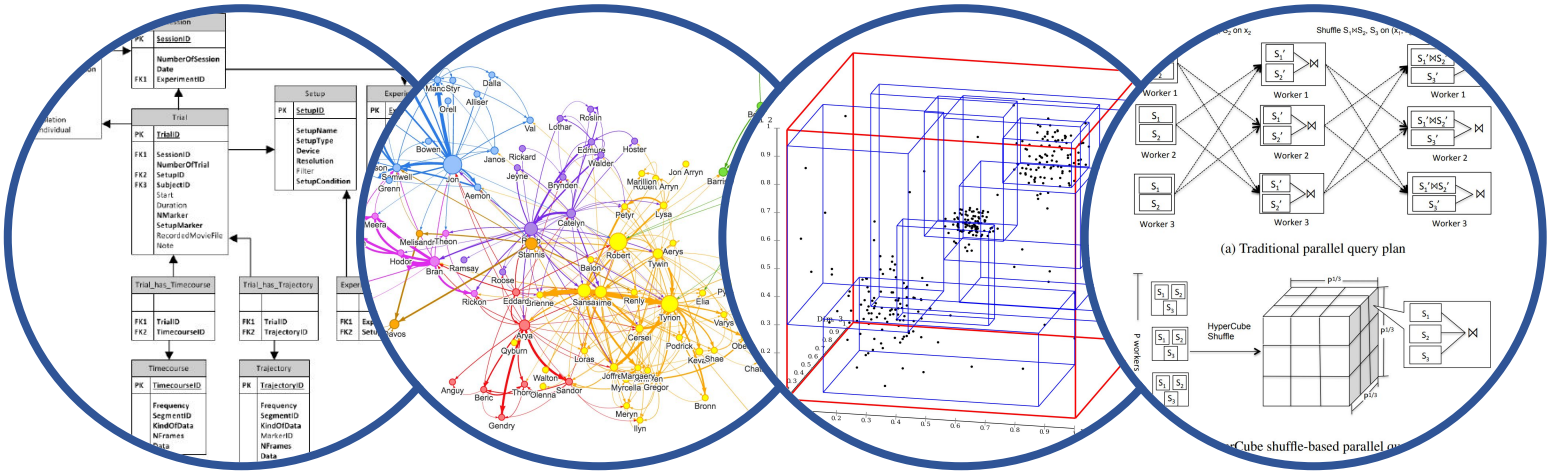


Introduction to Data Management

Databases in Theory and Practice

Alyssa Pittman

Based on slides by Jonathan Leang, Dan Suciu, et al

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

First, a story…

UW's Databases

The Seattle Times

Education | Local News

New UW payroll system behind schedule, more costly than expected

Originally published November 26, 2015 at 2:36 pm | *Updated November 27, 2015 at 6:19 am*

A project to modernize the University of Washington's payroll system is costing millions more and taking longer than expected.

Outline

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

What am I going to learn?

- Course Topics
  - Relational Databases
    - Queries
    - Database Design
    - Optimization
    - Transactions and Parallelism
  - NoSQL Databases
    - Semi-Structured Document Databases
    - Graph Databases
- Tools
  - Experimental to Enterprise Platforms
  - Cloud Services (AWS, 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

344 Staff

Instructor: Alyssa Pittman

- Email: smooo@cs.washington.edu
- Office hours: Wednesday 3:00–4:00 in CSE 216 or by appointment

TAs

- Andrew Guterman
- Jack Khuu
- Matthew Liu
- Phong Ly
- Samuel Oh
- Khang Phan
- Matt Tenczar
- Eric Yeh
- Stella Zhang

Communication - you to us

Getting in touch with us

- Office hours daily
- Piazza: https://piazza.com/washington/winter2020/cse344/home
  - Best place to ask questions
  - And to help other students!
- Mailing list: cse344-staff@cs.washington.edu
  - For personal inquiries

"Asking for help is not a sign of weakness, it's a sign of strength."

