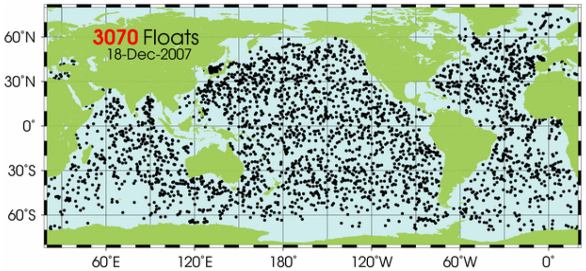
Database Management Systems
CSEP 544

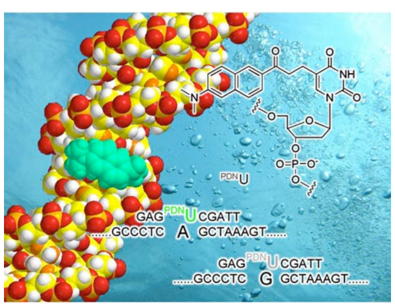
Lecture 1: Introduction

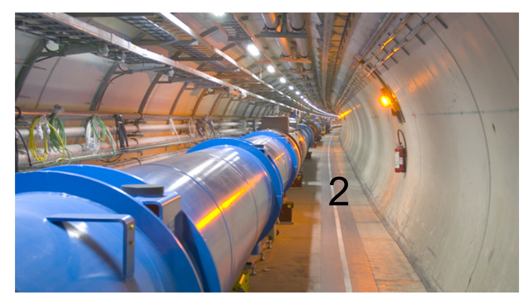
Data Models
SQL



Class Goals

- The world is drowning in data!
- Need computer scientists to help manage this data
  - Help domain scientists achieve new discoveries
  - Help companies provide better services (e.g., Facebook)
  - Help governments (and universities!) become more efficient
- Welcome to PMP 544: Database Management Systems
  - Existing tools PLUS data management principles
  - This is not just a class on SQL/Spark/Datalog!

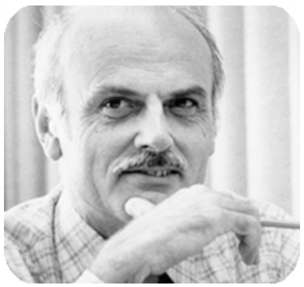




Turing Awards in Data Management

Charles Bachman, 1973

IDS and CODASYL



Ted Codd, 1981

Relational model

You could be next!!

Jim Gray, 1998

Transaction processing

Michael Stonebraker, 2014

INGRES and Postgres

Staff

- Instructor: Alvin Cheung
  - OH: Fridays, 4:30-5:30pm in CSE 530

- TA: Nick Anderson
  - OH: Tuesdays, 5:30-6:30pm in CSE 021
- TA: Cindy Suripto
  - OH: Mondays, 5:30-6:30pm in CSE 007

Course Format

- Website: http://cs.washington.edu/csep544
- 9 Lectures
  - Location: here! (videos posted online)
- Course communication: http://piazza.com/washington/fall2017/csep544
  - Please sign up!
- 8 homework assignments
  - One assignment each week
- 7 paper reviews
  - One paper each week on recent or classical topics in data management
  - Answer a few questions about them
- Take home final exam
  - You have 24 hours to complete the exam
  - Currently scheduled for 12/11

Grading

• Homeworks	40%
• Paper reviews	20%
• Final	30%
• Class participation	10%

- This is all subject to change

Communications

- Web page: http://cs.washington.edu/csep544
  - Syllabus is there
  - Lectures (and videos) will be available there (see calendar)
  - Homework assignments will be available there
- Piazza
  - Make sure you sign up: http://piazza.com/washington/fall2017/csep544
  - **THE** place to ask course-related questions
  - Log in today and enable notifications
  - Course staff will go through questions twice each day

Textbook

Main textbook, available at the bookstore:

- *Database Systems: The Complete Book*,
  Hector Garcia-Molina,
  Jeffrey Ullman,
  Jennifer Widom

  Second edition.

Textbook (and others) are REQUIRED READING !

Most important: COME TO CLASS ! ASK QUESTIONS !

