DAMG 6210
Database Design

Nik Bear Brown
@NikBearBrown

Entity Relationship Model (ERM)

Topics

- Entity Relationship Model (ERM)
- Entity Sets
- Relationship Sets
- Design Issues
- Mapping Constraints
- Keys
- E-R Diagram
- Extended E-R Features
- Design of an E-R Database Schema
- Reduction of an E-R Schema to Tables

Database Design Process

- Requirement collection and analysis

  DB requirements and functional requirements

- Conceptual DB design using a high-level model

  Easier to understand and communicate with others

- Logical DB design (data model mapping)

  Conceptual schema is transformed from a high-level data model into implementation data model

- Physical DB design

  Internal data structures and file organizations for DB are specified

Conceptual data model

- This is the highest level ER model in that it contains the least granular detail but establishes the overall scope of what is to be included within the model set. The conceptual ER model normally defines master reference data entities that are commonly used by the organization.

Logical data model

- A logical ER model does not require a conceptual ER model, especially if the scope of the logical ER model includes only the development of a distinct information system. The logical ER model contains more detail than the conceptual ER model. In addition to master data entities, operational and transactional data entities are now defined. The details of each data entity are developed and the entity relationships between these data entities are established. The logical ER model is however developed independent of technology into which it is implemented.

Physical data model

- One or more physical ER models may be developed from each logical ER model. The physical ER model is normally developed to be instantiated as a database.

- Tables, queries, etc. in an actual database.

Conceptual Design

- Conceptual design: (ER Model is used at this stage.)
  - What are the entities and relationships in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the integrity constraints or business rules that hold?
  - A database `schema' in the ER Model can be represented pictorially (ER diagrams).
  - An ER diagram can be mapped into a relational schema.

Entity-Relationship Model (ERM)

- Entity Sets
- Relationship Sets
- Design Issues
- Mapping Constraints
- Keys
- E-R Diagram
- Extended E-R Features
- Design of an E-R Database Schema
- Reduction of an E-R Schema to Tables

E-R Model Constructs

- Entities:
  - Entity instance–person, place, object, event, concept (often corresponds to a row in a table)
  - Entity Type–collection of entities (often corresponds to a table)
- Relationships:
  - Relationship instance–link between entities (corresponds to primary key–foreign key equivalencies in related tables)
  - Relationship type–category of relationship…link between entity types
- Attributes:
  - Properties or characteristics of an entity or relationship type (often corresponds to a field in a table)

Entities

- **Entity** – a person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data

- **Entity type** – a collection of entities that share common properties or characteristics

- **Entity instance** – A single occurrence of an entity type

An Entity…

- SHOULD BE:
  - An object that will have many instances in the database
  - An object that will be composed of multiple attributes
  - An object that we are trying to model
- SHOULD NOT BE:
  - A user of the database system
  - An output of the database system (e.g., a report)

ERM Definitions

- An entity is an object in an abstract world.
- An attribute of an entity can have a value from a value set (domain)
- Each entity belongs to some one entity type s.t. entities in one entity type have the same attributes (so each entity type is a set of similar entities).

ERM Definitions

- A key attribute of an entity type is one whose value uniquely identifies an entity of that type.
- A combination of attributes may form a composite key.
- If there is no applicable value for an attribute that attribute is set to a null value.

Entity Sets

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An *entity* is an object that exists and is distinguishable from other objects.
  - Example: person, tweet, company, event, film (a thing, a noun)
- Entities have *attributes*
  - Example: people have *names* and *addresses*
- An *entity set* is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, tweets

ER Model Basics

- Entity: Real-world object distinguishable from other objects. An entity is described (in DB) using a set of attributes.
- Entity Set: A collection of similar entities. E.g., all employees.
- All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)
- Each entity set has a key.
- Each attribute has a domain.

ER Model Example - Twitter

We want to model a Twiiter user and a tweet?

- Entities?
- Entity Sets?
- entity attributes. (Until we consider ISA hierarchies, anyway!)
- Each entity set has a key.
- Each attribute has a domain.

Twitter API

https://dev.twitter.com/

https://dev.twitter.com/rest/public

https://dev.twitter.com/rest/public/search

Twitter API JSON

https://dev.twitter.com/rest/reference/get/statuses/user_timeline

https://api.twitter.com/1.1/statuses/user_timeline.json

Social Network Applications

Accounts are free

Using Twitter

Skills: familiarity with the Twitter user interface and major features, using the hashtag (#) and at-sign (@), searching and tweeting images and videos

Concepts: evolution of Twitter applications and access modes, citizen journalism, trending topics and finding people as well as information

