How do I access it?

SELECT P.Name, P.UserID
FROM Payroll AS P
WHERE P.Job = 'TA';

SELECT What kind of data I want
P.Name, P.UserID
FROM Payroll AS P
WHERE P.Job = 'TA';

Structured Query Language - SQL

"SQL (standing for Structured Query Language) is the standard language for relational database management systems. When it originated back in the 1970s, the domain-specific language was intended to fulfill the need of conducting a database query that could navigate through a network of pointers to find the desired location. Its application in handling structured data has fostered in the Digital Age. In fact, the powerful database manipulation and definition capabilities of SQL and its intuitive tabular view have become available in some form on virtually every important computer platform in the world.

Some notable features of SQL include the ability to process sets of data as groups instead of individual units, automatic navigation to data, and the use of statements that are complex and powerful individually. Used for a variety of tasks, such as querying data, controlling access to the database and its objects, guaranteeing database consistency, updating rows in a table, and creating, replacing, altering and dropping objects, SQL lets users work with data at the logical level."

https://blog.ansi.org/?p=158690

Key points about SQL:

• A domain-specific language
• SQL only works on relational databases
• Not for general purpose programming (Java, C/C++, …)
• Logical level of interaction with data

Hello World

SELECT P.Name, P.UserID
FROM Payroll AS P
WHERE P.Job = 'TA';

Hello World

Payroll

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

Payroll

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>
SELECT P.Name, P.UserID
FROM Payroll AS P
WHERE P.Job = 'TA';