Today’s Agenda

• Go over Tuesday’s Quiz
• Finish the last SQL extension
  – Divide operator
• Database Design
  – Start ER conceptual data modeling
Example...

- Suppose we have this extra table in our Banking database

<table>
<thead>
<tr>
<th>Account</th>
<th>AccountTypes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>checking</td>
</tr>
<tr>
<td></td>
<td>savings</td>
</tr>
</tbody>
</table>

How do we find customers that have at least one account of each account type?

Relational Algebra: Divide Operator

- Suppose we have this extra table in our Banking database

<table>
<thead>
<tr>
<th>Account</th>
<th>AccountTypes</th>
<th>Query Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type</td>
<td>Owner</td>
</tr>
<tr>
<td></td>
<td>checking</td>
<td>J. Smith</td>
</tr>
<tr>
<td></td>
<td>savings</td>
<td></td>
</tr>
</tbody>
</table>

\[ \pi_{\text{Owner}, \text{Type}}(\text{Account}) \div \text{AccountTypes} \]

Find account owners who have ALL types of accounts
Relational Algebra: Divide Operator

For $R \div S$ where $R(r_1, r_2, r_3, r_4)$ and $S(s_1, s_2)$

Since $S$ has two attributes $s_1$ and $s_2$

... there must be two attributes in R (i.e., $r_3$ and $r_4$) that
are defined on the same domains, respectively

- i.e., $(r_3, r_4)$ is union-compatible with $(s_1, s_2)$

The query answer has the remaining attributes $(r_1, r_2)$
such that

... the answer has a tuple $(r_1, r_2)$ if it “appears” with
every $S$-tuple in $R$

---

Relational Algebra: Divide Operator

Division is another operator that can be defined using
the five basic operators (… what is another one?)

For $R(x, y) \div S(y)$ where $x = x_1, x_2, \ldots$ and $y = y_1, y_2, \ldots$

... we compute all $x$-values that are not “disqualified” by
some $y$-values in $S$

- an $x$-value is disqualified if for some $y$-value in $S$, $(x, y)$ is
  not in $R$

Disqualified $x$-values = $\pi_x((\pi_x(R) \times S) - R)$

$R \div S = \pi_x(R) - \pi_x((\pi_x(R) \times S) - R)$
Relational Algebra Divide Operator in SQL

Not explicitly supported, but can be simulated in various ways, e.g. …

```sql
SELECT A1.Owner FROM Account A1 WHERE NOT EXISTS ( SELECT T.Type FROM AccountType T WHERE NOT EXISTS ( SELECT A2.* FROM Account A2 WHERE A2.Type = T.Type AND A2.Owner = A1.Owner ) )
```

Find owners A1 such that

... there is no account type T

... that A1 does not posses

Conceptual Data Modeling

- Similar to software design …
  - requirements gathering and analysis
  - *application architecture and design*
  - implementation and testing
  - maintenance

Database design involves multiple steps (esp. prior to creating table definitions)

- Here we focus on conceptual design using the **Entity-Relationship Model**
  - similar to the use of UML diagrams for software design
ER Terminology

- An “Entity” is an object that can be distinguished from other objects
  - e.g., the individual “John Smith” or a particular company
  - described using a set of attribute-value pairs
  - one or more attributes as ids (i.e., keys)
- An “Entity Set” is a collection of similar entities
  - e.g., the set of employee entities
  - defined by the type of attributes and relationships entities of the set are characterized by
  - often informally called an “entity” if everyone knows we are talking about the schema
- An Entity Set is also be called an “Entity Type”
Entity-Relationship (ER) Diagrams [Chen 71]

Key:
- **Entity Set**
- **Relation Set**
- **Attribute**

Component:
- **Employee**
  - `ssn`
  - `name`
  - `lot`
- **Department**
  - `code`
  - `name`
- **Project**
  - `number`
  - `name`
  - `enddate`
  - `startdate`
  - `budget`

Relationship:
- **Manager**
- **Sponsor**
- **Project Member**
ER Terminology

- A "Relationship" is an association among 2 or more entities
  - e.g., John's home department is Pharmacy 2
  - just as entities are instances of entity sets, relationships are instances of relationship sets …

- A "Relationship Set" is a collection of similar relationships
  - e.g., the set of home department relationships
  - defined by the participating entity types and other constraints
  - often informally called a “relationship” if everyone knows we are talking about the schema

- A Relationship Set is also be called a "Relationship Type"

Entity-Relationship (ER) Diagrams [Chen 71]

Key:

- Entity Set
- Relationship Set
- Attribute
Entity-Relationship Model vs. Relational Model

A different data model than the relational model
- different constructs for modeling schemas and data
- DBMS systems (in various forms) have even been built on the ER model … though primarily used as a design tool

The relational model has:
- **Tables** (relations) with attributes, primary keys, foreign keys, rows, values

The ER model has:
- **Entities and Entity Sets** with attributes and entity identifiers (like keys)
- **Relationships and Relationship Sets** with cardinality constraints, roles, attributes, etc.

