System Concepts and Architecture

Rose-Hulman Institute of Technology
Curt Clifton

Data Model

- A set of concepts to describe
  - Database **structure**
  - **Basic operations** on the data

Categories of Data Models

- Conceptual
  - Closest to users' views
- Implementation
  - Intermediate level for programmers
- Physical
  - Actual hardware level

Database Schema

- A **description** of the database
- **Not** the actual data in it
- Tends to change seldom
- Shown with a Schema Diagram

Database State

- Actual content at an instant in time
- Every change results in a new state
- DBMS tries to ensure only **valid states** occur

Three-Schema Architecture

- Goals:
  - Support program-data independence
  - Represent multiple views of data

The Three Schemas

- Internal schema
  - Describes storage with physical data model
- Conceptual schema
  - Describes entire database structure
    with conceptual or implementation data model
- External schemas
  - Describe user views
    typically with same data model

Data Independence

- Two kinds:
  - Logical: change conceptual schema without changing external schemas
  - Physical: change internal schema without changing conceptual
- Just update **mappings**

Database System Architectures

- Centralized
  - All processing on one machine
  - Mainframe + dumb terminals
- Client-Server
  - Specialized *server* machines for each function
  - Smart *client* machines provide interfaces
  - Connected via some sort of network

Two Tier Client-Server

- Client runs UI and application programs
- Uses API to connect directly to DBMS
- Perhaps multiple DBMS

Three Tier Client-Server

- Intermediate layer
  - Application Server or Web Server
- Advantages
  - Security
  - Scalability
- Disadvantage
  - Complexity

Entity-Relationship Model

Rose-Hulman Institute of Technology

Curt Clifton

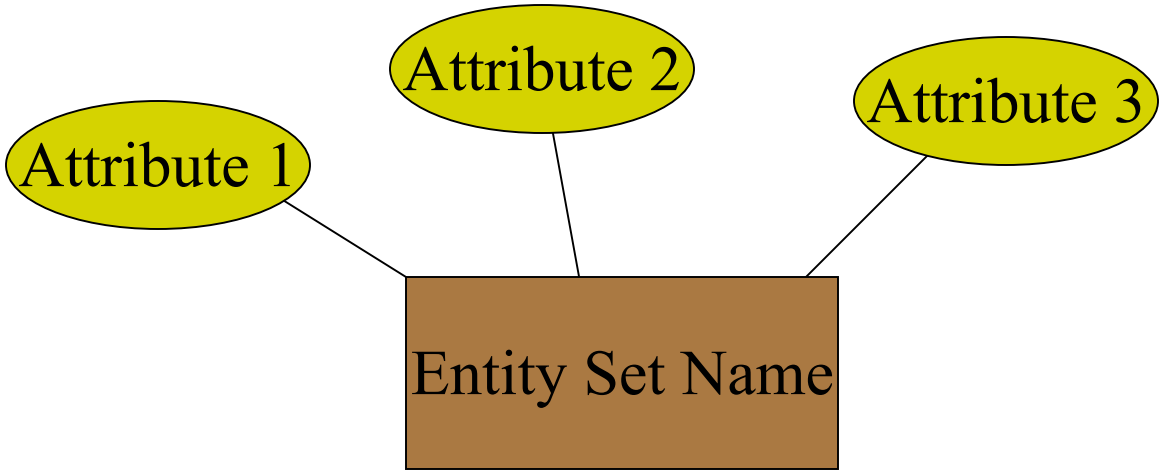
Entity-Relationship Model

- Lets us sketch database designs
  - Sketches called **ER Diagrams**
  - Simple enough share with customers
- Can convert sketches into implementations
  - Conversion is easy (with practice)

Entity Sets

- Entity: a “thing” that database tracks
- Entity set: a collection of similar entities
- Attribute: property of an entity
  - Simple values, like integers or strings
  - All entities in set have same properties (though different values)

Entity Set Notation



Entity set names are usually singular, i.e. "**Employee**" not "Employees"

Relationships

- Connect two (or more) entity sets
- Notation:

  - Try to make verbs read left-to-right, top-to-bottom

Values

- Entity set value:
  - The set of entities in it
- Relationship value:
  - A set of pairs (or triples, …) with one element from each related entity set

Multi-way Relationships

- Connect more than two entity sets
- Useful for more complex relationships

Relationship Constraints

- One-One:
  - Entity of first set can connect to just one entity in second set, and vice versa

Relationship Constraints

- One-Many:
  - Entity of first set can connect to just one entity in second set
  - Entity of second set can connect to many in first

