CSCI 466/566

Databases - Conceptual Data Modeling and ER Diagrams

Jon Lehuta

Northern Illinois University

August 28, 2015

Data Models

- A means of describing the structure of data
- A set of operations that manipulate the data (for data models that are implemented)
- Types of data models
  - Conceptual data model
  - Logical data models - relational, network, hierarchical, inverted list, or object-oriented

Conceptual Data Model

- Shows the structure of the data including how things are related
- Communication tool
- Independent of commercial DBMSes
- Relatively easy to learn and use
- Helps show the semantics or meaning of the data
- Graphical representation
- Entity-Relationship Model is very common

Logical Data Models - Relational

Relational Data Model

- Data is stored in relations (tables). These tables have one value per cell.
- Based upon a mathematical model.
- First presented by E. F. Codd in early 1970's.
- Commercial relational data models include DB2, Oracle, Ingress, MySQL, Microsoft Access

Logical Data Models - Network

Network Data Model

- Data is stored in records (vertices) and associations between them (edges)
- Complex model
- Based upon a model called CODASYL
- Designed by a committee back in the 1970′s.
- Commercial DBMSes that have used this include IDMS and TOTAL.

Logical Data Models - Hierarchical

Hierarchical Data Model

- Data is stored in a tree structure with parent/child relationships
- First commercial DBMS created by IBM in late 1960's used this.
- Commercial DBMSes that have used this include IMS and System 2000.

Logical Data Models - Inverted List

Inverted List

- Tabular representation of the data using indices to access the tables
- Almost relational, but it allows for non-atomic data values, which are not allowed in relations
- One commercial DBMS that used the inverted list model was called ADABAS.

Logical Data Models - Object Oriented

Object Oriented

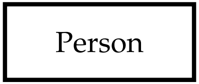
- Data stored as objects which contain
  - Identifier
  - Name
  - Lifetime
  - Structure
- Commercial Object Oriented (OO) DBMSes include O2 (Ardent) and ObjectStore

Entity-Relationship Model

- First introduced in 1976 by Peter P. Chen
- Meant to be simple and easy to read.
- Should be able to convey the design both to database designers and unsophisticated users

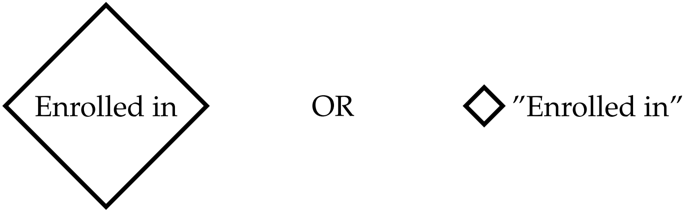
Entities

- Principle objects about which information is kept - These are the *things* we store data about.
- If you look at the ER Diagram like a spoken language, the entities are nouns - Person, place, thing, event.
- When drawn on the ER diagram, entities are shown as rectangles with the name of the entity inside.



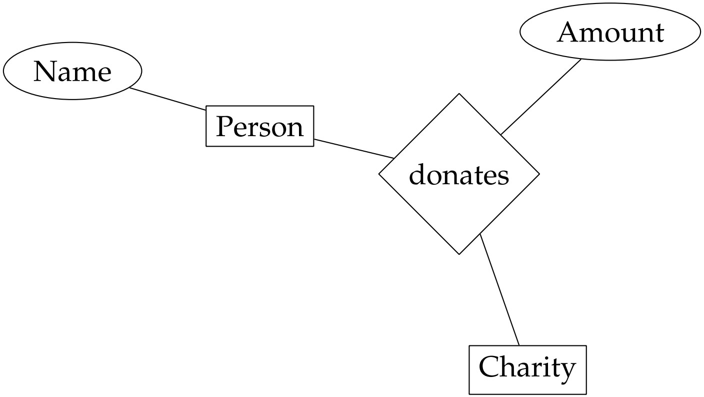
Relationships

- Relationships connect one or more entities together to show an association.
- A relationship *cannot* exist without at least one associated entity.
- Graphically represented as a diamond with the name of the relationship inside, or just beside it.



Attributes

- Characteristics of entities **OR** of relationships
- Represent some small piece of associated data
- Represented by either a rounded rectangle or an oval.



