CMSC424: Database Design
Introduction/Overview

Professor: Pete Keleher
keleher@cs.umd.edu

Outline

- Relational Model (Chapter 2)
  - Basics
  - Keys
  - Relational operations
  - Relational algebra basics

- SQL (Chapter 3)
  - Setting up the PostgreSQL database
  - Data Definition (3.2)
  - Basics (3.3-3.5)
  - Null values (3.6)
  - Aggregates (3.7)
Context

- **Data Models**
  - Conceptual representation of the data

- **Data Retrieval**
  - How to ask questions of the database
  - How to answer those questions

- **Data Storage**
  - How/where to store data, how to access it

- **Data Integrity**
  - Manage crashes, concurrency
  - Manage semantic inconsistencies

Relational Data Model

**Introduced by Ted Codd (late 60's – early 70's)**

- Before = “Network Data Model” (Cobol as DDL, DML)
- Very contentious: Database Wars (Charlie Bachman vs. Ted Codd)

**Relational data model contributes:**

1. *Separation of logical, physical data models (data independence)*
2. *Declarative query languages*
3. *Formal semantics*
4. *Query optimization (key to commercial success)*

**1st prototypes:**

- Ingres → CA
- Postgres → Illustra → Informix → IBM
- System R → Oracle, DB2
Key Abstraction: Table

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>500</td>
</tr>
</tbody>
</table>

Terms:

- Tables (aka: Relations)

Why called Relations?
Closely correspond to mathematical concept of a relation

Relations

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>500</td>
</tr>
</tbody>
</table>

Considered equivalent to math sets...

\{ (Downtown, A-101, 500),
  (Brighton, A-201, 900),
  (Brighton, A-217, 500) \}

Relational database semantics defined in terms of mathematical relations
Relations

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>500</td>
</tr>
</tbody>
</table>

Considered equivalent to...

\{(Downtown, A-101, 500),
  (Brighton, A-201, 900),
  (Brighton, A-217, 500)\}

Terms:
- Tables (*relations*)
- Rows (*n-tuples, or tuples*)
- Columns (*attributes*)
- Schema (e.g.: Acct_Schema = (bname, acct_no, balance))

Definitions

**Relation Schema (or Schema)**

A *list of attributes and their domains* (often elide domains)
E.g. account(account-number, branch-name, balance)

*Programming language equivalent: A variable (e.g. x)*

**Relation Instance**

A particular instantiation of a relation with actual values
Will change with time

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>500</td>
</tr>
</tbody>
</table>

*Programming language equivalent: Value of a variable*
Definitions

Domains of an attribute/column
The set of permitted values
- e.g., **bname** is **String**, balance is a positive real number
We assume domains are atomic, values are indivisible
- no lists or arrays

Null value
A special value used if the value of an attribute for a row is:
- **unknown** (e.g., don’t know address of a customer)
- **inapplicable** (e.g., “spouse name” attribute for a customer)
- **withheld/hidden**
Different interpretations all captured by a single concept – leads to major headaches and problems

Relational Schema for a University

- classroom(building, room_number, capacity)
- department(dept_name, building, budget)
- course(course_id, title, dept_name, credits)
- instructor(ID, name, dept_name, salary)
- section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
- teaches(ID, course_id, sec_id, semester, year)
- student(ID, name, dept_name, tot_cred)
- takes(Id, course_id, sec_id, semester, year, grade)
- advisor(s_ID, i_ID)
- time_slot(time_slot_id, day, start_time, end_time)
- prereq(course_id, prereq_id)
Outline

- Overview of modeling
- Relational Model (Chapter 2)
  - Basics
  - Keys
  - Relational operations
  - Relational algebra basics
- SQL (Chapter 3)
  - Setting up the PostgreSQL database
  - Data Definition (3.2)
  - Basics (3.3-3.5)
  - Null values (3.6)
  - Aggregates (3.7)

Keys

- Let $K \subseteq R$ (set of all attributes)
- $K$ is a superkey of $R$ if values for $K$ are sufficient to identify a unique tuple of any possible relation $r(R)$
  - Example: $\{ID\}$ and $\{ID, name\}$ are both superkeys of instructor.
- Superkey $K$ is a candidate key if $K$ is minimal (i.e., no subset of it is a superkey)
  - Example: $\{ID\}$ is a candidate key for Instructor
- One of the candidate keys is selected to be the primary key
  - Typically one that is small and immutable (doesn’t change often)
- Primary key typically highlighted (e.g., underlined)
Tables in a University Database

classroom(building, room_number, capacity)
department(dept_name, building, budget)
course(course_id, title, dept_name, credits)
instructor(ID, name, dept_name, salary)

Primary Keys

takes(ID, course_id, sec_id, semester, year, grade)

What about ID, course_id?
No. May repeat:
(“1011049”, “CMSC424”, “102”, “Fall”, 2015, null)

