SWEN 220
Mathematical Models of Software

Entity Relationship Modeling & Diagrams
Agenda

1. Conceptual / Logical / Physical Design of Databases
2. Entity & Attribute Basics
3. Relationships & Role Basics
4. Cardinality Constraints
5. Odds & Ends
Database Design Concepts
Database Design Levels

1. Requirements
   a. Why a DB? What's it for?
   b. Understand the data domain.

2. Conceptual DB Design
   a. Model of the data domain.
   b. Domain elements (entities) and associations (relationships).

3. Logical DB Design
   a. Convert conceptual model to relations (tables).
   b. Normalize relations to prevent operational anomalies.

4. Physical DB Design - the DBMS
   a. Organization of relations - data structures (typically on external storage).
   b. Mechanisms for performance enhancement (e.g., indexing).
Conceptual Design

An *entity* is:
- A uniquely identifiable "thing" capable of independent existence.
- Possibly corresponds to real world object (e.g., a Player, a Team)
- Possibly a concept (e.g., a Position like shortstop)
- Typically the nouns in our system (will become relations / tables).

A *relationship* is:
- The way entities are associated with each other.
- Usually verbs (or verb phrases).
- Example: Player (entity) PlaysFor (relationship) a Team (entity).
- These become foreign keys and/or mapping relations.
Conceptual Design Notation

Capture visually in some well-defined notation (Chen)
Typically the notation is graphic, not just text.
For us, Entity-Relationship Diagrams (ERDs).
   Include attributes (information about) each type of entity.
   Connect entities via relationships (which may have attributes).
   Business rules & constraints (cannot play on two teams).
   Convertible to tables with keys, etc.

Could use UML - and many modelers do!
ERD Notation & Drawing Tools

• We will use the Chen ERD notation.
• As noted in the syllabus, Lucidchart is a good drawing tool option as it supports Chen notation. Be sure to register for an academic account using your student@rit.edu email address.
• You are welcome to use other drawing/modeling tools for assignments so long as they follow Chen notation.
• If you are really stuck, use a scanned image of a neatly hand drawn diagram.
Entity & Attribute Basics
Entities

An *entity* is group of distinct objects or concepts.
An *entity instance* is a specific member of the entity set.
Entities have *attributes* of different types.
Entities

An *entity* is group of distinct objects or concepts.
An *entity instance* is a specific member of the entity set.
Entities have *attributes* of different types.
Entities

An *entity* is group of distinct objects or concepts. An *entity instance* is a specific member of the entity set. Entities have *attributes* of different types.
Entities

Also OK to simply list attributes next to the entity.

<table>
<thead>
<tr>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player</td>
</tr>
</tbody>
</table>

ssn  
birthdate  
name  
first  
last  
...
Entities

An *entity* is group of distinct objects or concepts. An *entity instance* is a specific member of the entity set. Entities have *attributes* of different types.

Simple attributes
Entities

An entity is a group of distinct objects or concepts. An entity instance is a specific member of the entity set. Entities have attributes of different types.

Composite Attributes
(attributes with internal structure)
Entities

An *entity* is group of distinct objects or concepts. An *entity instance* is a specific member of the entity set. Entities have *attributes* of different types.

```
Player

- ssn
- birthDate
- age
- name
- first
- last
- address
- street
- city
- state
- positions

Multi-valued Attributes
(may play multiple positions)
```
Entities

An *entity* is a group of distinct objects or concepts.
An *entity instance* is a specific member of the entity set.
Entities have *attributes* of different types.

![Diagram showing attributes of a Player entity]

**Derived Attributes**
(computable from other attributes)
Entities

An *entity* is a group of distinct objects or concepts.

An *entity instance* is a specific member of the entity set.

Entities have *attributes* of different types.

![Diagram of a player entity with attributes: birthDate, ssn, first, last, name, address, city, state, street, positions, age, Key Attribute(s) (unique instance identification).]
Entities

An *entity* is group of distinct objects or concepts. An *entity instance* is a specific member of the entity set. Entities have *attributes* of different types.

![Entity Diagram]

Key Attribute(s)
(may be compound)
Entities

An *entity* is group of distinct objects or concepts. An *entity instance* is a specific member of the entity set. Entities have *attributes* of different types.

An Entity Instance
Relationships & Role Basics
Relationships

Join two or more entities (typically 2)
Relationship name indicates the association purpose.
Relationships

Relationships are bidirectional:
  Given team, can find players on the roster.
  Given player, can locate the team the player is on.
Relationships

May have multiple relationships between entities:

Record history of the teams a player was ever on
Relationships can have attributes too!

When was a player on each team?
Relationships

And an entity may be in a relation to itself.

Relate players to their siblings (e.g., Ripken brothers)
Cardinality Constraints
Constraints

1. *Constraints* come in many forms:
   a. A player must be 18 years old to be on a team.
   b. Player addresses must be complete (no missing or NULL entries).

2. Among the most important are *cardinality constraints*:

   1. How many B's can associate with a given A under R?
   2. How many A's can associate with a given B under R?
   3. We need both the minimum and the maximum.
Cheezy Cardinality Example

1. Every person has a unique favorite food.
2. A food may be the favorite of 0 or more persons (I mean, who really likes Brussel sprouts?).
Cheezy Cardinality Example

1. Every person has a unique favorite food.
2. A food may be the favorite of 0 or more persons (I mean, who **really** likes Brussel sprouts?).

