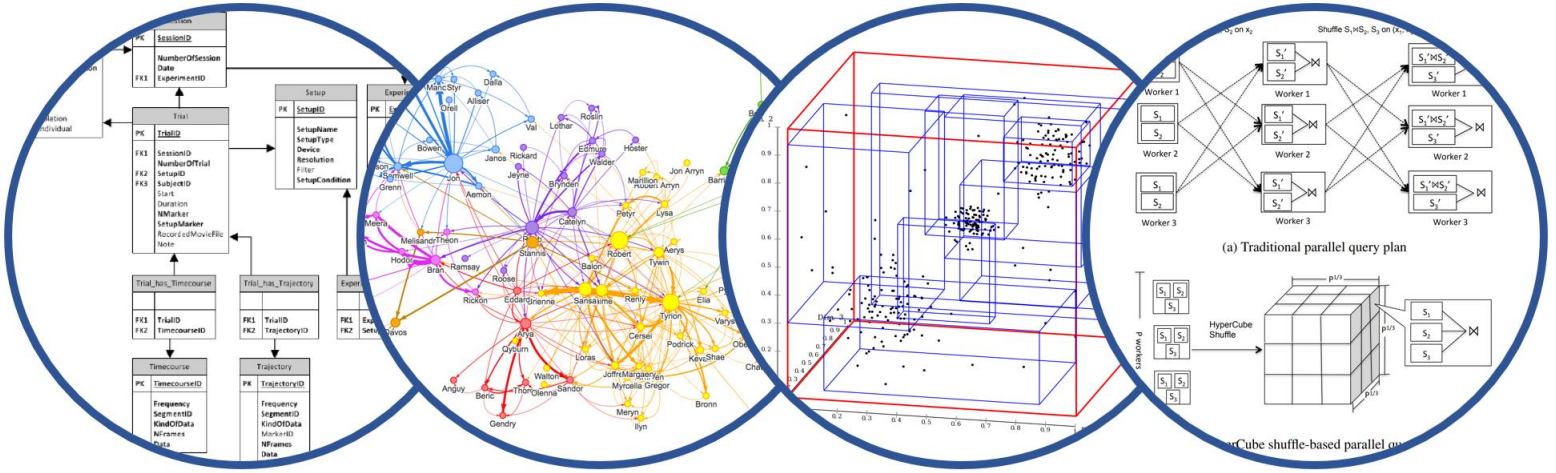


CSE 414: Intro to Data Management

Introduction

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

Outline

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

414 Staff

- Instructor: Ryan Maas (maas@cs.washington.edu)

- TAs

Shana Hutchison	Vincent Soesanto
Steve Ma	Snigdha Goel
Patrick Render	Tal Wolman
Frank Yu	Prati Dhamija
Varsha Konda	Leona Kazi

- Office hour zoom links will be in Canvas under the Zoom tab

Course Format

- Lectures: MWF on Zoom, please attend!
  - Enabling video encouraged but not required
  - Recordings posted to Canvas after class
- Sections: Separate Zoom meeting from lectures
  - Interactive worksheets and activities
- 7 homework assignments
  - First assignment published on website before section
- 4 quizzes
  - Roughly one per every 2 homework assignments
  - Assigned on Gradescope, 24 hours to complete
  - These replace typical midterm and final exam
  - Length to complete should be no longer than 45 minutes

Adjusting to Online Courses

- Teaching over zoom is new to both students and lecturers
- We’re working on improving the experience
- Feel free to ask questions in chat, or raise your hand and un-mute yourself!
- Look for survey later today about what you think has/hasn’t worked in other classes over the last 6 months, we’re always looking for suggestions