Mapping ER models to Relational Schemas

Employee(ssn, name, lot, home-dept)
**ProjectMember(ssn, number)**
Department(code, name, manager)
Project(number, name, startdate, enddate, budget, sponsor)

This mapping assumes:
- Employees have one home department
- Departments have one manager
- Employees can participate in many projects
**ER Cardinality Constraints**

An Employee can have **0 or 1** home Departments

**ER Cardinality Constraints**

A Department can have **0 or many** Employees
ER Cardinality Constraints

A Department must have **exactly one** Manager

and so on …
ER Cardinality Constraints

- Constraints are expressed over Entity Sets and Relationship Sets
- Constraints on the members of the corresponding Entities and Relationships

Does this satisfy the cardinality constraints?
ER Cardinality Constraints

- Constraints are expressed over Entity Sets and Relationship Sets
- Constraints on the members of the corresponding Entities and Relationships

Employee 1..* home 0..1 Department

What about now?

Employee Entities

Department Entities

And now?
ER Cardinality Constraints

• There are various notations used for writing cardinality constraints …

**Examples of “One to Many” constraints**

<table>
<thead>
<tr>
<th>one</th>
<th>many</th>
<th>zero..one</th>
<th>one..many</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>n</td>
<td>0:1</td>
<td>1:n</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Maximum cardinalities only  Minimum and maximum cardinalities

**Examples of “Many to Many” constraints**

<table>
<thead>
<tr>
<th>many</th>
<th>many</th>
<th>one..many</th>
<th>one..many</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>n</td>
<td>1:n</td>
<td>1:n</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Maximum cardinalities only  Minimum and maximum cardinalities
• Each instance of the relationship has a value for the attribute …

• Try all three locations for the attribute …

*Which one makes sense?
Because Employees have zero or one home departments, … start date will work as an Employee or home attribute. **What about startdate at Department?**

**Relationship Set Attributes**

**Employee**
- ssn
- name
- lot
- startdate

**Department**
- code
- name
- startdate

**Relationship**
- home

**What about now?**
- ... Since employees can have multiple home departments, it must be a relationship attribute.
Relationships can have role names

- An employee "manages" zero or one department
- A department is "managed by" exactly one employee

Note: some notations use the opposite side of the relationship set to specify cardinality and roles

• Entity sets can participate in different roles for the same relationship set
**Exercise …**

- Form groups of 2
- Draw an ERD for a database that stores information about Students, Faculty, Courses, and Course Offerings
  - Faculty can serve as a “course coordinator”
  - Faculty can be qualified to teach a course
  - Courses can have other courses as prerequisites
  - One or more faculty can teach course offerings
- Identify entity sets, attributes, and keys
- Identify relationships (and roles, if needed)
- Define cardinality constraints

---

**Weak Entity Sets**

In this model assume

- we need to record the insurance policies of employees
- we need to track dependents w.r.t. the policies
- we only need to store the name and date-of-birth of dependents (and nothing else)
- that, e.g., when employees leave, we no longer track their policies or dependents
Note that we only identify dependents through their corresponding employees

- we assume dependents of an employee have unique names
- but different employees could have dependents with the same name … since names are not guaranteed to be unique, e.g., “John Smith”
Weak Entity Sets

This is a “partial” key
(must be combined with the strong entity’s key, ssn, to identify the dependent)

This is a “strong” entity set

This is a “weak” entity set
(it would not be in the database if the strong entity were not present)

This is the “identifying” relationship

Cardinalities for an identifying relationship set

They must be like this
For Thursday

• Reading
  – Ch 2: Intro, 2.1-2.4.3, 2.5

• Be sure to know:
  – Group by, having, order by, subqueries
  – Next time we’ll talk about ternary relationships and converting ER diagrams to relational schemas

• Homework
  – Homework 3 assigned soon …