Watch this video

http://commoncraft.com/twitter-search

Two special characters

@username: specify a particular Twitter user

Hashtag (#)*:* tag posts on a given topic

@username: at beginning of tweet is private DM

140 character limit

Twitter Resources

- Twitter support:
  https://support.twitter.com/
- A video introduction to Twitter:
  http://screencast.com/t/Sf1wAW8ta
- A “Prezi” of the above video:
  http://bit.ly/fQ3pbK
- Book about the use of Twitter in business by tech journalist Julio Ojeda-Zapata:
  http://yourtech.typepad.com/twitinbiz/
- Interview of Juilio Ojeda Zapata on his use of Twitter as a journalist (audio and transcript).
  http://www.onthemedia.org/transcripts/2008/08/22/06
- Introduction to Twitter in business:
  http://business.twitter.com/
- Twitter search in plain English (3m 19s video):
  http://commoncraft.com/twitter-search
- Talk by Clay Shirky, which includes examples of Twitter in citizen journalism:
  http://cis275topics.blogspot.com/2010/09/how-social-media-can-make-history.html
- Interview of Twitter co-founders on their backgrounds, Twitter evolution and plans for the future:
  http://cis275topics.blogspot.com/2010/09/evolution-and-impact-of-twitter.html

Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
- *Domain* – the set of permitted values for each attribute
- Attribute types:
  - *Simple* and *composite* attributes.
  - *Single-valued* and *multi-valued* attributes
    - E.g. multivalued attribute: *phone-numbers*
  - *Derived* attributes
    - Can be computed from other attributes
    - E.g. *age*, given date of birth

Attributes

- Attribute–property or characteristic of an entity or relationship type
- Classifications of attributes:
  - Required versus Optional Attributes
  - Simple versus Composite Attribute
  - Single-Valued versus Multivalued Attribute
  - Stored versus Derived Attributes
  - Identifier Attributes

ER Model Basics

- *Relationship*: Association among two or more entities. e.g., Miley works in the Music Store.
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1 in E1, ..., en in En
    - Same entity set could participate in different relationship sets, or in different “roles” in same set.

Relationship Sets

- A relationship is an association among several entities (e.g. Banks have customers, teachers have students)
- A *relationship* set is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

where $(e_1, e_2, \ldots, e_n)$ is a relationship

Relationship Sets

- A relationship is an association among several entities (e.g.
- A *relationship* set is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

where $(e_1, e_2, \ldots, e_n)$ is a relationship

Mathematical Definition of a Relation

- Consider two sets, D_1 & D_2, where $D_1 = \{2, 4\}$ and $D_2 = \{1, 3, 5\}$.
- Cartesian product, $D_1 \times D_2$, is set of all ordered pairs, where first element is member of D_1 and second element is member of D_2.

$$D_1 \times D_2 = \{(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)\}$$

- Alternative way is to find all combinations of elements with first from D_1 and second from D_2.

Mathematical Definition of a Relation

- Any subset of Cartesian product is a relation; e.g.

  $R = \{(2, 1), (4, 1)\}$

- May specify which pairs are in relation using some condition for selection; e.g.
  - second element is 1:

    $R = \{(x, y) \mid x \hat{I} D_1, y \hat{I} D_2, \text{ and } y = 1\}$
  - first element is always twice the second:

    $S = \{(x, y) \mid x \hat{I} D_1, y \hat{I} D_2, \text{ and } x = 2y\}$

Mathematical Definition of a Relation

- Consider three sets D_1, D_2, D_3 with Cartesian Product $D_1 \times D_2 \times D_3$; e.g.

$D_1 = \{1, 3\}$ $D_2 = \{2, 4\}$ $D_3 = \{5, 6\}$

$D_1 \times D_2 \times D_3 = \{(1,2,5), (1,2,6), (1,4,5), (1,4,6), (3,2,5), (3,2,6), (3,4,5), (3,4,6)\}$

- Any subset of these ordered triples is a relation.

Mathematical Definition of a Relation

- Cartesian product of n sets $(D_1, D_2, \ldots, D_n)$ is:

$$D_1 \times D_2 \times \ldots \times D_n = \{(d_1, d_2, \ldots, d_n) \mid d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n\}$$

usually written as:

$$\prod_{i=1}^{n} D_i$$

- Any set of n-tuples from this Cartesian product is a relation on the n sets.

Keys

- When you have your preliminary entities and attributes defined, you can start thinking about keys.
- There are several types of keys:
  - Primary Keys
  - Candidate Keys
  - Natural Keys
  - Composite Keys
  - Surrogate Keys

Primary Keys

- A primary key uniquely identifies a row of data.
- A primary key must be unique for every row (that is, it can never repeat in the table that will result from the entity).
- For instance, a student ID can uniquely identify an individual student and the data associated with him or her.
- Every entity should have a primary key.