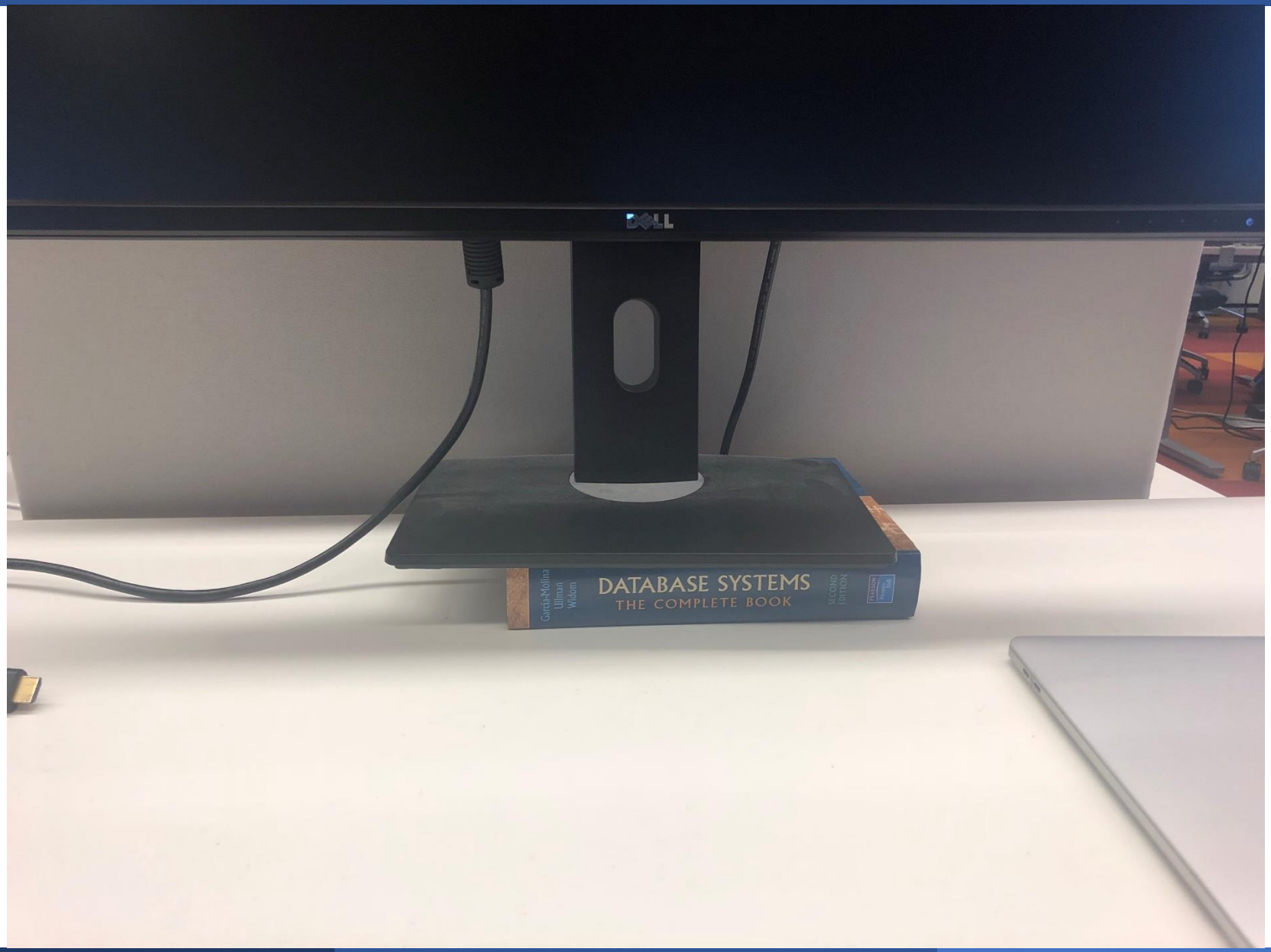
What am I going to learn?

- Course Topics
  - Queries
  - Database Design
  - Optimization
  - Transactions
  - Semi-Structured Document Databases
- Tools:
  - Experimental to Enterprise Platforms
  - Cloud Services (Microsoft Azure)
  - Parallel systems (Apache Spark)

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

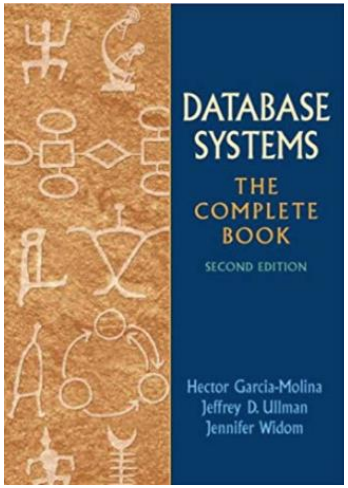
References


DELL
DATABASE SYSTEMS
THE COMPLETE BOOK
SECOND EDITION
Garcia-Molina
Ullman
Widom

Textbook

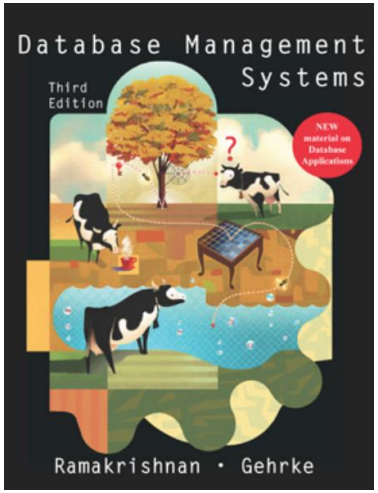
Main textbook, available at the bookstore or pdf:

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



Also useful:

- *Database Management Systems*
  (3rd Edition)



Administrivia

- Website:

https://sites.google.com/cs.washington.edu/cse-414-21wi

As a shortcut, https://cs.washington.edu/414 will also redirect you there

Communication

- Ed message board
  https://edstem.org/us/courses/3117/discussion/

- **THE** place to ask course-related questions
  - Log in today, enable notifications

- Class mailing list
  - Very low traffic, most announcements will be posted on Ed and pushed to email

Let’s get started!