Other Texts

Available at the Engineering Library
(some on reserve):

- *Database Management Systems*, Ramakrishnan
- *Fundamentals of Database Systems*, Elmasri, Navathe
- *Foundations of Databases*, Abiteboul, Hull, Vianu
- *Data on the Web,* Abiteboul, Buneman, Suciu

Nine Lectures

10/3 L1: Introduction and Data Models

10/10 L2: Relational Data Model and SQL

10/17 L3: SQL and Datalog

10/24 L4: Non-relational Data Models

10/31 L5: Physical Design and Query Optimization

11/7 L6: Distributed Query Processing: Parallel DBMS

11/9 L7: Distributed Query Processing: Spark

11/14 L8: Design Theory

11/21 L9: Transactions and Recovery

Eight Homework Assignments

H1&H2: Basic SQL with SQLite

H3: Advanced SQL with SQL Server

H4: Datalog and Relational Algebra

H5: NoSQL

H6: Spark with AWS

H7: Schema Design

H8: Transactional Application

Check calendar for due dates -- Submit via git!

About the Assignments

- Homework assignments will take time but most time should be spent *learning*
- Do them on your own
- Very practical assignments
- Put everything on your resume!!!
  - SQL, SQLite, SQL Server, SQL Azure JDBC, JSon, Spark, AWS, Datalog, LogicBlox…

Deadlines and Late Days

- Assignments are expected to be done on time, but things happen, so…
- You have up to 4 late days
  - No more than 2 on any one assignment
  - Use in 24-hour chunks
- Late days = safety net, not convenience!
  - You should not plan on using them
  - If you use all 4 you are doing it wrong

Exams

- Take home final
  - Currently scheduled for Monday 12/11
  - Details TBD

Academic Integrity

- Anything you submit for credit is expected to be your own work
  - Of course OK to exchange ideas, but not detailed solutions
  - We all know difference between collaboration and cheating
  - Attempt to gain credit for work you did not do is misconduct
- I trust you implicitly, but will come down hard on any violations of that trust

Lecture Notes

- Will be available before class online
- Feel free to bring them to class to take notes

Now onto the real stuff…

Outline of Today’s Lecture

- Overview of database management systems
  - Why they are helpful
  - What are some of their key features
  - What are some of their key concepts

- Course content

Database

What is a database ?

Database

What is a database ?

- A collection of files storing related data

Give examples of databases

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 ?

Give examples of DBMSs

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*

Give 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

An Example: Online Bookseller

- What data do we need?
  -
  -
  -
  -
- What capabilities on the data do we need?
  -
  -
  -

An Example: Online Bookseller

- What data do we need?
  - Data about books, customers, pending orders, order histories, trends, preferences, etc.
  - Data about sessions (clicks, pages, searches)
  - Note: data must be persistent! Outlive application
  - Also note that data is large… won't fit all in memory
- What capabilities on the data do we need?
  -
  -
  -

An Example: Online Bookseller

- What data do we need?
  - Data about books, customers, pending orders, order histories, trends, preferences, etc.
  - Data about sessions (clicks, pages, searches)
  - Note: data must be persistent! Outlive application
  - Also note that data is large… won't fit all in memory
- What capabilities on the data do we need?
  - Insert/remove books, find books by author/title/etc., analyze past order history, recommend books, …
  - Data must be accessed efficiently, by many users
  - Data must be safe from failures and malicious users

Discussion

- Did you ever encounter a data management problem?
  - Experimental data from a homework?
  - Personal data?
  - Other data?
- How did you manage your data?
- Lesson: Need to learn how to *model* data

Using Databases

- Jane and John both have ID number for gift certificate (credit) of $200 they got as a wedding gift
  - Jane @ her office orders "The Selfish Gene, R. Dawkins" ($80)
  - John @ his office orders "Guns and Steel, J. Diamond" ($100)

- Questions:
  - What is the ending credit?
  - What if second book costs $130?
  - What if system crashes?

- Lesson: A DBMS needs to handle various user issues!

So what functions should a DBMS provide?

1. Describe real-world entities in terms of stored data
2. Persistently store large datasets
3. Efficiently query & update
   – Must handle complex questions about data
   – Must handle sophisticated updates
   – Performance matters
4. Change structure (e.g., add attributes)
5. Concurrency control: enable simultaneous updates
6. Crash recovery
7. Security and integrity

DBMS Benefits

- Expensive to implement all these features inside the application
- DBMS provides these features (and more)
- DBMS simplifies application development

Key Data Management Concepts

- **Data models**: how to describe real-world data
  - Relational, NoSQL, Distributed …
- **Declarative query languages**
  - Say what you want not how to get it
- **Data independence**
  - Physical independence: can change how data is stored on disk without maintenance to applications
  - Logical independence: can change schema w/o affecting apps
- **Query optimizer**
  - Query plans and how they are executed
- **Physical design**
- **Transactions**
  - isolation and atomicity

Review this slide during the quarter!