Communication - us to you

- Course website: http://cs.uw.edu/344
  - Everything will be here
- Piazza announcements
- Class mailing list
  - Very low traffic, only important announcements

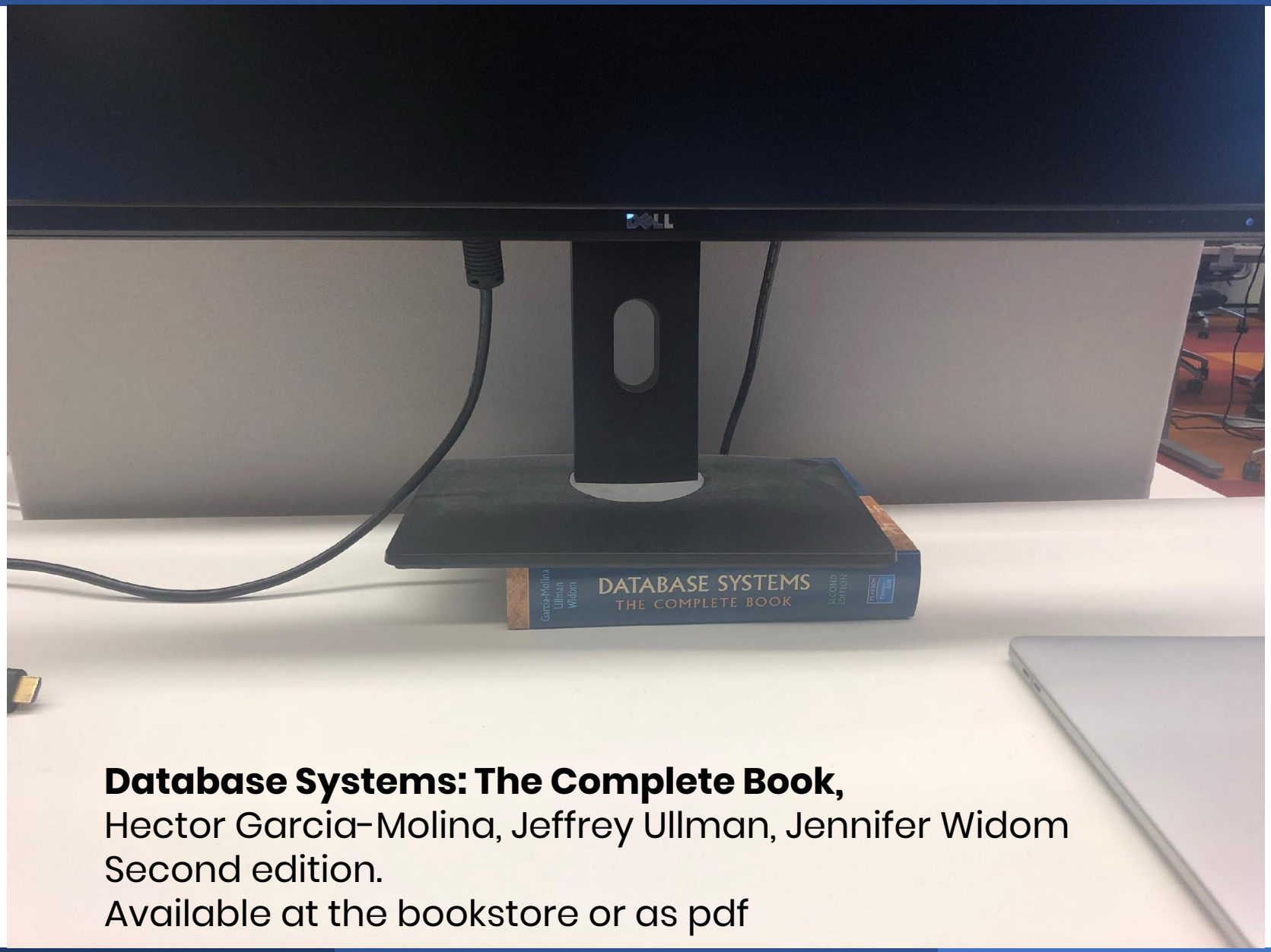
Course Format

- Lectures: this room, please attend!
- Sections: Thursdays, see web for locations.
  - Bring your laptop!
- 8 homework assignments
- Midterm and final

Exams

- Midterm (February 10th in class) and Final (March 16th 2:30 pm)
- 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,...
- Location: this classroom

References



Database Systems: The Complete Book,
Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom
Second edition.
Available at the bookstore or as pdf

Administrivia

- Grading:
  - 40% HW, 20% Midterm, 30% Final, 10% Participation
  - 4 late days, 2 days max per assignment
- Collaboration:
  - HW must be done and typed up individually, though you can talk to other students about your approach
  - We will run cheating detection

Administrivia

Let’s get started!

Database

What is a database?

- A collection of files storing related data

Examples?

