Database: what is it?

- Persistent data
- Query and update language for accessing and modifying data
- Query optimization
- Transactions and concurrency control

Provides meaning-based view of data: shields from irrelevant detail
Basic Architecture of a Database System

Data Independence
logical and view levels are independent of physical level
The Relational Model

(a) The customer table

<table>
<thead>
<tr>
<th>customer_id</th>
<th>customer_name</th>
<th>customer_street</th>
<th>customer_city</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>12 Alma St.</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>3 Main St.</td>
<td>Harrison</td>
</tr>
<tr>
<td>182-73-6091</td>
<td>Turner</td>
<td>123 Putnam Ave.</td>
<td>Stamford</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>100 Main St.</td>
<td>Harrison</td>
</tr>
<tr>
<td>336-66-9999</td>
<td>Lindsay</td>
<td>175 Park Ave.</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>72 North St.</td>
<td>Rye</td>
</tr>
</tbody>
</table>

(b) The account table

<table>
<thead>
<tr>
<th>account_number</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>A-215</td>
<td>700</td>
</tr>
<tr>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>A-305</td>
<td>350</td>
</tr>
<tr>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>A-217</td>
<td>750</td>
</tr>
<tr>
<td>A-222</td>
<td>700</td>
</tr>
</tbody>
</table>

(c) The depositor table

<table>
<thead>
<tr>
<th>customer_id</th>
<th>account_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>A-101</td>
</tr>
<tr>
<td>192-83-7465</td>
<td>A-201</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>A-215</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>A-102</td>
</tr>
<tr>
<td>182-73-6091</td>
<td>A-305</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>A-217</td>
</tr>
<tr>
<td>336-66-9999</td>
<td>A-222</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>A-201</td>
</tr>
</tbody>
</table>
Relational db: theory → practice

Frege: FO

Tarski: algebra for FO

Codd: relational databases
Databases: implemented logic!

• FO lies at the core of modern database systems
  “Databases = FO on every desk!”

• Relational query languages are based on FO:
  SQL

• More powerful query languages (all the way to XML) are based on extensions of FO
Why is FO so successful as a query language?

• easy to use syntactic variants
  
  SQL, QBE

• efficient implementation via relational algebra
  amenable to analysis and simplification

• potential for perfect scaling to large databases
  very fast response can be achieved
  using parallel processing
Journey of a Query

SQL ~ FO

Relational Algebra

Query Rewriting

Query Execution Plan

Execution

Physical Level

select … from … where

π₁₃(P▹◁Q) ▹◁ …

π₁₄(P▹◁S) ▹◁ Q ▹◁ R

Physical Level
Journey of a Query

SQL \sim FO

Relational Algebra

Query Rewriting

Query Execution Plan

Execution

Physical Level

select \ldots from \ldots where

\pi_{13}(P \bowtie Q) \bowtie \ldots

\pi_{14}(P \bowtie S) \bowtie Q \bowtie R

\pi_{14}

Q R

P S
The Relational Model

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>Name</th>
<th>SSN</th>
<th>HomePhone</th>
<th>Address</th>
<th>OfficePhone</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Benjamin Bayer</td>
<td>305-61-2435</td>
<td>373-1616</td>
<td>2918 Bluebonnet Lane</td>
<td>null</td>
<td>19</td>
<td>3.21</td>
</tr>
<tr>
<td></td>
<td>Katherine Ashly</td>
<td>381-62-1245</td>
<td>375-4409</td>
<td>125 Kirby Road</td>
<td>null</td>
<td>18</td>
<td>2.89</td>
</tr>
<tr>
<td></td>
<td>Dick Davidson</td>
<td>422-11-2320</td>
<td>null</td>
<td>3452 Elgin Road</td>
<td>749-1253</td>
<td>25</td>
<td>3.53</td>
</tr>
<tr>
<td></td>
<td>Charles Cooper</td>
<td>489-22-1100</td>
<td>376-9821</td>
<td>265 Lark Lane</td>
<td>749-6492</td>
<td>28</td>
<td>3.93</td>
</tr>
<tr>
<td></td>
<td>Barbara Benson</td>
<td>533-69-1238</td>
<td>839-8461</td>
<td>7384 Fontana Lane</td>
<td>null</td>
<td>19</td>
<td>3.25</td>
</tr>
</tbody>
</table>
The Relational Model

• columns in each table are named by attributes
• each attribute has an associated domain (set of allowed values)
• data in each table consists of a set of rows (tuples) providing values for the attributes
Relation Schema

“type declaration”

• Relation name
• Set of attributes
• Domain of each attribute
• Integrity constraints

Example

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

integer  char strings  13-digits
Attribute Types

• Each attribute of a relation has a name
• The set of allowed values for each attribute is called the domain of the attribute
• Attribute values are (normally) required to be atomic; that is, indivisible
• Sometimes, the special value null is considered a member of every domain
Relation Instance

An instance of a relation schema is the current content of the relation: a finite set of rows (tuples) over the attributes, with values from the attribute domains.

<table>
<thead>
<tr>
<th>account_number</th>
<th>branch_name</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-101</td>
<td>Downtown</td>
<td>500</td>
</tr>
<tr>
<td>A-215</td>
<td>Mianus</td>
<td>700</td>
</tr>
<tr>
<td>A-102</td>
<td>Perryridge</td>
<td>400</td>
</tr>
<tr>
<td>A-305</td>
<td>Round Hill</td>
<td>350</td>
</tr>
<tr>
<td>A-201</td>
<td>Brighton</td>
<td>900</td>
</tr>
<tr>
<td>A-222</td>
<td>Redwood</td>
<td>700</td>
</tr>
<tr>
<td>A-217</td>
<td>Brighton</td>
<td>750</td>
</tr>
</tbody>
</table>
Relations are Unordered Sets

The tuples are not considered to be ordered, even though they appear to be so in the displayed tabular form.