Candidate Keys

- A candidate key is an attribute or attributes of an entity that have the potential to become a primary key.
- Candidate keys are not actual keys, but are a list of attributes that should be considered when choosing the primary key.

Natural Keys

- There are basically two ways of making keys: natural and surrogate.
- Natural keys are keys formed by using an attribute that “naturally” belongs to the entity, such as a student ID or a phone number.

Composite Keys

- Composite keys are keys composed of more than one attribute.
- For example, to get a unique designation of a course section, it is necessary to combine the quarter, the year, and the item number.
- Composite keys are *one* key made out of many parts.

Surrogate Keys

- Surrogate keys are keys that have no meaning.
- Often they are just integers incremented row by row.
- They can also be things such as time stamps of auto-generated IDs.

Relational Keys

- Superkey
  - An attribute, or set of attributes, that uniquely identifies a tuple within a relation.

- Candidate Key
  - Superkey (K) such that no proper subset is a superkey within the relation.
  - In each tuple of R, values of K uniquely identify that tuple (uniqueness).
  - No proper subset of K has the uniqueness property (irreducibility).

Relational Keys

- Primary Key
  - Candidate key selected to identify tuples uniquely within relation.
- Alternate Keys
  - Candidate keys that are not selected to be primary key.
- Foreign Key
  - Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.

Integrity Constraints

- Null
  - Represents value for an attribute that is currently unknown or not applicable for tuple.
  - Deals with incomplete or exceptional data.
  - Represents the absence of a value and is not the same as zero or spaces, which are values.

Integrity Constraints

- Entity Integrity
  - In a base relation, no attribute of a primary key can be null.

- Referential Integrity
  - If foreign key exists in a relation, either foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be wholly null.

Degree of a Relationship Set

- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are *binary* (or degree two). Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets.
- Relationships between more than two entity sets are rare. Most relationships are binary.

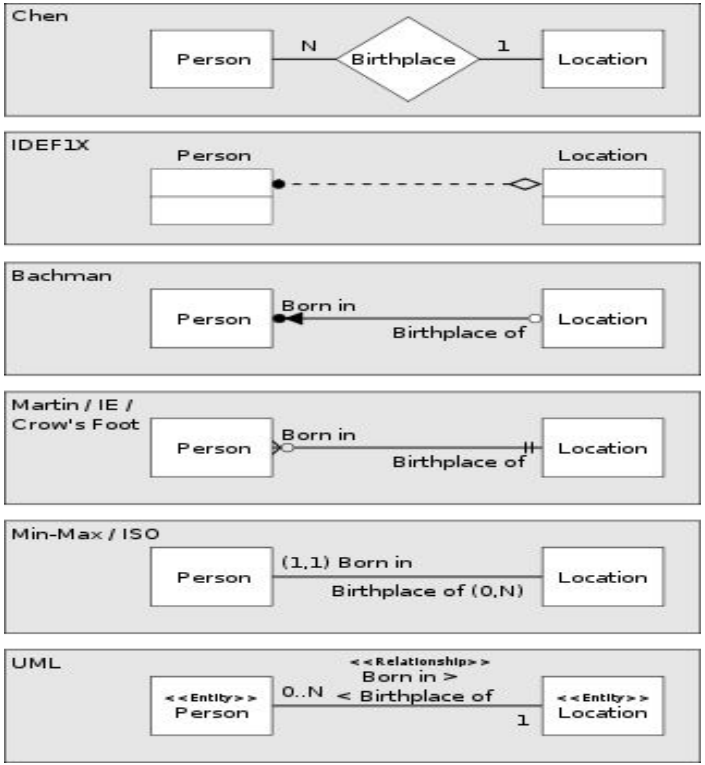
Role naming

- It is common to name roles with phrases such as is the owner of and is owned by. Correct nouns in this case are owner and possession. Thus person plays the role of owner and car plays the role of possession rather than person plays the role of, is the owner of, etc.
- The use of nouns has direct benefit when generating physical implementations from semantic models. When a person has two relationships with car then it is possible to generate names such as owner_person and driver_person, which are immediately meaningful

Mapping Cardinalities

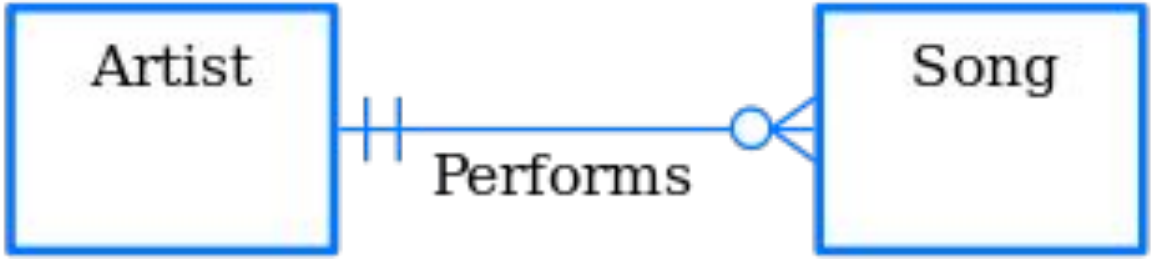
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities



- Various methods of representing the same one to many relationship. In each case, the diagram shows the relationship between a person and a place of birth: each person must have been born at one, and only one, location, but each location may have had zero or more people born at it.