Database

What is a database?

- A collection of files storing related data

Examples?

- Payroll database
- UW student database
- Amazon products database
- Airline reservation database

Database Management System: DBMS

What is a DBMS?

Database Management System: DBMS

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*

Database Management System: DBMS

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?

Database Management System: DBMS

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?

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

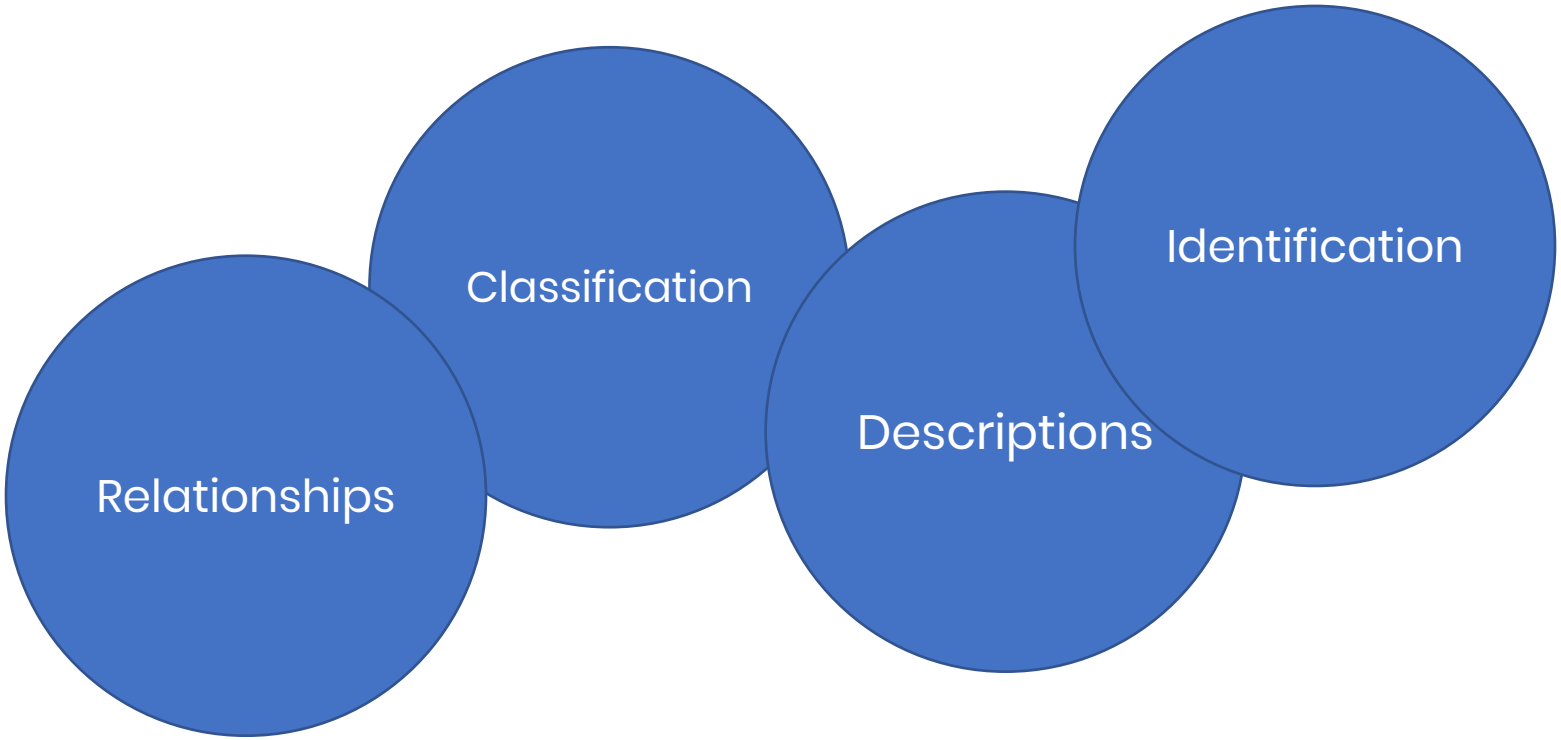
We will focus on relational DBMSs most quarter

Think About This

How do we describe information?

Think About This

How do we describe information?



Think About This

How do we describe information?

Data Model

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.

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

Data Model Zoo

There are lots of models out there!