What about ID, course_id, sec_id?
May repeat:
(“1011049”, “CMSC424”, “101”, “Fall”, 2015, null)

What about ID, course_id, sec_id, semester?
## Tables in a University Database

- **classroom** (building, room_number, capacity)
- **department** (dept_name, building, budget)
- **course** (course_id, title, dept_name, credits)
- **instructor** (ID, name, dept_name, salary)
- **section** (course_id, sec_id, semester, year, building, room_number, time_slot_id)
- **teaches** (ID, course_id, sec_id, semester, year)
- **student** (ID, name, dept_name, tot_cred)
- **takes** (ID, course_id, sec_id, semester, year, grade)
- **advisor** (s_ID, i_ID)
- **time_slot** (time_slot_id, day, start_time, end_time)
- **prereq** (course_id, prereq_id)

## Keys

- **Foreign key**: Primary key of a relation that appears in another relation
  - [ID] from **student** appears in **takes**, **advisor**
  - **student** called **referenced** relation
  - **takes** is the **referencing** relation
  - Typically shown by an arrow from referencing to referenced

- **Foreign key constraint**: the tuple corresponding to that primary key must exist
  - Imagine:
    - Tuple: (‘student101’, ‘CMSC424’) in **takes**
    - But no tuple corresponding to ‘student101’ in **student**
  - Also called **referential integrity constraint**
Schema Diagram for University Database

Schema Diagram: Banking Enterprise
More Examples

- Married(person1_ssn, person2_ssn, date_married, date_divorced)

- Account(cust_ssn, acct_number, cust_name, balance, cust_address)

- RA(student_id, project_id, supervisor_id, appt_start, appt_end)

- Person(Name, DOB, Born, Education, Religion, …)
  - Information typically found on Wikipedia Pages
  - Unclear what could be a primary key here: you could in theory have two people who match on all of those

- If a single account per customer, then: cust_ssn
- Else: (cust_ssn, account_number)

- Could be smaller if there are some restrictions – requires some domain knowledge of the data being stored
Outline

- Overview of modeling
- Relational Model (Chapter 2)
  - Basics
  - Keys
  - Relational operations
  - Relational algebra basics
- SQL (Chapter 3)
  - Setting up the PostgreSQL database
  - Data Definition (3.2)
  - Basics (3.3-3.5)
  - Null values (3.6)
  - Aggregates (3.7)

Relational Query Languages

- Example schema: $R(A, B)$
- Practical languages
  - SQL
    - select A from R where B = 5;
  - Datalog (sort of practical)
    - q(A) :- R(A, 5)
- Formal languages
  - Relational algebra
    - $\pi_A(\sigma_{B=5}(R))$
  - Tuple relational calculus
    - $\{ t : \{A\} \ | \ \exists s : \{A, B\} ( R(A, B) \land s.B = 5 ) \}$
  - Domain relational calculus
    - Similar to tuple relational calculus (declarative)
Relational Operations

- Some languages “procedural” and have operations
  - Each operation takes one or two relations as input, and produces a single relation as output
  - Examples: C and Java

- “Declarative” (also called “non-procedural”) languages specify output, but not operations
  - SQL, Relational Algebra
  - Relational calculus
  - Datalog (used as an intermediate layer in quite a few systems today)

Select Operation

Choose a subset of the tuples that satisfies some predicate
Denoted by $\sigma$ in relational algebra

Relation $r$

$\sigma_{A=B \land D > 5}(r)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\beta$</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>23</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>23</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Project

Choose a subset of the columns (for all rows)
Denoted by $\prod$ in relational algebra

Relation $r$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\beta$</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>23</td>
<td>10</td>
</tr>
</tbody>
</table>

$\Pi_{A,D}(r)$

<table>
<thead>
<tr>
<th>A</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>7</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>7</td>
</tr>
<tr>
<td>$\beta$</td>
<td>3</td>
</tr>
<tr>
<td>$\beta$</td>
<td>10</td>
</tr>
</tbody>
</table>

Relational algebra following “set” semantics – so no duplicates
SQL allows for duplicates – we will cover the formal semantics later

Set Union, Difference

Relation $r, s$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
</tr>
</tbody>
</table>

$\checkmark$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>3</td>
</tr>
</tbody>
</table>

$r \cup s$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>3</td>
</tr>
</tbody>
</table>

$r \setminus s$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
</tr>
</tbody>
</table>

Must be compatible schemas

What about intersection?
Can be derived

$r \cap s = r \setminus (r \setminus s)$;
Cartesian Product

Combine tuples from two relations

If one contains N tuples, the other M, result contains N*M tuples

Rarely useful – usually want pairs of tuples that satisfy some condition

Join

Combine tuples satisfying predicate from two relations

- Equivalent to Cartesian Product followed by a Select