CSC 310
Database Theory and Implementation

Lecture Set # 3
Blase B. Cindric
University of Mount Union
Determining Multiplicities for Relationships

Given: A relationship between two entity sets E1 and E2.

Technique: Step 1 --

Consider a single entity from E1; can it be related to only one entity from E2, or can it be related to more than one entity from E2?

This determines whether the cardinality (the maximum #, or second symbol) is 1 or * on the side of the relationship CLOSEST to E2.

The direction is critical here!
Determining Multiplicities for Relationships

Technique: Step 2 --

Then consider a single entity from E2; can it be related to only one entity from E1, or to more than one entity from E1?

This determines the cardinality on the side of the relationship CLOSEST to E1.
Determining Multiplicities for Relationships

Example: NBA Basketball Teams and Players

Consider one Team: consists of one player or many players?

Consider one Player: plays for one team or many?

Note that the answers to these questions depend upon knowledge of real-world application area....
Tabular Representation of E-R Schema

We will use the Relational Database Model in our work this semester. In this model, data are represented as 2-dimensional tables. Once we have an E-R diagram that captures the essential features of the area of interest, we need to convert this into a series of tables.
Representing Entity Sets as Tables

Let $E$ be an entity set with attributes $a_1, a_2, \ldots, a_n$

We represent $E$ by a table with $n$ columns (each of which corresponds to one of the attributes of $E$).

Each entity in $E$ is represented by a row in this table.
Representing Many-to-Many Relationship Sets as Tables

Let $R$ be a many-to-many relationship set between two entity sets whose primary key attributes are $a_1, a_2, \ldots, a_m$, and that has descriptive attributes are $b_1, b_2, \ldots, b_n$.

We represent $R$ by a table with $m+n$ columns (one column for each of the $a_j$’s and $b_i$’s).

The contents of a table at any one time are called an *instance* of the entity set or relationship set it represents.
Consider this E-R diagram:

Notice that ShipsTo is a many-to-many relationship (why?)
## Converting E-R Diagrams to Tables

### Step 1: Represent each entity set as a table:

#### PARTS table:

<table>
<thead>
<tr>
<th>P#</th>
<th>PName</th>
<th>Color</th>
<th>Weight</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Nut</td>
<td>Red</td>
<td>15</td>
<td>Paris</td>
</tr>
<tr>
<td>P2</td>
<td>Bolt</td>
<td>Green</td>
<td>12</td>
<td>Athens</td>
</tr>
<tr>
<td>P3</td>
<td>Cam</td>
<td>Blue</td>
<td>20</td>
<td>London</td>
</tr>
<tr>
<td>P4</td>
<td>Cog</td>
<td>Red</td>
<td>17</td>
<td>London</td>
</tr>
</tbody>
</table>

#### SUPPLIERS table:

<table>
<thead>
<tr>
<th>S#</th>
<th>SName</th>
<th>City</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Bolts R Us</td>
<td>London</td>
<td>20</td>
</tr>
<tr>
<td>S2</td>
<td>Cam World</td>
<td>Paris</td>
<td>10</td>
</tr>
<tr>
<td>S3</td>
<td>Fastenate</td>
<td>Athens</td>
<td>30</td>
</tr>
</tbody>
</table>
Step 2: Represent each many-to-many relationship set as a table:

**SHIPS_TO table:**

<table>
<thead>
<tr>
<th>*</th>
<th>*</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P#</td>
<td>S#</td>
<td>QTY</td>
</tr>
<tr>
<td>P1</td>
<td>S1</td>
<td>200</td>
</tr>
<tr>
<td>P2</td>
<td>S1</td>
<td>700</td>
</tr>
<tr>
<td>P1</td>
<td>S2</td>
<td>300</td>
</tr>
<tr>
<td>P3</td>
<td>S2</td>
<td>500</td>
</tr>
<tr>
<td>P2</td>
<td>S3</td>
<td>450</td>
</tr>
<tr>
<td>P3</td>
<td>S3</td>
<td>50</td>
</tr>
</tbody>
</table>
Converting E-R Diagrams to Tables

Special Case: Multivalued Attributes:

If M is a multivalued attribute (of either an entity set or a relationship set), we form a new table TM with a column C that will hold one value at a time from M, and columns for the primary key of the entity set or relationship set that M comes from.

example: an e-mail address table:

```
*          *
Student#   e-mailAddress
11111111   helga@mountunion.edu
11111111   superswede@jones.ibm.co.se
11111111   thors_hammer@gmail.com
```
The Relational Database Model (Ch. 4)

All data are (conceptually) stored in *relations* (similar to tables, with some extra restrictions)

**Relational Database Structure**

**Terms:**
- Tables are called *relations*
- Rows or records are called *tuples* (short for n-tuples)
- Columns or fields are called *attributes*

