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
  - Queries
  - Database Design
  - Optimization
  - Transactions and Parallelism
  - Semi-Structured Document Databases
- Tools:
  - Experimental to Enterprise Platforms
  - Cloud Services (AWS, Azure)
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 3:30 in CSE 216 or by appointment

TAs

- Jonathan Leang
- Amy Xu
- Andrew Guterman
- Guna Prasaad
- Jack Khuu
- Khang Phan
Communication - you to us

Getting in touch with us

- Office hours daily
- Ed Discuss

“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
- Ed Discuss: https://us.edstem.org/courses/130/
  - Best place to ask questions
- 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 (October 30th in class) and Final (December 9th 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
Grading:
• 50% HW, 20% Midterm, 30% Final
• 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
REALITY CAN BE WHATEVER I WANT.
Let’s get started!
What is a database?

- A collection of files storing related data

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

Examples?
- Payroll database
- UW student database
- Amazon products database
- Airline reservation database
What is a 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
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?
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?
How do we describe information?

- Classification
- Identification
- Descriptions
- Relationships
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

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 3. The network model, on the other hand, has spawned a variety of other solutions 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

<table>
<thead>
<tr>
<th>Person 1</th>
<th>Person 2</th>
<th>Friend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Ann</td>
<td>1</td>
</tr>
<tr>
<td>Ann</td>
<td>Bob</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>Joe</td>
<td>0</td>
</tr>
</tbody>
</table>

Graph model

Relational Model
Data Model Zoo

There are lots of models out there!

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

![DBMS Ranking Table](https://db-engines.com/en/ranking)

1. Oracle ➢ Relational DBMS
2. MySQL ➢ Relational DBMS
3. Microsoft SQL Server ➢ Relational DBMS
4. PostgreSQL ➢ Relational DBMS
5. MongoDB ➢ Document store
6. IBM Db2 ➢ Relational DBMS
7. Redis ➢ Key-value store
8. Elasticsearch ➢ Search engine
9. Microsoft Access ➢ Relational DBMS
10. SQLite ➢ Relational DBMS
Again, how we describe information?

Most common answer: The Relational Model
Components of the Relational Model

Payroll(UserId, Name, Job, Salary)
### 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>

* I wish
### Table/Relation

<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>
### Components of the Relational Model

#### Table/Relation

<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>
Components of the Relational Model

Table/Relation

Columns/Attributes/Fields

<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>

Rows/Tuples/Records
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**

<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>
Characteristics of the Relational Model

- **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>

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

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**

<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>
<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>
<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>banana</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 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**

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>JobName</th>
<th>HasBananas</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>HasBananas</td>
<td>0</td>
<td>50000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>banana picker</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td></td>
<td></td>
<td>60000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td></td>
<td></td>
<td>90000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td></td>
<td></td>
<td>100000</td>
</tr>
</tbody>
</table>

No sub-tables allowed!
But how is this data ACTUALLY stored?

<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>
But how is this data ACTUALLY stored?

<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>

Don’t know. Don’t care.

**Physical Data Independence**
Alright, I have data and a schema. How do I access it?
“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.”

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

**SQL Query**

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

**SELECT**

What kind of data I want

**FROM**

Where the data coming from

**WHERE**

Filter the data
### 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>

**SQL Query**

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

```sql
SELECT P.Name, P.UserID
FROM Payroll AS P
WHERE P.Job = 'TA';
```
How does a computer understand abstract SQL text?
- Code has to boil down to instructions at some point
- Relational Database Management Systems (RDBMSs) use **Relational Algebra** (RA)

```sql
SELECT P.Name, P.UserID
FROM Payroll AS P
WHERE P.Job = 'TA';
```
- 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';
```
• Code has to boil down to instructions at some point

• Relational Database Management Systems (RDBMSs) use **Relational Algebra** (RA).

```
π_{P.Name, P.UserID} \\
σ_{P.Job='TA'} \\
Payroll P
```

For-each semantics
- Code has to boil down to instructions at some point

- Relational Database Management Systems (RDBMSs) use **Relational Algebra (RA)**.

\[
\begin{align*}
\pi_{P.Name,P.UserID} & \quad \text{For-each semantics} \\
\sigma_{P.Job=\text{\textquoteleft}TA\textquoteright} & \\
\text{Payroll } P & \\
\end{align*}
\]

for each row in P:

```python
    if (row.Job == 'TA'):
        output (row.Name, row.UserID)
```
Database Internals

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

\[ \pi_{P}\text{.Name,}P\text{.UserID} \]
\[ \sigma_{P\text{.Job}='TA'} \]
\[ Payroll\ P \]

Tuples “flow” up the RA tree getting filtered and modified
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>

```
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>

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>

**SQL Query**

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

March 31, 2019

Introduction
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>

**SQL Query**

```sql
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>

```
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>

**SQL Query**

```
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>

```
SELECT P.Name, P.UserID
FROM Payroll AS P
WHERE P.Job = 'TA';
```
Today’s Takeaways!

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

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

<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>

<table>
<thead>
<tr>
<th>Name</th>
<th>UserID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack</td>
<td>123</td>
</tr>
<tr>
<td>Allison</td>
<td>345</td>
</tr>
</tbody>
</table>
Payroll(UserId, Name, Job, Salary)

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

* Case insensitive, but useful for readability
### 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>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>912</td>
<td>Alyssa</td>
<td>Lecturer</td>
<td>45000</td>
</tr>
</tbody>
</table>

**Insert Statement**

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

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>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
</tbody>
</table>

Can omit column names if inserting to all columns in order

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

Can omit column names if inserting to all columns in order
Homework

- HW1 released today via GitLab
  - Check the website!

- TAs will cover relevant tools in section