  - Use N for arbitrary number greater than 1, or put specific number

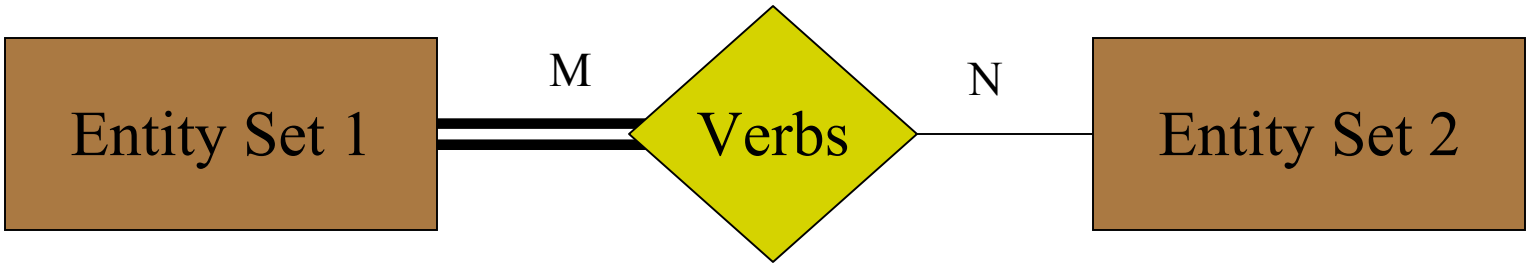
Relationship Constraints

- Many-Many:
  - An entity of either set can connect to many entities in the other set

  - Use N and M for arbitrary number greater than 1, or put specific number (or omit)

Relationship Constraints

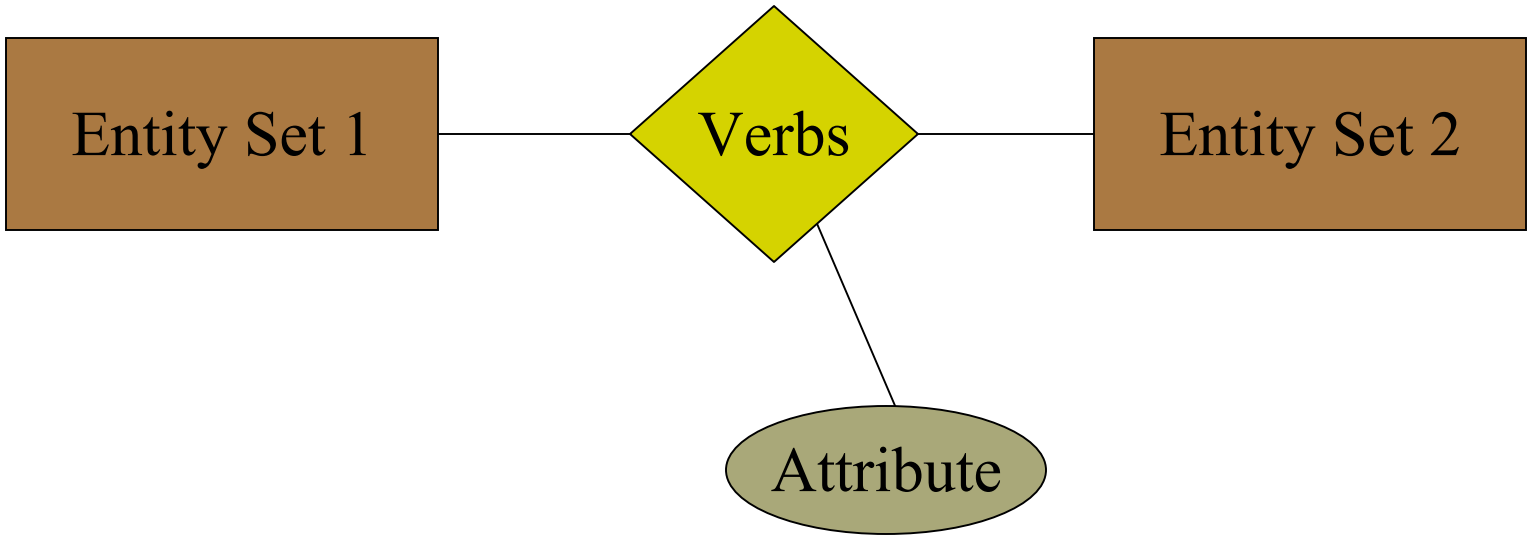
- Numbers on lines indicate maximums
- Can also show that every entity must participate



  - Every entity of first set must be related to at least one entity of the second set

