Quiz 1 Review

What we’re covering:

- Relational Algebra / Calculus
- SQL
- E-R Diagrams
- Keys and Integrity Constraints
Relational Algebra

- Be familiar with basic Relational Algebra operators, such as:
  - select (σ)
  - project (p)
  - union (U)
  - set difference (–)
  - cartesian product (X)
  - rename (ρ)

**Closure Property**
**Union (\( \cup \))**

**Notation:** \( \text{Relation}_1 \cup \text{Relation}_2 \)

\( R \cup S \) valid only if:

1. \( R, S \) have same number of columns (arity)
2. \( R, S \) corresponding columns have same name and domain (compatibility)

**Example:**

\[ (\pi_{\text{cname}}(\text{depositor})) \cup (\pi_{\text{cname}}(\text{borrower})) = \]

**Schema:**

<table>
<thead>
<tr>
<th>Deppositor</th>
<th>Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{cname} )</td>
<td>( \text{cname} )</td>
</tr>
<tr>
<td>( \text{acct_no} )</td>
<td>( \text{Ino} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
</tr>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Hayes</td>
</tr>
<tr>
<td>Turner</td>
</tr>
<tr>
<td>Jones</td>
</tr>
<tr>
<td>Lindsay</td>
</tr>
<tr>
<td>Jackson</td>
</tr>
<tr>
<td>Curry</td>
</tr>
<tr>
<td>Williams</td>
</tr>
<tr>
<td>Adams</td>
</tr>
</tbody>
</table>
Natural Join

Notation: \( Relation_1 \bowtie Relation_2 \)

Idea: combines \( \rho, \times, \sigma \)

\[
\begin{array}{|c|c|c|c|}
\hline
A & B & C & D \\
\hline
1 & \alpha & + & 10 \\
2 & \alpha & - & 10 \\
2 & \alpha & - & 20 \\
3 & \beta & + & 10 \\
\hline
\end{array}
\quad \bowtie \quad
\begin{array}{|c|c|c|}
\hline
E & B & D \\
\hline
\alpha & + & 10 \\
\alpha & - & 20 \\
\beta & + & 10 \\
\beta & + & 10 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|}
\hline
A & B & C & D & E \\
\hline
1 & \alpha & + & 10 & \text{`a`} \\
2 & \alpha & - & 10 & \text{`a`} \\
2 & \alpha & - & 20 & \text{`a`} \\
3 & B & + & 10 & \text{`b`} \\
3 & \beta & + & 10 & \text{`c`} \\
\hline
\end{array}
\]

depositor \bowtie borrower

\[
\equiv \pi_{\text{cname, acct_no, lno}} (\sigma_{\text{cname} = \text{cname2}} (\text{depositor} \times \rho_{t(\text{cname2, lno})} (\text{borrower})))
\]
Division

Notation: $\text{Relation}_1 \div \text{Relation}_2$

Idea: expresses “for all” queries

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>2</td>
</tr>
<tr>
<td>α</td>
<td>3</td>
</tr>
<tr>
<td>β</td>
<td>1</td>
</tr>
<tr>
<td>γ</td>
<td>1</td>
</tr>
<tr>
<td>γ</td>
<td>3</td>
</tr>
<tr>
<td>γ</td>
<td>4</td>
</tr>
<tr>
<td>γ</td>
<td>6</td>
</tr>
<tr>
<td>δ</td>
<td>1</td>
</tr>
<tr>
<td>δ</td>
<td>2</td>
</tr>
</tbody>
</table>

Query: Find values for A in r which have corresponding B values for all B values in s
Aggregate Functions and Operations

- An **aggregate function** takes a collection of values and returns a single value as a result.
  
  - **avg:** average value
  - **min:** minimum value
  - **max:** maximum value
  - **sum:** sum of values
  - **count:** number of values

- **Aggregate operation** in relational algebra

  \[ G_1, G_2, \ldots, G_n \quad g \quad F_1(A_1), F_2(A_2), \ldots, F_n(A_n) (E) \]

  - E is any relational-algebra expression
  - \( G_1, G_2, \ldots, G_n \) is a list of attributes on which to group
    (can be empty)
  - Each \( F_i \) is an aggregate function
  - Each \( A_i \) is an attribute name
Tuple Relational Calculus

- **Query** has the form: \{ T \mid p(T) \}
  - \( p(T) \) denotes a formula in which tuple variable \( T \) appears.

- **Answer** is the set of all tuples \( T \) for which the formula \( p(T) \) evaluates to \textit{true}.
Question: Find all cities of residence of all employees who work directly for “Jones” in relational algebra and relational calculus

Relational Algebra: \( \pi_{\text{City}}(\sigma_{\text{manager_name} = \text{‘Jones’}}(\text{manages} \bowtie \text{employees})) \)

\[
\{ t \mid \exists m \in \text{manages} \exists e \in \text{employee}(e[\text{person_name}] = m[\text{person_name}] \\
\land m[\text{manager_name}] = \text{‘Jones’} \\
\land t[\text{city}] = e[\text{city}]) \}
\]
SQL

Be familiar with basic Query construction such as:

```sql
select name
from instructor
where dept_name = 'Comp. Sci.' and salary > 80000
```
SQL: Cartesian Product, Joins

Find the Cartesian product \textit{instructor} \times \textit{teaches}

\begin{verbatim}
select *
from instructor, teaches
\end{verbatim}

Joins

\begin{verbatim}
select name, course_id
from instructor, teaches
where instructor.ID = teaches.ID;
\end{verbatim}
Nested Queries

