Intro to Database Systems (15-445/645)

FALL 2022

Andy Pavlo

Course Intro & Relational Model
TODAY’S AGENDA

Course Logistics
Relational Model
Relational Algebra
I do **not** control the wait list.

I do **not** take bribes.

Admins will move students moved off the wait list as new spots become available.

If you are not currently enrolled, the likelihood that you will get in is unfortunately very low.

*15-445/645 will be offered in Spring 2023!*
LECTURE RULES

Please interrupt me for the following reasons:
→ I am speaking too fast.
→ You don't understand what I am talking about.
→ You have a database-related question.

Do **not** interrupt me for the following reasons:
→ Whether you can use the bathroom.
→ Questions about blockchains.

I will **not** answer questions about the lecture immediately after class.
COURSE OVERVIEW

This course is about the design/implementation of database management systems (DBMSs).

This is **not** a course about how to use a DBMS to build applications or how to administer a DBMS.
→ See [CMU 95-703](#) (Heinz College)
PROJECTS

All projects will use the CMU DB Group BusTub academic DBMS.
→ Each project builds on the previous one.
→ We will not teach you how to write/debug C++17.

Total of four late days the entire semester for projects only.

You must complete Project #0 before Sept 11th.
COURSE LOGISTICS

Course Policies + Schedule: Course Web Page
Discussion + Announcements: Piazza
Homeworks + Projects: Gradescope
Final Grades: Canvas

Non-CMU students will be able to complete all assignments using Gradescope (PXWVR5).
→ D
→ Somebody needs to finish my Wikipedia article.
COURSE LOGISTICS

Pavlo was born and raised in the streets of Baltimore, MD. After completing bachelor’s and master’s degrees from Rochester Institute of Technology and Brown University, he completed his Ph.D. from Brown University under Stan Zdonik and Mike Stonebraker.

Non-CMU students will be able to complete all assignments using Gradescope (PXWVR5).

→ D

→ Somebody needs to finish my Wikipedia article.
PLAGIARISM WARNING

The homework and projects must be your own original work. They are not group assignments. You may not copy source code from other people or the web.

Plagiarism is not tolerated. You will get lit up. → Please ask me if you are unsure.

See CMU's Policy on Academic Integrity for additional information.
¡Databases! – A Database Seminar Series
Mondays @ 4:30pm (starting on 9/12)
→ Live on Zoom. Published to Youtube afterwards
→ https://db.cs.cmu.edu/seminar2022
Carnegie Mellon University

¡Databases!

A Database Seminar Series
FALL 2022

https://db.cs.cmu.edu/seminar2022
Databases
DATABASE

Organized collection of inter-related data that models some aspect of the real-world.

Databases are the core component of most computer applications.
DATABASE EXAMPLE

Create a database that models a digital music store to keep track of artists and albums.

Things we need for our store:
→ Information about Artists
→ What Albums those Artists released
FLAT FILE STRAWMAN

Store our database as comma-separated value (CSV) files that we manage ourselves in our application code.
→ Use a separate file per entity.
→ The application must parse the files each time they want to read/update records.
Create a database that models a digital music store.

**Artist**(name, year, country)

- "Wu-Tang Clan", 1992, "USA"
- "Notorious BIG", 1992, "USA"
- "GZA", 1990, "USA"

**Album**(name, artist, year)

- "Enter the Wu-Tang", "Wu-Tang Clan", 1993
- "St. Ides Mix Tape", "Wu-Tang Clan", 1994
- "Liquid Swords", "GZA", 1990
Example: Get the year that GZA went solo.

```
for line in file.readlines():
    record = parse(line)
    if record[0] == "GZA":
        print(int(record[1]))
```

**Artist(name, year, country)**

"Wu-Tang Clan", 1992, "USA"
"Notorious BIG", 1992, "USA"
"GZA", 1990, "USA"
FLAT FILES: DATA INTEGRITY

How do we ensure that the artist is the same for each album entry?

What if somebody overwrites the album year with an invalid string?

What if there are multiple artists on an album?

What happens if we delete an artist that has albums?
FLAT FILES: IMPLEMENTATION

How do you find a particular record?

What if we now want to create a new application that uses the same database?

What if two threads try to write to the same file at the same time?
What if the machine crashes while our program is updating a record?

What if we want to replicate the database on multiple machines for high availability?
A database management system (DBMS) is software that allows applications to store and analyze information in a database.

A general-purpose DBMS supports the definition, creation, querying, update, and administration of databases in accordance with some data model.
DATA MODELS

A data model is a collection of concepts for describing the data in a database.