Each tuple in a relation (row in a table) corresponds to one entity or one relationship between entities

ex. one student, one supplier, one shipment of a part to a supplier
Relational Database Structure

Example: Relation S (for Suppliers)

* 

<table>
<thead>
<tr>
<th>S#</th>
<th>SName</th>
<th>City</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Bolts R Us</td>
<td>London</td>
<td>20</td>
</tr>
<tr>
<td>S2</td>
<td>Cam World</td>
<td>Paris</td>
<td>10</td>
</tr>
<tr>
<td>S3</td>
<td>Fastenate</td>
<td>Athens</td>
<td>30</td>
</tr>
</tbody>
</table>

**Primary Key**: attribute(s) that uniquely identify a tuple
(no two tuples may have the same value in primary key)

**degree**: # of attributes in a relation (does not change over time)
(4 in this example instance)

**cardinality**: # of tuples in a relation (may change over time)
(3 in this example instance)
Relational Database Structure

*domain:* universe of possible values from which an attribute draws its actual values

*scalar:* individual value for an attribute in a tuple

cannot be decomposed in the relational model

(this rule is overlooked in most implementations -- ex. substring extraction)

Every attribute is defined on one underlying domain
Properties of Relations (Sec. 4.2.4)

1) relation name is distinct from all other relations in the DB
2) all attribute values are atomic (no multivalued attrs allowed)
3) each attribute has a distinct name from all others in the relation
   (note: the same attr name can be used in different relations)
4) all values in an attribute (in all tuples) come from the same domain
5) no duplicate tuples
   reason: sets don’t have duplicates in mathematics
   relations are based on set theory
   consequence: there is always a primary key for any relation (may have to be all the attrs, but a primary key will always exist!)
6) attributes are unordered
7) tuples are unordered
Properties of Relations (Sec. 4.2.4)

If a two-dimensional table satisfies all 7 of these properties, it is called a *normalized relation*, or just a *relation*.

Properties 2 and 5 (no multivalued attributes and no duplicate tuples) are the crucial ones for Relational DB design.

Implementations of the Relational Model (such as Microsoft Access, SQL Server, Oracle, etc.) typically do not enforce these properties, and work with tables, not relations.
Kinds of Relations

- Base relations
  actually stored on a secondary storage medium
- View (or virtual relation)
  derived from base relations, not actually stored
- Snapshot
  derived from base relations, stored
  a read-only copy of a relation at some point in time
  periodically refreshed by system
- Query results
  results of some data retrieval operation
- Intermediate results
  from nested queries
Translation Rules from E-R Model to Relational DB Schemas

Rule 1: For each entity set in the E-R model, create a relation schema by the same name

Rule 1-A: For each simple attribute in an entity set, create an attribute with the same name in the relation schema

Rule 1-B: Set the primary key of the relation schema to the same set of attributes as the primary key of the entity set

Rule 2: For each one-to-many relationship in the E-R model, add the primary key of the parent entity set to the relation schema for the child entity set

parent: entity set on the “1” side of the relationship
child: entity set on the “*” side of the relationship
Translation Rules from E-R Model to Relational DB Schemas

Rule 2-A: If a one-to-many relationship has attributes, add them to the child relation schema

Rule 3: For each one-to-one relationship in the E-R model, choose one entity set to act as parent and treat the other entity set as child; then apply rules 2 and 2-A

Rule 4: For each many-to-many relationship in the E-R model, create a new relation schema that will represent the relationship

Rule 4-A: Set the primary key of the relation schema to the primary keys of the entity sets that participate in the relationship
Translation Rules from E-R Model to Relational DB Schemas

Rule 5: For each multivalued attribute in the E-R model, define a new relation schema that contains one column for the multivalued attribute and also contains the primary key of the entity set or relationship set from which the multivalued attribute came.
Translate this E-R Diagram into a Relational DB Schema:

- **Student**
  - studentNum {PK}
- **Course**
  - courseNum {PK}
  - section {PK}
- **Faculty**
  - facIDNum {PK}

**Takes**
- 1 .. *

**Advises**
- 0 .. *

**Teaches**
- 1 .. 1

- 1 .. *
Another Example

Translate this E-R Diagram into Relational DB Schemas:

<table>
<thead>
<tr>
<th>Table</th>
<th>Relationship</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buyer</td>
<td>0 .. 1</td>
<td>property</td>
</tr>
<tr>
<td></td>
<td>1 .. *</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 .. *</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 .. 1</td>
<td></td>
</tr>
<tr>
<td>StaffMember</td>
<td>1 .. *</td>
<td>WorksAt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Branch</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Relational Data Model is concerned with:

- **Data Structure**
  - flat two-dimensional tables

- **Data Manipulation**
  - relational algebraic operations

- **Data Integrity**
  - two kinds of integrity ---
    - application-specific: Student #'s are 9-digit integers, gpa cannot be negative, etc.
    - general: applies to any application area
      - key based rules, seen later this semester….