- Find courses offered in Fall 2009 and in Spring 2010

```sql
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
  course_id in (select course_id
                 from section
                 where semester = 'Spring' and year= 2010);
```
SQL: Exists, In, Except

- The **exists** construct returns the value **true** if the argument subquery is nonempty.
- **exists** $r \iff r \neq \emptyset$
- **not exists** $r \iff r = \emptyset$

```
(select course_id from section where sem = 'Fall' and year = 2009)
except
(select course_id from section where sem = 'Spring' and year = 2010)
```

You can think of except as the set difference operation from Relational Algebra!
Question: Assume that the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.

```
employee (person_name, street, city)
works (person_name, company_name, salary)
company (company_name, city)
manages (person_name, manager_name)
```

```
select S.company_name
from company S
where not exists ((select city
    from company
    where company_name = 'Small Bank Corporation')
except
    (select city
    from company T
    where S.company_name = T.company_name))
```

All cities
where SBC
is located in

All cities of
a company
What is a Data Model?

- Framework for organizing and interpreting data

Example: E/R Data Model
E/R Data Model

Basics

- **Entities**
  - noun phrases (e.g., Bob Smith, Thayer St. Branch)
  - contained in entity sets (e.g. Employee, Branch)
  - have attributes (e.g., Employee = (essn, ename, ...))

- **Relationships**
  - verb phrases (e.g., works_at, works_for)
  - relate 2 (binary) or more (n-ary) entities
  - relationship sets characterize relationships amongst entity sets
    - e.g., \((\text{Bob Smith}, \text{Thayer St Branch}) \in \text{Works}_\text{At}\)
E/R Data Model

**Keys**

- Key = set of attributes identifying individual entities or relationships

![Diagram of Employee with attributes: essn, name, eaddress, ephone]

- **A. Superkey:**
  - any attribute set that distinguishes identities
  - e.g., \{essn\}, \{essn, name, eaddress\}

- **B. Candidate Key:**
  - “minimal superkey” (can’t remove attributes and preserve “keyness”)
  - e.g., \{essn\}, \{name, eaddress\}

- **C. Primary Key:**
  - candidate key chosen as the key by a DBA
  - e.g., \{essn\} (denoted by underline)
E/R Data Model
Existence Dependencies and Weak Entity Sets

- Idea:
  - Existence of one entity depends on another

- Example: Loans and Loan Payments

Diagram:
- Loan
  - lno
  - lamt

- Loan_Pmt

- Payment
  - pno
  - pdate
  - pamt

- Weak Entity Set
- Identifying Relationship
- Total Participation
# E/R Diagrams and Relations

<table>
<thead>
<tr>
<th>Relationship Cardinality</th>
<th>Relational Schema</th>
</tr>
</thead>
</table>
| n:m                      | \( E_1 = (a_1, ..., a_n) \)  
                          | \( E_2 = (b_1, ..., b_m) \)  
                          | \( R = (a_1, b_1, c_1, ..., c_n) \) |
| n:1                      | \( E_1 = (a_1, ..., a_n, b_1, c_1, ..., c_n) \)  
                          | \( E_2 = (b_1, ..., b_m) \) |
| 1:n                      | \( E_1 = (a_1, ..., a_n) \)  
                          | \( E_2 = (b_1, ..., b_m, a_1, c_1, ..., c_n) \) |
| 1:1                      | Treat as n:1 or 1:n |
Question: Consider a database used to record the marks that students get in different exams of different course offerings (sections). Construct an ER diagram that models exams as entities.
Referential Integrity Constraints

Idea:

Prevent “dangling tuples” (e.g.: A loan with bname, Waltham when no Waltham tuple in branch)

Illustrated:

Referential Integrity:

Ensure that: Foreign Key $\rightarrow$ Primary Key value

Note: Need not ensure (i.e.: Not all branches have to have loans)

CSCI 127: Introduction to Database Systems
Domains (cont.):

CREATE DOMAIN bank-balance integer(
    CONSTRAINT not-overdrawn,
    CHECK (value ≥ 0),
    CONSTRAINT not-null-value
    CHECK (value NOT NULL)
)

CREATE TABLE depositor(
    ...
    balance bank-balance
    ...
)
Functional Dependencies

An Example:

$$\text{loan-info} =$$

<table>
<thead>
<tr>
<th>bname</th>
<th>lno</th>
<th>cname</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dntn</td>
<td>L-17</td>
<td>Jones</td>
<td>1000</td>
</tr>
<tr>
<td>Dntn</td>
<td>L-17</td>
<td>Williams</td>
<td>1000</td>
</tr>
<tr>
<td>Redwood</td>
<td>L-23</td>
<td>Smith</td>
<td>1000</td>
</tr>
<tr>
<td>Perry</td>
<td>L-15</td>
<td>Hayes</td>
<td>1500</td>
</tr>
<tr>
<td>Redwood</td>
<td>L-23</td>
<td>Johnson</td>
<td>1000</td>
</tr>
</tbody>
</table>

True or False?
- amt $\rightarrow$ lno?
- lno $\rightarrow$ cname?
- lno $\rightarrow$ lno?
- bname $\rightarrow$ lno?

Can’t always decide by looking at populated db’s

Observe:

Tuples with the same value for lno will always have the same value for amt

We write: lno $\rightarrow$ amt
(lno “determines” amt, or amt is “functionally determined” by lno)
Question: Define a super key in terms of FD’s

Answer: Any set of attributes that functionally determine all attributes in a relation
GOOD LUCK!

YOU GOT THIS