<table>
<thead>
<tr>
<th>account_number</th>
<th>branch_name</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-101</td>
<td>Downtown</td>
<td>500</td>
</tr>
<tr>
<td>A-215</td>
<td>Mianus</td>
<td>700</td>
</tr>
<tr>
<td>A-102</td>
<td>Perryridge</td>
<td>400</td>
</tr>
<tr>
<td>A-305</td>
<td>Round Hill</td>
<td>350</td>
</tr>
<tr>
<td>A-201</td>
<td>Brighton</td>
<td>900</td>
</tr>
<tr>
<td>A-222</td>
<td>Redwood</td>
<td>700</td>
</tr>
<tr>
<td>A-217</td>
<td>Brighton</td>
<td>750</td>
</tr>
</tbody>
</table>
Relational Integrity Constraints (aka dependencies)

- Constraints are *properties* that must hold on all valid relation instances of a database.
- Some very common types of constraints:
  - Key constraints
  - Referential integrity constraints
Key Constraints

• **Key**: A minimal set of attributes $K$ of $R$ such that no two tuples in any valid relation instance $r(R)$ will have the same value for $K$. That is, for any distinct tuples $t_1$ and $t_2$ in $r(R)$, $t_1(K) \neq t_2(K)$.

**Example**: The CAR relation schema:

\[
\text{CAR(State, Reg#, SerialNo, Make, Model, Year)}
\]

has two natural keys $\text{Key1} = \{\text{State, Reg#}\}$, $\text{Key2} = \{\text{SerialNo}\}$.

• If a relation has several *candidate keys*, one is chosen arbitrarily to be the **primary key**.
# Key Constraints

<table>
<thead>
<tr>
<th>CAR</th>
<th>LicenseNumber</th>
<th>EngineSerialNumber</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas ABC-739</td>
<td>A69352</td>
<td></td>
<td>Ford</td>
<td>Mustang</td>
<td>96</td>
</tr>
<tr>
<td>Florida TVP-347</td>
<td>B43696</td>
<td></td>
<td>Oldsmobile</td>
<td>Cutlass</td>
<td>99</td>
</tr>
<tr>
<td>New York MPO-22</td>
<td>X83554</td>
<td></td>
<td>Oldsmobile</td>
<td>Delta</td>
<td>95</td>
</tr>
<tr>
<td>California 432-TFY</td>
<td>C43742</td>
<td></td>
<td>Mercedes</td>
<td>190-D</td>
<td>93</td>
</tr>
<tr>
<td>California RSK-629</td>
<td>Y82935</td>
<td></td>
<td>Toyota</td>
<td>Camry</td>
<td>98</td>
</tr>
<tr>
<td>Texas RSK-629</td>
<td>U028365</td>
<td></td>
<td>Jaguar</td>
<td>XJS</td>
<td>98</td>
</tr>
</tbody>
</table>

The primary key attributes are *underlined*. 
Referential Integrity

EMPLOYEE

DEPARTMENT

DEPT_LOCATIONS

PROJECT

WORKS_ON

DEPENDENT
Other Types of Constraints

Semantic Integrity Constraints: based on application semantics and cannot be expressed by the model per se
- e.g., “the max. no. of hours per employee for all projects they work on is 40 hrs per week”

- A constraint specification language is used to express these (e.g. assertions SQL)
Core database issues

• Query languages
• Query processing
• Database design
• Transaction management
• Concurrency control
Query language

Language for extracting information from the database

SQL is the most widely used query language based on logic, primarily declarative
Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation
Query Processing (Cont.)

- Alternative ways of evaluating a given query
  - Equivalent formulations
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of evaluation plans
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions
Database Design

The process of designing the structure of the database:

• **Logical Design** – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  – Application decision – What attributes should we record in the database? What dependencies hold among them?
  – Computer science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?

• **Physical Design**
  – Deciding on the physical layout of the database
Transaction Management

• A transaction is a collection of operations that performs a single logical function in a database application.

• Transaction-management ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

• Concurrency-control manages the interaction among the concurrent transactions, to ensure the consistency of the database.
DB Theory vs. core database issues

✓ Query languages
✓ Query processing
✓ Database design
• Transaction management
✓ Concurrency control
Beyond standard relational databases

- XML, JSON, graph databases ("No SQL")
- Deductive databases
- Temporal databases
- Multimedia databases
- Geographic information systems
- Real-time and active databases
- Database-driven Web applications/services
- Big data, data analytics, parallel processing (map-reduce)
- **Data science: databases + ML**
CSE 233 Outline

- FO (aka CALC), relational algebra
- Static analysis for query processing
- Dependency theory
- Extending FO with recursion: Datalog and fixpoint logics
- Expressiveness and complexity
  - Ehrenfeucht-Fraisse games, 0/1 laws
  - The quest for a language for PTIME
- Highly expressive languages
Other topics (if time)

- Incomplete information
- Complex objects
- Selected research topics
Databases at UCSD

- Prof. Alin Deutsch
- Prof. Arun Kumar
- Prof. Yannis Papakonstantinou
- Prof. Victor Vianu

Database group Web site: [https://dbucsd.github.io](https://dbucsd.github.io)
papers, seminars, bragging….

- Intersections with other CSE groups, HDSI