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

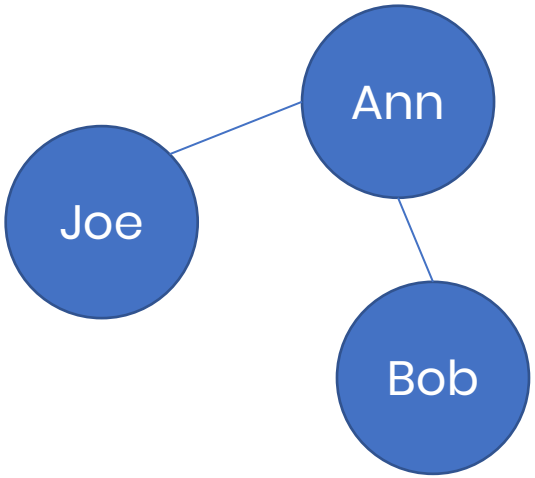
Rank Jan 2019	Rank Dec 2018	Rank Jan 2018	DBMS	Database Model
1.	1.	1.	Oracle	Relational DBMS
2.	2.	2.	MySQL	Relational DBMS
3.	3.	3.	Microsoft SQL Server	Relational DBMS
4.	4.	4.	PostgreSQL	Relational DBMS
5.	5.	5.	MongoDB	Document store
6.	6.	6.	IBM Db2	Relational DBMS
7.	7.	↑ 9.	Redis	Key-value store
8.	8.	↑ 10.	Elasticsearch	Search engine
9.	9.	↓ 7.	Microsoft Access	Relational DBMS
10.	10.	↑ 11.	SQLite	Relational DBMS

https://db-engines.com/en/ranking

Multiple Representation

Most data can be represented in different ways

An example of Facebook friends



Graph model

Person 1	Person 2	Friend
Joe	Ann	1
Ann	Bob	1
Bob	Joe	0

Relational Model

The Relational Model

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section

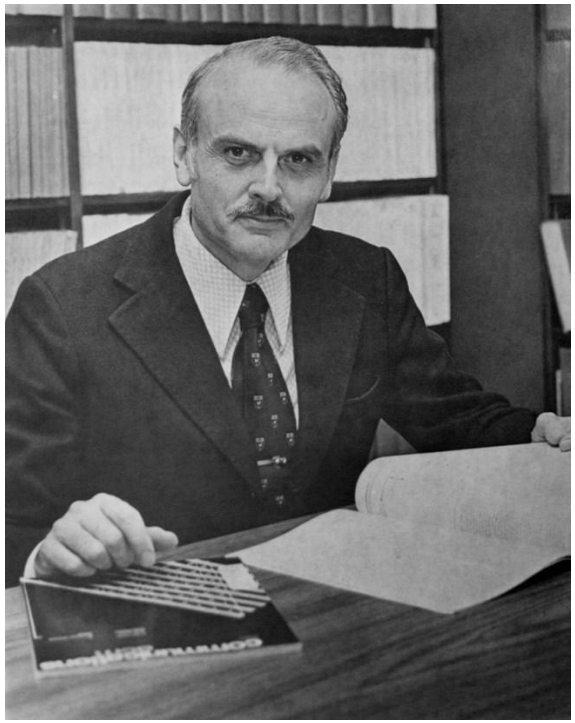
systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

The Relational Model

Most common way people describe information.



Components of the Relational Model

Payroll(UserId, Name, Job, Salary)

Components of the Relational Model

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000*
345	Allison	TA	60000*
567	Magda	Prof	90000
789	Dan	Prof	100000

* salaries may not reflect reality

Components of the Relational Model

Table/ Relation

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Components of the Relational Model

Table/ Relation

Rows/ Tuples/ Records

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Components of the Relational Model

Table/ Relation

Columns/Attributes/Fields

Rows/ Tuples/ Records

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Characteristics of the Relational Model

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

Characteristics of the Relational Model

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

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Characteristics of the Relational Model

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

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

=

UserID	Name	Job	Salary
567	Magda	Prof	90000
123	Jack	TA	50000
789	Dan	Prof	100000
345	Allison	TA	60000

Order doesn't matter

Characteristics of the Relational Model

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

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000
789	Dan	Prof	100000

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**

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000
789	Dan	Prof	100000

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**

UserID	Name	Job	Salary
123	Jack	TA	banana
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Violates attribute type assuming INT

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**

No sub-tables allowed!