What is this class about?

- Data models
  - Relational: SQL and Datalog
  - NoSQL: SQL++
- RDMBS internals
  - Relational algebra
  - Query optimization and physical design
- Parallel query processing
  - Spark and Hadoop
- Conceptual design
  - E/R diagrams
  - Schema normalization
- Transactions
  - Locking and schedules
  - Writing DB applications

Data models

Query Processing

Using DBMS

Who are the players?

- **DB application developer**: writes programs that query and modify data (this class)
- **DB designer**: establishes schema (this class)
- **DB administrator**: loads data, tunes system, keeps whole thing running (this class, 444)
- **Data analyst**: data mining, data integration (this class, 446)
- **DBMS implementor**: builds the DBMS (444)

What to do now

- Go to gitlab.cs.washington.edu and create your gitlab account
- Go to http://bit.do/544final to fill out your final exam date preference (default date: 12/11)
- Go to http://bit.do/hw3 after you have signed up for a new live.com account

Data Models

Class overview

- Data models
  - Relational: SQL, RA, and Datalog
  - NoSQL: SQL++
- RDMBS internals
  - Query processing and optimization
  - Physical design
- Parallel query processing
  - Spark and Hadoop
- Conceptual design
  - E/R diagrams
  - Schema normalization
- Transactions
  - Locking and schedules
  - Writing DB applications

Data models

Query Processing

Using DBMS

Review

- What is a database?
  – A collection of files storing related data

- What is a DBMS?
  – An application program that allows us to manage efficiently the collection of data files

Data Models

- Suppose we have book data: author, title, publisher, pub date, price, etc
  - How should we organize such data in files?

Data model: a general, conceptual way of structuring data

Data Models

- Relational
  - Data represented as relations
- Semi-structured (JSon)
  - Data represented as trees
- Key-value pairs
  - Used by NoSQL systems
- Graph
- Object-oriented
- We will study the first three in this class

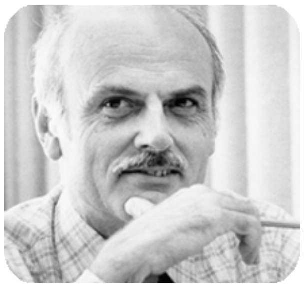
3 Elements of Data Models

- Instance
  - The actual data
- Schema
  - Describe what data is being stored
- Query language
  - How to retrieve and manipulate data

Turing Awards in Data Management

Charles Bachman, 1973

IDS and CODASYL



Ted Codd, 1981

Relational model

Jim Gray, 1998

Transaction processing

Michael Stonebraker, 2014

INGRES and Postgres

Relational Model

columns / attributes / fields

- Data is a collection of relations / tables:

rows / tuples / records

cname	country	no_employees	for_profit
GizmoWorks	USA	20000	True
Canon	Japan	50000	True
Hitachi	Japan	30000	True
HappyCam	Canada	500	False

- mathematically, relation is a set of tuples
  - each tuple appears 0 or 1 times in the table
  - order of the rows is unspecified

The Relational Data Model

- “degree” or “arity” of a relation
  - Number of attributes
- Each attribute has a type.
  - Examples types:
    - Strings: CHAR(20), VARCHAR(50), TEXT
    - Numbers: INT, SMALLINT, FLOAT
    - MONEY, DATETIME, …
    - Few more that are vendor specific
  - Statically and strictly enforced

Keys

- An attribute that uniquely identifies a record
  - Example?

- A key can consist of multiple attributes
  - What does that mean?

Keys

- Key = subset of columns that uniquely identifies tuple
- A relation can have many keys
  - But only one of them can be chosen to be the *primary key*
    - Will see what that means later on this quarter
- Foreign key:
  - An attribute(s) that is a key for other relations

Relation Model: Example

- Instance

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

- Schema

```
Company(name, country, employees, for_profit)

Company(name: varchar(30), country: char(20),
        employees: int, for_profit: char(1))
```

Relational Model: Example

```
Company(cname, country, no_employees, for_profit)

Country(name, population)
```

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

name	population
USA	320M
Japan	127M

Aside: Semi-Structured Model: Example

```
Company(cname, country, no_employees, for_profit)

Country(name, population)
```

- Key-value example:
  - Key: `cname`, Value = `{country, no_employees, for_profit}`
    - What operations can we do efficiently?
  - What about:
    Key: `country`, Value = `{cname, no_employees, for_profit}`

- Can we store this data using a graph?
  - Hint: JSon

Query Language

- SQL
  - **S**tructured **Q**uery **L**anguage
  - Developed by IBM in the 70s
  - Most widely used language to query relational data
- We will see other languages for the relational model later on
  - Datalog, relational algebra, etc.