Database

What is a database ?

Give examples of databases

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 ?

- *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

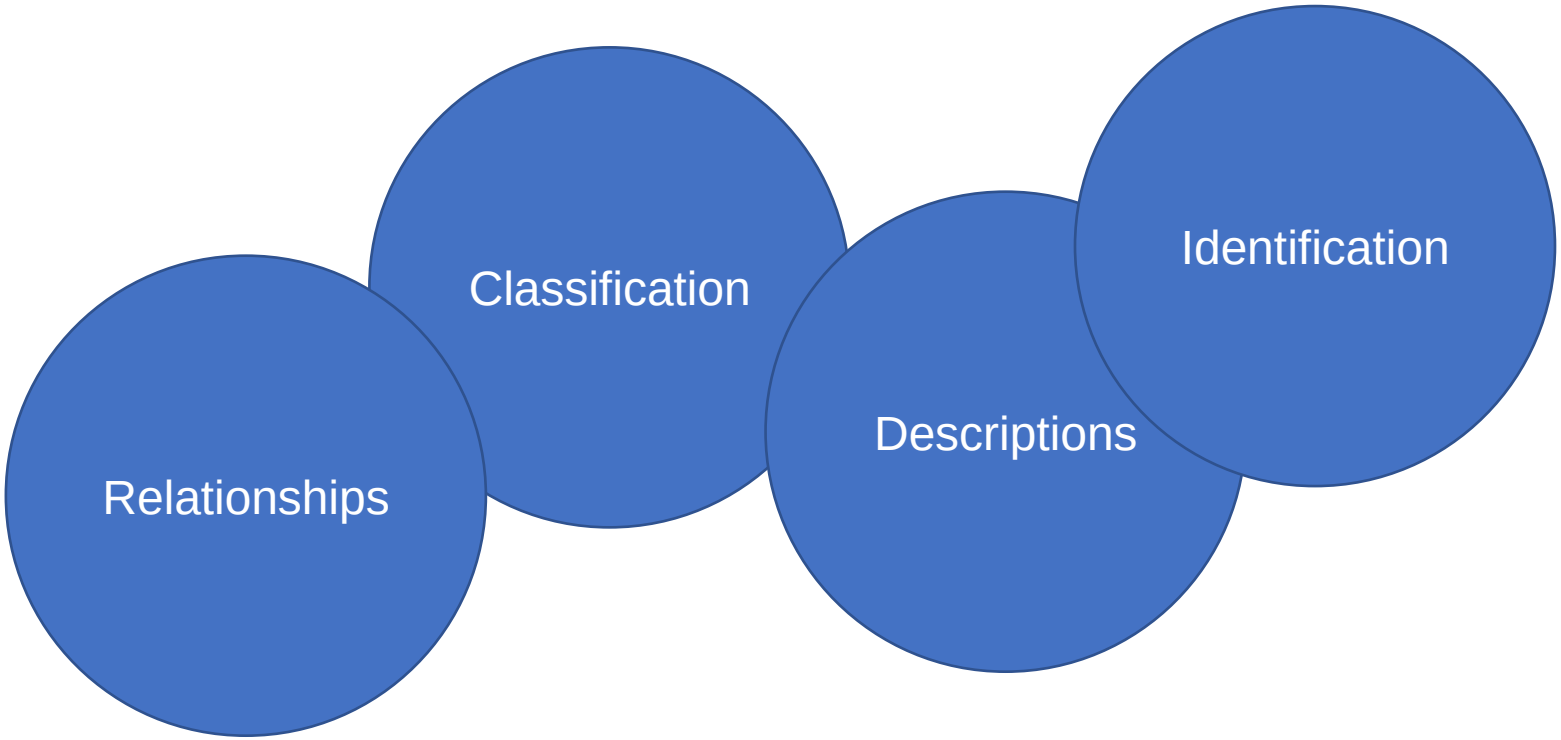
We will focus on relational DBMSs most quarter

Think About This

How do we describe information?

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
- …

What is 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.