UserID	Name	Job	Salary
123	Jack	JobName: TA, HasBananas: 0 JobName: banana picker, HasBananas: 1	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Characteristics of the Relational Model

But how is this data ACTUALLY stored?

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Characteristics of the Relational Model

But how is this data ACTUALLY stored?

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Don't know. Don't care.

Physical Data Independence

Structured Query Language - SQL

Alright, I have data and a schema.

How do I access it?

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.**"

Read more at the ANSI Blog: The SQL Standard – ISO/IEC 9075:2016 https://blog.ansi.org/?p=158690

Structured Query Language - SQL

- Key points about SQL:
  - A domain-specific language
    - SQL only works on relational databases
    - Not for general purpose programming (Java, C/C++, ...)
  - A declarative language
  - **Logical level of interaction with data**

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

SELECT
What kind of data I want

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

SELECT
What kind of data I want

FROM
Where the data coming from

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

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

SELECT
What kind of data I want

FROM
Where the data coming from

WHERE
Filter the data

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

?

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Name	UserID
Jack	123
Allison	345

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

SQL

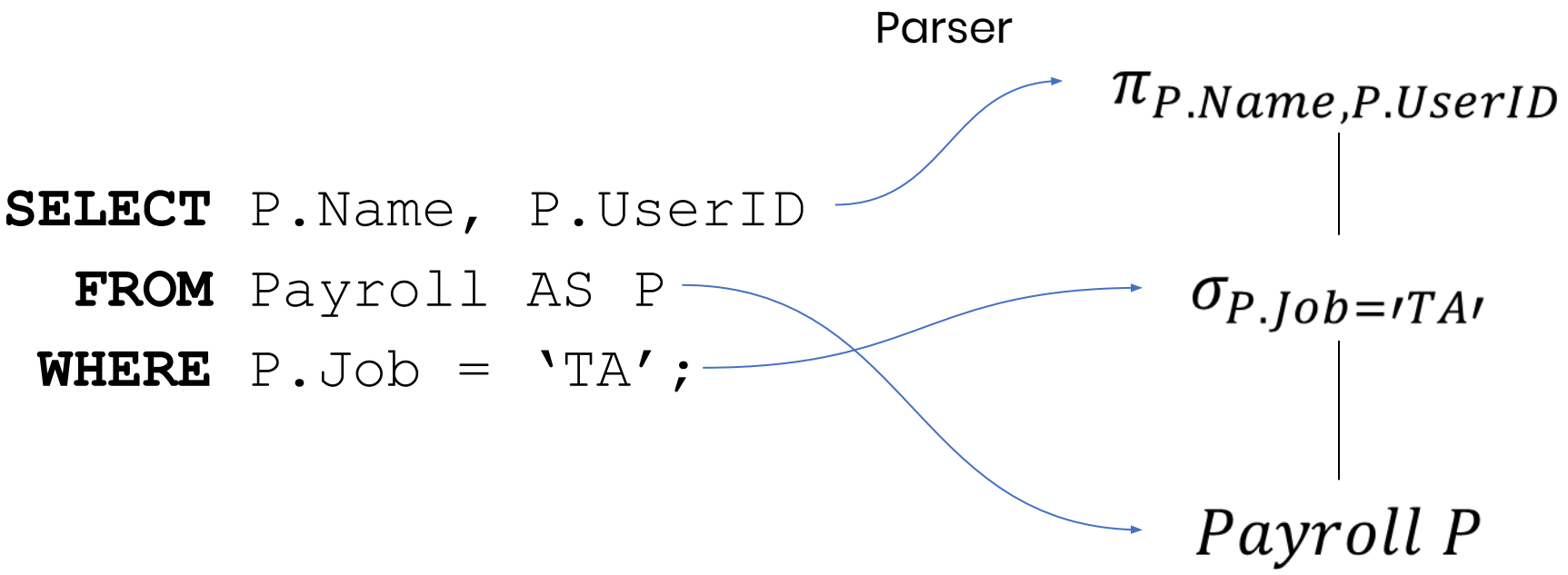
Database Internals

- Code has to boil down to instructions at some point
- Relational Database Management Systems (RDBMSs) use **Relational Algebra** (RA)