A schema is a description of a particular collection of data, using a given data model.
DATA MODELS

- Relational
- Key/Value
- Graph
- Document / Object
- Wide-Column / Column-family
- Array / Matrix / Vectors
- Hierarchical
- Network
- Multi-Value

This Course
EARLY DBMSs

Early database applications were difficult to build and maintain on available DBMSs in the 1960s.
→ Examples: IDS, IMS, CODASYL
→ Computers were expensive, humans were cheap.

Tight coupling between logical and physical layers.

Programmers had to (roughly) know what queries the application would execute before they could deploy the database.
EARLY DBMSs

Ted Codd was a mathematician working at IBM Research in the late 1960s.

He saw IBM's developers spending their time rewriting database programs every time the database’s schema or layout changed.

Devised the relational model in 1969.

Edgar F. Codd
Ted Codd was a mathematician working at IBM Research in the late 1960s. He saw IBM's developers spending their time rewriting database programs every time the database's schema or layout changed. Codd devised the relational model in 1969. Users of large data banks must be protected from having to rework the data that is organized in complex, interrelated ways. In particular, the problem of ensuring that the data remains accessible even when the data is changed. In this paper, Codd presents a relational model of data that enables users to work with databases in a consistent and predictable way. The model is based on the concept of a relational database, which stores data in tables and relationships between those tables. The relational model provides a powerful and flexible way to store and retrieve data, and it has become the foundation for many modern database systems.
The relational model defines a database abstraction based on relations to avoid maintenance overhead.

Key tenets:
→ Store database in simple data structures (relations).
→ Physical storage left up to the DBMS implementation.
→ Access data through high-level language, DBMS figures out best execution strategy.
RELATIONAL MODEL

Structure: The definition of the database's relations and their contents.

Integrity: Ensure the database's contents satisfy constraints.

Manipulation: Programming interface for accessing and modifying a database's contents.
A relation is an unordered set that contain the relationship of attributes that represent entities.

A tuple is a set of attribute values (also known as its domain) in the relation.
→ Values are (normally) atomic/scalar.
→ The special value NULL is a member of every domain (if allowed).

Artist(name, year, country)

<table>
<thead>
<tr>
<th>name</th>
<th>year</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wu-Tang Clan</td>
<td>1992</td>
<td>USA</td>
</tr>
<tr>
<td>Notorious BIG</td>
<td>1992</td>
<td>USA</td>
</tr>
<tr>
<td>GZA</td>
<td>1990</td>
<td>USA</td>
</tr>
</tbody>
</table>

\[ n \text{-ary Relation} = \text{Table with } n \text{ columns} \]
A relation's primary key uniquely identifies a single tuple. Some DBMSs automatically create an internal primary key if a table does not define one.

Auto-generation of unique integer primary keys:
→ **SEQUENCE** (SQL:2003)
→ **AUTO_INCREMENT** (MySQL)
A relation's primary key uniquely identifies a single tuple.
Some DBMSs automatically create an internal primary key if a table does not define one.

Auto-generation of unique integer primary keys:
→ **SEQUENCE** (SQL:2003)
→ **AUTO_INCREMENT** (MySQL)
RELATIONAL MODEL: FOREIGN KEYS

A foreign key specifies that an attribute from one relation has to map to a tuple in another relation.
## RELATIONAL MODEL: FOREIGN KEYS

### Artist Table

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>year</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Wu-Tang Clan</td>
<td>1992</td>
<td>USA</td>
</tr>
<tr>
<td>456</td>
<td>Notorious BIG</td>
<td>1992</td>
<td>USA</td>
</tr>
<tr>
<td>789</td>
<td>GZA</td>
<td>1990</td>
<td>USA</td>
</tr>
</tbody>
</table>

### Artist-Album Table

<table>
<thead>
<tr>
<th>artist_id</th>
<th>album_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>11</td>
</tr>
<tr>
<td>123</td>
<td>22</td>
</tr>
<tr>
<td>789</td>
<td>22</td>
</tr>
<tr>
<td>456</td>
<td>22</td>
</tr>
</tbody>
</table>

### Album Table

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Enter the Wu-Tang</td>
<td>1993</td>
</tr>
<tr>
<td>22</td>
<td>St.Ides Mix Tape</td>
<td>1994</td>
</tr>
<tr>
<td>33</td>
<td>Liquid Swords</td>
<td>1995</td>
</tr>
</tbody>
</table>
DATA MANIPULATION LANGUAGES (DML)

Methods to store and retrieve information from a database.