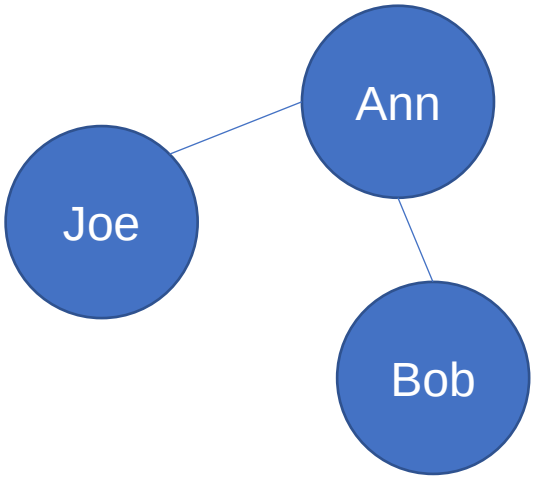
In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive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

Multiple Representation

Same 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

Data Model Zoo

There are lots of models out there!

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

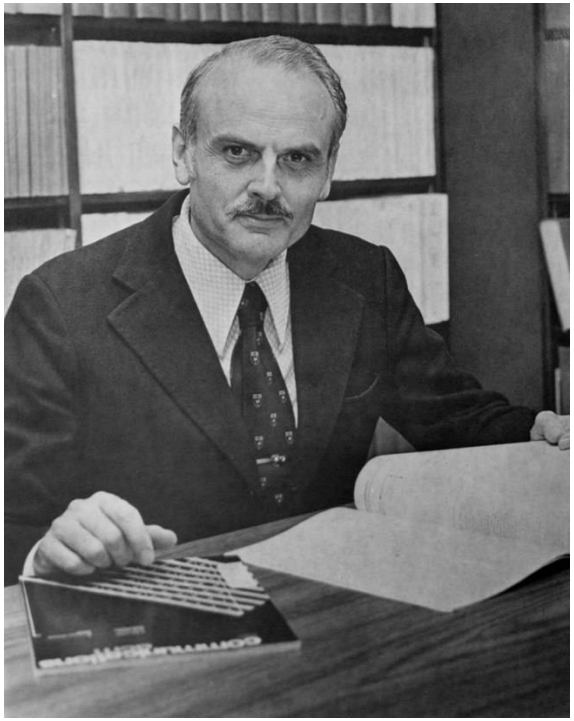
Rank Jan 2019	Rank Dec 2018	Rank Jan 2018	DBMS	Database Model
1.	1.	1.	Oracle	Relational DBMS
2.	2.	2.	MySQL	Relational DBMS
3.	3.	3.	Microsoft SQL Server	Relational DBMS
4.	4.	4.	PostgreSQL	Relational DBMS
5.	5.	5.	MongoDB	Document store
6.	6.	6.	IBM Db2	Relational DBMS
7.	7.	↑ 9.	Redis	Key-value store
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

The Relational Model

Again, how we describe information?

Most common answer: **The Relational Model**



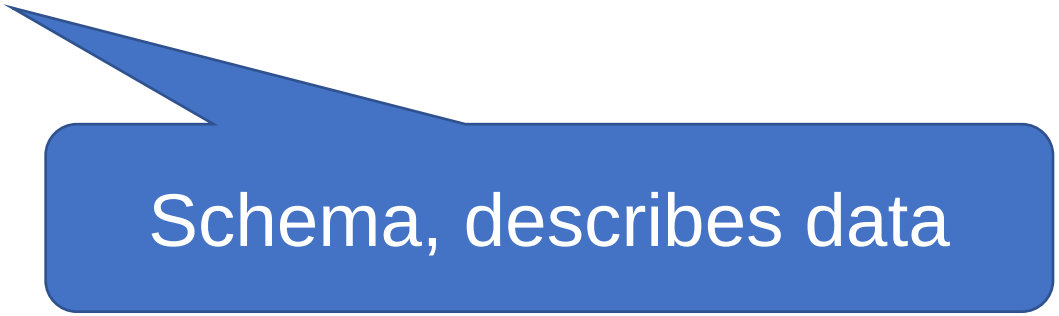
Ted Codd

Components of the Relational Model

Payroll (UserId, Name, Job, Salary)

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

* I wish

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

Instance of actual data

* I wish

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

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

Rows/ Tuples/ Records

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**

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**

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

- Originally defined with **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

- Originally defined with **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 / HasBananas: TA / 0; banana farmer / 1	0000
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++, …)
  - **Logical level of interaction with data**

Recap - Basic SQL query

Payroll

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

```
SELECT *
  FROM Payroll
```

Recap - Basic SQL query

Payroll

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

```
SELECT  *
  FROM  Payroll
```

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

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';
```