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

Database Internals

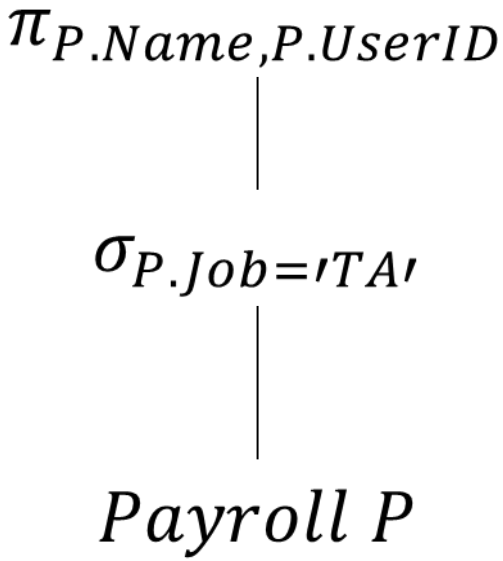
- Code has to boil down to instructions at some point
- Relational Database Management Systems (RDBMSs) use **Relational Algebra** (RA)



Database Internals

- Code has to boil down to instructions at some point
- Relational Database Management Systems (RDBMSs) use **Relational Algebra** (RA).

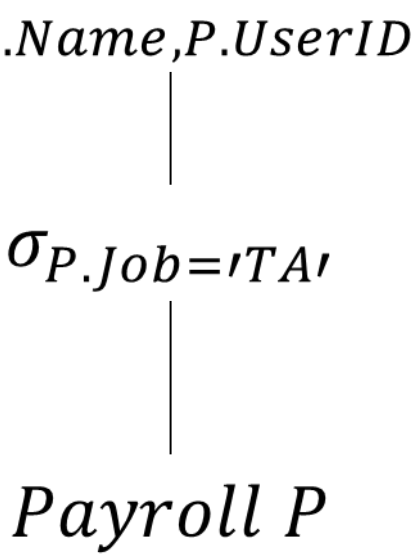
For-each semantics



Database Internals

- Code has to boil down to instructions at some point
- Relational Database Management Systems (RDBMSs) use **Relational Algebra** (RA).

For-each semantics




```
for each row in P:
  if (row.Job == 'TA'):
    output (row.Name, row.UserID)
```

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Name	UserID

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Name	UserID
Jack	123

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Name	UserID
Jack	123

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Name	UserID
Jack	123
Allison	345

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Name	UserID
Jack	123
Allison	345

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Name	UserID
Jack	123
Allison	345

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

Hello World

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Name	UserID
Jack	123
Allison	345

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

Create Table Statement

Payroll(UserId, Name, Job, Salary)

```
CREATE TABLE Payroll (
 UserId INT,
 Name VARCHAR(100),
 Job VARCHAR(100),
 Salary INT);
```

* Case insensitive, but useful for readability

Insert Statement

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
789	Dan	Prof	100000

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
789	Dan	Prof	100000
912	Alyssa	Lecturer	45000

```
INSERT INTO
   Payroll(UserId, Name, Job, Salary)
VALUES(912, 'Alyssa', 'Lecturer', 45000);
```

Insert Statement

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
789	Dan	Prof	100000

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
789	Dan	Prof	100000
912	Alyssa	Lecturer	45000

Can omit column names if inserting to all columns in order

```
INSERT INTO
   Payroll(UserId, Name, Job, Salary)
VALUES(912, 'Alyssa', 'Lecturer', 45000);
```

Insert Statement

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
789	Dan	Prof	100000

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
789	Dan	Prof	100000
912	Alyssa	Lecturer	45000

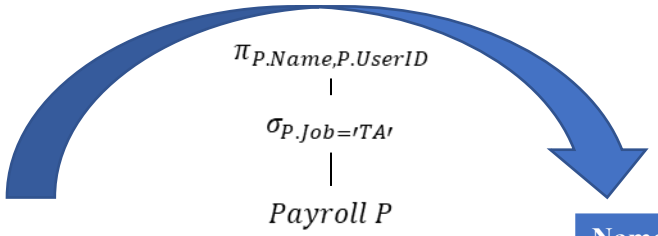
Can omit column names if inserting to all columns in order

```
INSERT INTO
   Payroll
VALUES(912, 'Alyssa', 'Lecturer', 45000);
```

Today's Takeaways!

- The **Relational Model** concept
- How a basic **SELECT-FROM-WHERE** query works
- Basic execution process (**RA**) inside a RDBMS

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000




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

Name	UserID
Jack	123
Allison	345

Homework

- HW1 released today via GitLab
  - Check the website!
- TAs will cover relevant tools in section