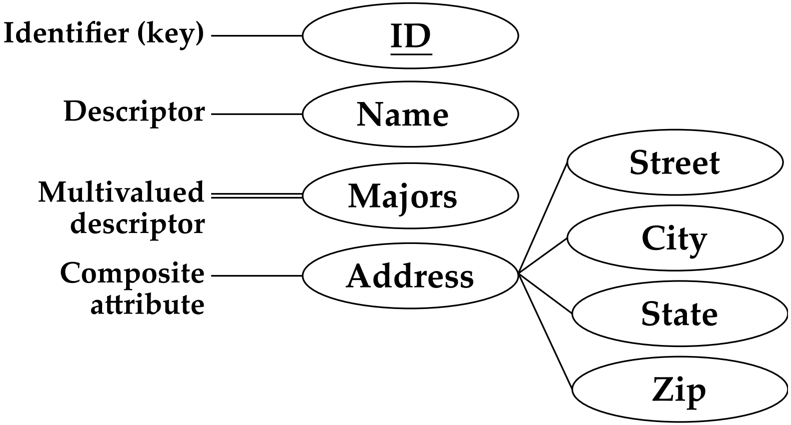
Attributes on Entities

When an attribute is attached to an entity, it is expected to have a value for every instance of that entity, unless it is allowed to be null. For instance, in the diagram on the last page, Name was an attribute of Person. Every person that we store data about will have a value for Name.

Attributes on Relationships

When an attribute is attached to a relationship, it is only expected to have a value when the entities involved in the relationship come together in the appropriate way.
In the diagram from before, the Amount attribute is attached to the donates relationship, which connects the Person and Charity entities. Amount will have one value for each time a Person donates to a Charity, denoting how much that person donated to the charity. It will not necessarily have a value for a given person, or a given charity. This can be referred to as the *intersection data*.

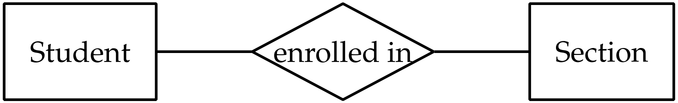
Types of Attribute



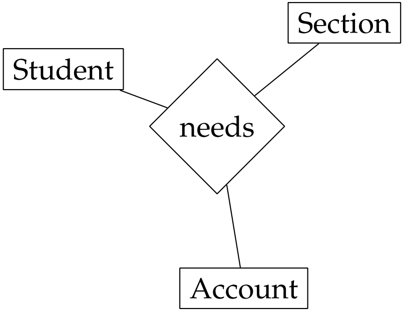
Degree of a Relationship

The **degree** of a relationship is the number of entities associated with it.

- binary

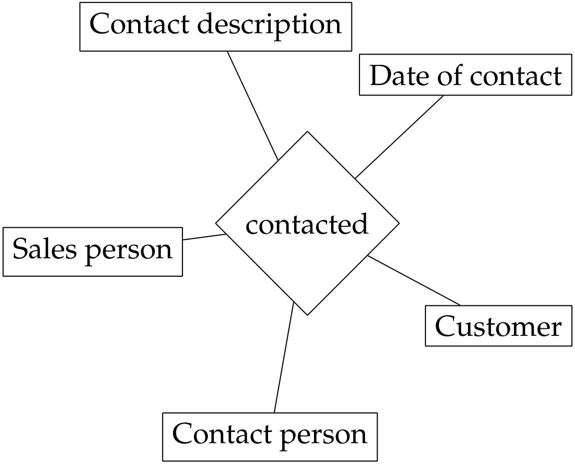


- ternary



Degree of a Relationship

There is no limit to how many entities there can be in a relationship. After binary, and ternary, we start to call the relationships *n-ary*, where *n* is the degree.

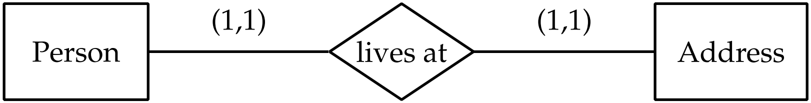


Connectivity of a Relationship

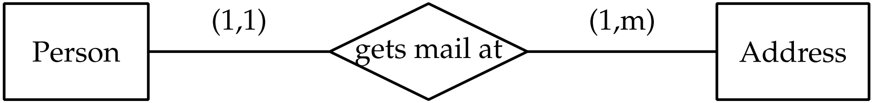
- A constraint of the mapping of associated entities
- Written as (*minimum*, *maximum*).
- Minimum is usually zero or one.
- Maximum is a number (commonly one) or can be a letter denoting *many*.
- The actual number is called the *cardinality*.