Our First DBMS

- SQL Lite
- Will switch to SQL Server later in the quarter

Demo

Discussion

- Tables are NOT ordered
  - they are sets or multisets (bags)
- Tables are FLAT
  - No nested attributes
- Tables DO NOT prescribe how they are implemented / stored on disk
  - This is called **physical data independence**

Table Implementation

- How would you implement this?

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

- What if we store this table in a *row major* order?
  - What operations will we be able to do efficiently?
- What if we store it in a *column major* order?

Table Implementation

- How would you implement this?

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

- What happens when you alter a table?

Physical data independence

The logical definition of the data remains unchanged, even when we make changes to the actual implementation

Adding Attributes

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

- Let's add a list of product that each company produces
  - How? Recall that tables are flat!

Adding Attributes

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

Product(pname, price, category, manufacturer)

pname	price	category	manufacturer
SingleTouch	149.99	photography	Canon
AC	300	Appliance	Hitachi

Normal forms

Organizing data into normal forms removes data redundancies.

Will revisit this later in the quarter.

SQL Basics

SQL

- SQL
  - **S**tructured **Q**uery **L**anguage
  - Most widely used language to query relational data
  - One of the many languages for querying relational data
  - A **declarative** programming language

Selections in SQL

```sql
SELECT *
FROM   Product
WHERE  price > 100.0
```

selection predicate

Joins in SQL

```
SELECT pname, price
FROM   Product, Company
WHERE  manufacturer=cname AND
       country='Japan' AND price < 150
```

```
Product(pname, price, category, manufacturer)
Company(cname, country)
```

What does this query do?

Joins in SQL

```
SELECT pname, price
FROM   Product, Company
WHERE  manufacturer=cname AND
       country='Japan' AND price < 150
```

```
Product(pname, price, category, manufacturer)
Company(cname, country)
```

Retrieve all Japanese products that cost < $150

Joins in SQL

```
Product(pname, price, category, manufacturer)
Company(cname, country)
```

pname	price	manufacturer
MultiTouch	199.99	Canon
SingleTouch	49.99	Canon
SuperGizmo	250.00	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan

```
SELECT pname, price
FROM   Product, Company
WHERE  manufacturer=cname AND
       country='Japan' AND price < 150
```

join predicate

Joins in SQL

```
Product(pname, price, category, manufacturer)
Company(cname, country)
```

pname	price	manufacturer
MultiTouch	199.99	Canon
SingleTouch	49.99	Canon
SuperGizmo	250.00	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan

Retrieve all American companies that manufacture "gadget" products

Joins in SQL

```
Product(pname, price, category, manufacturer)
Company(cname, country)
```

pname	price	manufacturer
MultiTouch	199.99	Canon
SingleTouch	49.99	Canon
SuperGizmo	250.00	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan

```
SELECT DISTINCT cname
FROM   Product, Company
WHERE  country='USA' AND category = 'gadget'
       AND manufacturer = cname
```

Joins in SQL

- This query is called an inner join
  - Each row in the result **must come from both tables in the join**

- In our example, notice that companies that didn't make any "gadgets" did not show up
  - What if we want to retain those in the results as well?

Outer Joins

```
Employee(id, name)
Sales(employeeID, productID)
```

id	name
1	'Joe'
2	'Jack
3	'Jill'

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

```
SELECT *
FROM   Employee E, Sales S
WHERE  E.id = S.employeeID
```

Outer Joins

```
Employee(id, name)
Sales(employeeID, productID)
```

id	name
1	'Joe'
2	'Jack
3	'Jill'

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

```
SELECT *
FROM    Employee E INNER JOIN Sales S
ON      E.id = S.employeeID
```

Outer Joins

```
Employee(id, name)
Sales(employeeID, productID)
```

id	name
1	'Joe'
2	'Jack
3	'Jill'

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

```
SELECT *
FROM   Employee E LEFT OUTER JOIN Sales S
ON     E.id = S.employeeID
```