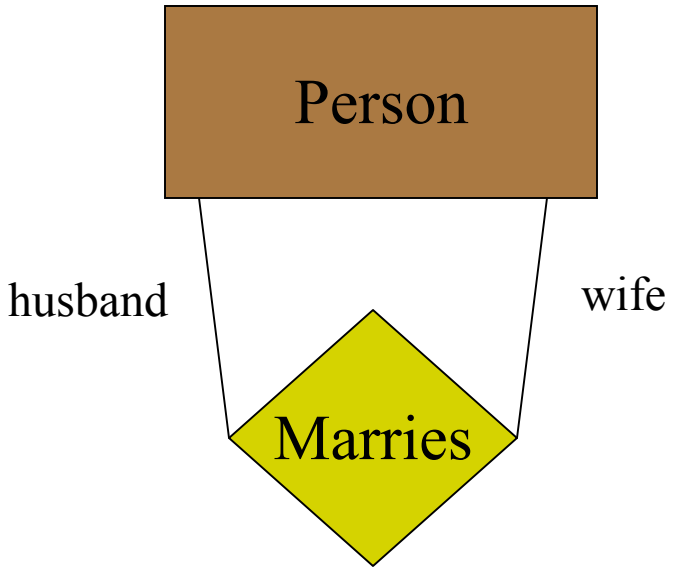
Attributes on Relationships

- Sometimes attribute is property of relationship instead of either entity



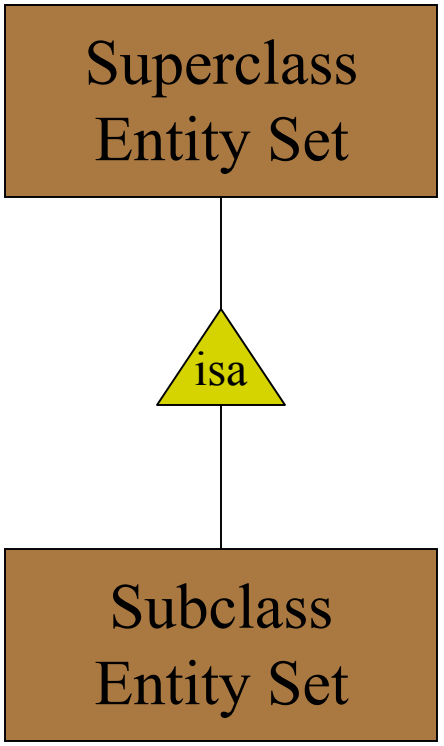
Recursive Relationships

- When an entity set is related to itself
- Label edges with *roles*
- Consider "Cousin Of"
  - Symmetrical
  - No clear role names



Subclasses

- Subclass = fewer entities
  - Have more properties
- Entity of subclass set is also in superclass set
  - Has all attributes of both sets



Keys

- Let us tell entities apart
- The key for an entity set is a **subset of the attributes** for that entity set, **such that no two entities agree** on all the attributes

Showing Keys

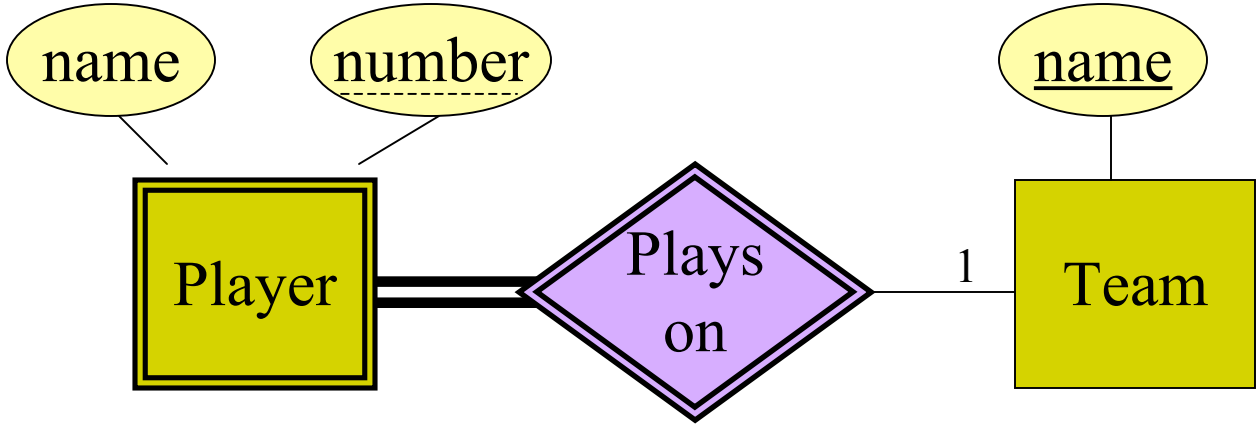
- Each entity must have a key
- Shown by underlining names of key attributes
- For subclass hierarchies:
  - Only the root entity set has a key
  - All entities in hierarchy use that key

Weak Entity Sets

- When even all the attributes aren't enough for a key…
- Use a many-one relationship to "borrow" an additional attribute for the key

Example Weak Entity Set

- Consider football players in a fantasy league
  - Is Name a key?
  - Is Number a key?
- Need Number + Team Played On



Practice with E-R Diagrams

- In groups of 2–3 work on HW Problem 3.21
  - On back of handout