3. Read as "Person thru Favorite to exactly 1 Food."
Cheezy Cardinality Example

1. Every person has a unique favorite food.
2. A food may be the favorite of 0 or more persons (I mean, who really likes Brussel sprouts?).
3. Read as "Person thru Favorite to exactly 1 Food."
4. Read as "Food thru Favorite to 0 to N Persons."
Cheezy Cardinality Example

1. Every person has a unique favorite food.
2. A food may be the favorite of 0 or more persons (I mean, who *really* likes Brussel sprouts?).
3. Read as "Person thru Favorite to exactly 1 Food."
4. Read as "Food thru Favorite to 0 to N Persons."
5. In this second case, the upper limit is unbounded.
Digression on Notation

1. There is no universal standard for showing cardinality.
2. Three popular ways are:
   a. *Explicit lower and upper bounds.*
Digression on Notation

1. There is no universal standard for showing cardinality.
2. The three most popular ways are
   a. Explicit lower and upper bounds.
   b. *Upper bound* given. The *lower bound* is determined by context and/or documented separately. Our normal approach.

```
  +------------+      1      +-----------+
  |  Person    |   N   |  Favorite |  Food   |
  +------------+      +-----------+
```
Digression on Notation

1. There is no universal standard for showing cardinality.
2. Three popular ways are:
   a. Explicit lower and upper bounds.
   b. Upper bound given. The lower bound is determined by context and/or documented separately. Our normal approach.
   c. *Crows foot notation* - a graphical representation of the bounds where:
      i. is zero (minimum only).
      ii. is one (minimum or maximum).
      iii. is many (maximum only)
N's for different relations are independent. Thus a Team may have:

- 53 Players playing for it and
- 359 Players who played for it.

For Sibling, the N's are the same.
Full Sports Model with Cardinalities

- **Player**
  - `position`
  - `name` (first, last)
  - `birthDate`
  - `age`
  - `ssn`
  - `address` (street, city, state)

- **Team**
  - `league`
  - `name`
  - `address` (street, city, state)
  - `stadium`

- **PlayedFor**
  - `startDate`
  - `endDate`

- **PlaysFor**
  - `1`

- **Sibling**
  - `N`
A *relationship instance* shows an association involving *one entity instance* from each of the participating *entities*. Thus relationship instances *themselves* are always 1 to 1. One can think of this as "wiring" the entity instances together.
One-to-One (1:1) Relationships (Relatively Rare)

One person manages one department.

Person

<table>
<thead>
<tr>
<th>Manage</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 .. 1</td>
<td>0 .. 1</td>
</tr>
</tbody>
</table>

Person

| Joe |
| Jane |
| Jack |
| Janet |
| June |
| Jeff |

Manage

| Engineering |
| Finance |
| Production |
One-to-Many (1:N) Relationships

Players play for at most one Team; Teams have many Players

Player

Team

PlaysFor

1 .. N

0 .. 1

Player

T. Taylor

T. Brady

J. Nelson

M. Bennett

L. Alexander

S. McCoy

B. Starr

PlayFor

Team

Bills

Patriots

Packers
One-to-Many (1:N) Relationships

Players play for at most one Team; Teams have many Players
One-to-Many (1:N) Relationships

Players play for at most one Team; Teams have many Players

Player

1 .. N

PlaysFor

0 .. 1

Team

Player

T. Taylor
T. Brady
J. Nelson
M. Bennett
L. Alexander
S. McCoy
B. Starr

PlayFor

Team

Bills

Patriots

Packers
Many-to-Many (M:N) Relationships

Some Players played for many Teams; Teams had many Players
Many-to-Many (M:N) Relationships

Some Players played for many Teams; Teams had many Players

Player

1 .. N

PlayedFor

1 .. M

Team

---

<table>
<thead>
<tr>
<th>Player</th>
<th>PlayFor</th>
<th>Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. Kelly</td>
<td>●</td>
<td>Bills</td>
</tr>
<tr>
<td>B. Favre</td>
<td>●</td>
<td>Packers</td>
</tr>
<tr>
<td>M. Lynch</td>
<td>●</td>
<td>Seahawks</td>
</tr>
<tr>
<td>J. Elway</td>
<td>●</td>
<td>Broncos</td>
</tr>
<tr>
<td>J. Lofton</td>
<td>●</td>
<td>Vikings</td>
</tr>
<tr>
<td>B. Starr</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>

---
Many-to-Many (M:N) Relationships

Some Players played for many Teams; Teams had many Players

<table>
<thead>
<tr>
<th>Player</th>
<th>PlayFor</th>
<th>Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. Kelly</td>
<td>·</td>
<td>Bills</td>
</tr>
<tr>
<td>B. Favre</td>
<td>·</td>
<td>Packers</td>
</tr>
<tr>
<td>M. Lynch</td>
<td>·</td>
<td>Seahawks</td>
</tr>
<tr>
<td>J. Elway</td>
<td>·</td>
<td>Broncos</td>
</tr>
<tr>
<td>J. Lofton</td>
<td>·</td>
<td>Vikings</td>
</tr>
<tr>
<td>B. Starr</td>
<td>·</td>
<td></td>
</tr>
</tbody>
</table>
Odds & Ends
Possible ERD Extensions

**Weak entities**

1. Entities that must be associated with another entity.
2. Cannot exist on their own
3. Often turn relationships with attributes into weak entities.

```
Player     N  PlaysFor  1  Team
           
Player     1  Has      1  PlaysFor
           
PlaysFor   N  Has      1  Team
```
Possible ERD Extensions

Weak entities

*Inheritance*

Diagram of ERD extensions with entities and attributes.