**Procedural:**
→ The query specifies the (high-level) strategy to find the desired result based on sets / bags.

**Non-Procedural (Declarative):**
→ The query specifies only what data is wanted and not how to find it.

← Relational Algebra
← Relational Calculus
REATIONAL ALGEBRA

Fundamental operations to retrieve and manipulate tuples in a relation.
→ Based on set algebra.

Each operator takes one or more relations as its inputs and outputs a new relation.
→ We can "chain" operators together to create more complex operations.

σ Select
π Projection
∪ Union
∩ Intersection
− Difference
× Product
⋈ Join
Choose a subset of the tuples from a relation that satisfies a selection predicate.

→ Predicate acts as a filter to retain only tuples that fulfill its qualifying requirement.

→ Can combine multiple predicates using conjunctions / disjunctions.

**Syntax:** \( \sigma_{\text{predicate}}(R) \)

```
SELECT * FROM R
WHERE a_id='a2' AND b_id>102;
```
RELATIONAL ALGEBRA: PROJECTION

Generate a relation with tuples that contains only the specified attributes.
→ Can rearrange attributes’ ordering.
→ Can manipulate the values.

Syntax: $\Pi_{A_1,A_2,...,A_n}(R)$
Generate a relation that contains all tuples that appear in either only one or both input relations.

**Syntax:** \((R \cup S)\)

**Example:**

<table>
<thead>
<tr>
<th>a_id</th>
<th>b_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>101</td>
</tr>
<tr>
<td>a2</td>
<td>102</td>
</tr>
<tr>
<td>a3</td>
<td>103</td>
</tr>
</tbody>
</table>

\(R(a_id,b_id)\)

<table>
<thead>
<tr>
<th>a_id</th>
<th>b_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>a3</td>
<td>103</td>
</tr>
<tr>
<td>a4</td>
<td>104</td>
</tr>
<tr>
<td>a5</td>
<td>105</td>
</tr>
</tbody>
</table>

\(S(a_id,b_id)\)

\[(SELECT * FROM R) \quad \text{UNION ALL} \quad (SELECT * FROM S);\]
Generate a relation that contains only the tuples that appear in both of the input relations.

**Syntax:** \((R \cap S)\)
Generate a relation that contains only the tuples that appear in the first and not the second of the input relations.

Syntax: \((R - S)\)
RELATIONAL ALGEBRA: PRODUCT

Generate a relation that contains all possible combinations of tuples from the input relations.

Syntax: \((R \times S)\)

**SELECT * FROM R CROSS JOIN S;**

**SELECT * FROM R, S;**
Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.

**Syntax:** \((R \bowtie S)\)

```sql
SELECT *
FROM R NATURAL JOIN S;
```

```sql
SELECT *
FROM R JOIN S USING (a_id, b_id);
```
RELATIONAL ALGEBRA: EXTRA OPERATORS

Rename (ρ)
Assignment (R←S)
Duplicate Elimination (δ)
Aggregation (γ)
Sorting (τ)
Division (R÷S)
Relational algebra still defines the high-level steps of how to compute a query.

→ $\sigma_{b\_id=102}(R \bowtie S)$ vs. $(R \bowtie (\sigma_{b\_id=102}(S)))$

A better approach is to state the high-level answer that you want the DBMS to compute.

→ Retrieve the joined tuples from $R$ and $S$ where $b\_id$ equals 102.
The relational model is independent of any query language implementation.

**SQL** is the *de facto* standard (many dialects).

```python
for line in file.readlines():
    record = parse(line)
    if record[0] == "GZA":
        print(int(record[1]))
```

```sql
SELECT year FROM artists
WHERE name = 'GZA';
```
DATA MODELS

- Relational
- Key/Value
- Graph
- **Document / Object** ← Leading Alternative
- Wide-Column / Column-family
- Array / Matrix / Vectors
- Hierarchical
- Network
- Multi-Value
Embed data hierarchy into a single object.

**Artist**

**ArtistAlbum**

**Album**

$R_1(id,\ldots)$

$R_2(artist\_id,album\_id)$

$R_3(id,\ldots)$
Embed data hierarchy into a single object.
Document Data Model

Embed data hierarchy into a single object.

Application Code

```java
class Artist {
    int id;
    String name;
    int year;
    Album albums[];
}

class Album {
    int id;
    String name;
    int year;
}
```
CONCLUSION

Databases are ubiquitous.

Relational algebra defines the primitives for processing queries on a relational database.

We will see relational algebra again when we talk about query optimization + execution.
Modern SQL
→ Make sure you understand basic SQL before the lecture.