Connectivity of a Relationship

- one-to-one

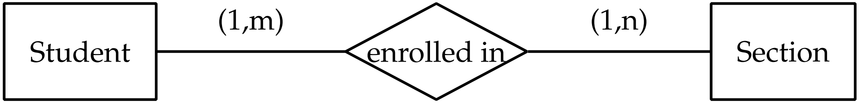


- one-to-many

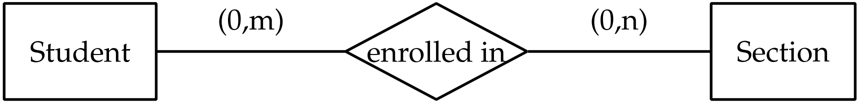


Connectivity of a Relationship

- mandatory many-to-many

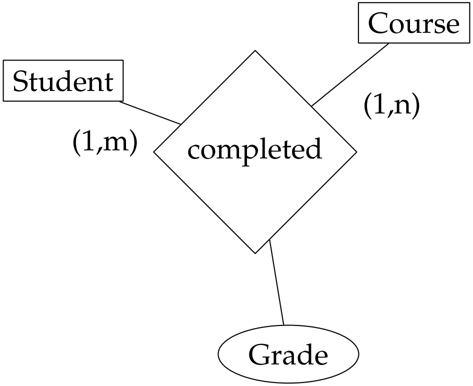


- optional many-to-many



Attributes on Relationships (revisited)

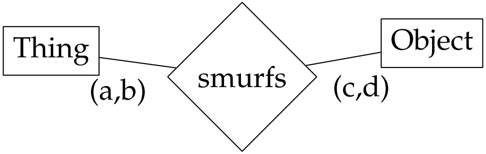
- Must be on a many-to-many relationship. (1-many and 1-to-1 relationships should have the attribute on one of the entities involved.
- intersection data
- Someone needs to know **all** of the associated entities to access the attribute.



Reading Cardinalities

For binary relationships:

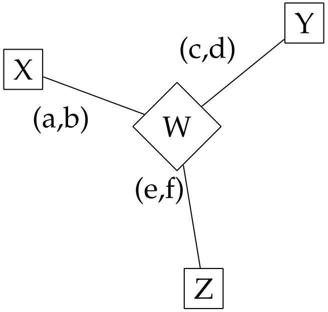
- For each Thing that smurfs, there are a minimum of *c*, and a maximum of *d* Objects.
- For each Object that smurfs/is smurfed, there is a minimum of *a* and a maximum of *b* Things.



Reading Cardinalities

For higher degree relationships:

- For each appropriate combination of X and Y in relationship W, there are a minimum of *e* and a maximum of *f* Z.
- For each appropriate combination of X and Z in relationship W, there are a minimum of *c* and a maximum of *d* Y.
- For each appropriate combination of Y and Z in relationship W, there are a minimum of *a* and a maximum of *b* X.

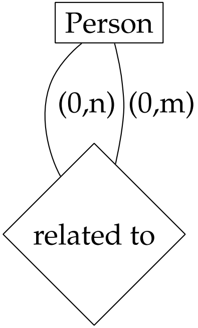


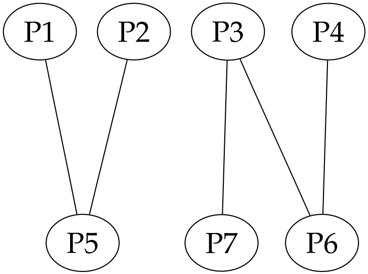
Recursive Relationships

It is possible for an entity to have a relationship with itself. This is called a recursive relationship. It makes more sense if you think of entities as collections of objects of their appropriate type.

Recursive Relationships - Many-To-Many

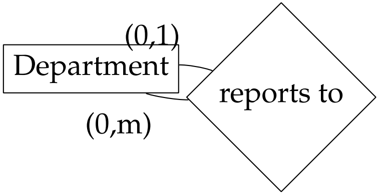
- A many-to-many recursive relationship means that the objects are arranged in a network structure
- Notice that the minimum is 0 on both sides. This is important.

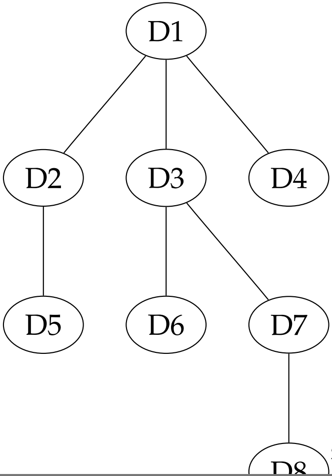




Recursive Relationships - One-To-Many

- A one-to-many recursive relationship means that the objects are arranged in a tree structure
- Notice that the minimum is still 0 on both sides. This is important.





Entity or Attribute?

Sometimes it isn't clear whether something should be an entity or an attribute of some other entity. Usually the decision will come down to how complicated it is to store the data, and how important it is. If it ends up being used in multiple places, it might be a clue that you should use an entity.

More to Come

The rules here form the basis of the ER Diagram model. There are more tools available, and we will be going over them next time, but you should be able to model most situations with just what we've gone over today.