Mapping Cardinalities

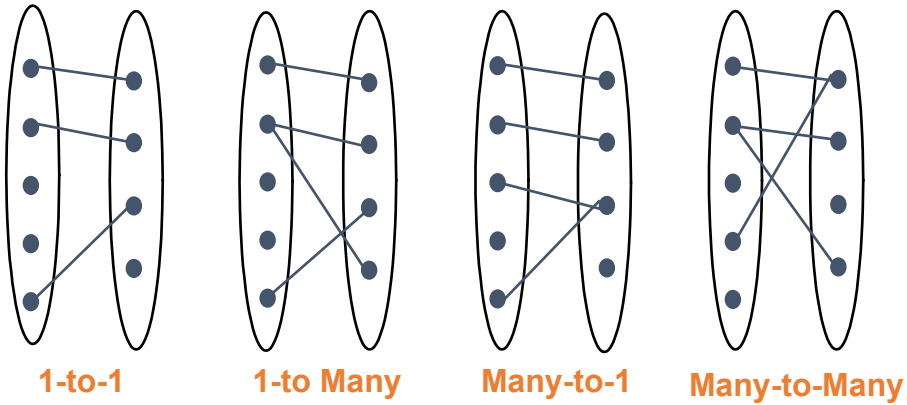


- Two related entities shown using Crow's Foot notation. In this example, an optional relationship is shown between Artist and Song; the symbols closest to the song entity represents "zero, one, or many", whereas a song has "one and only one" Artist. The former is therefore read as, an Artist (can) perform(s) "zero, one, or many" song(s).

Mapping Cardinalities

- Bachman notation
- Barker's Notation
- EXPRESS
- Martin notation
- (min, max)-notation of Jean-Raymond Abrial in 1974
- UML class diagrams
- Merise
- Object-Role Modeling

Key Constraints



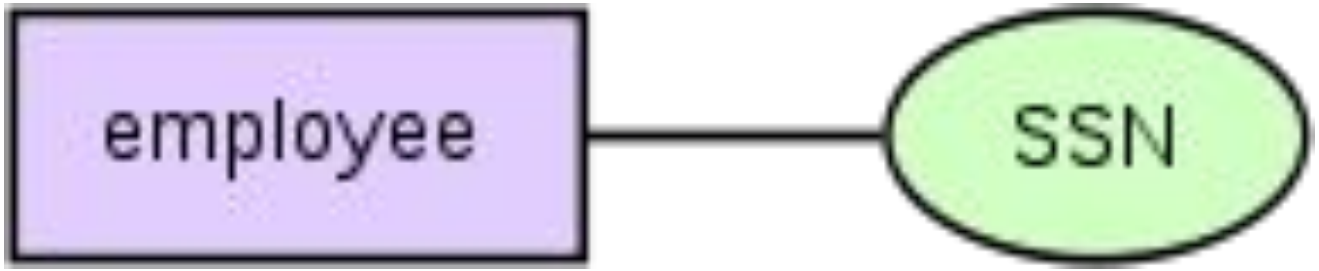
E-R Diagrams

- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Ellipses represent attributes
- Double ellipses represent multivalued attributes.
- Dashed ellipses denote derived attributes.
- Underline indicates primary key attributes

E-R Diagrams

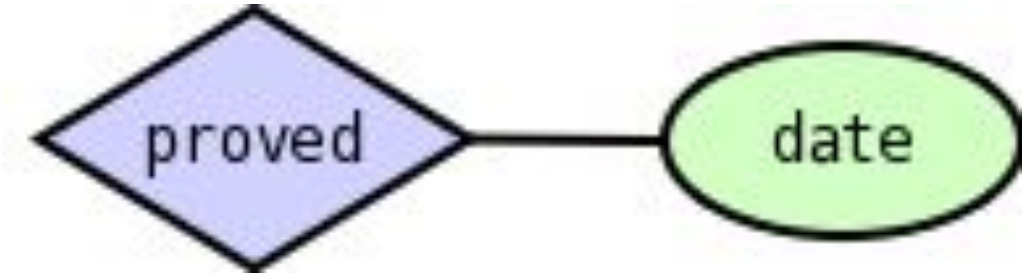
- Two related entities

E-R Diagrams



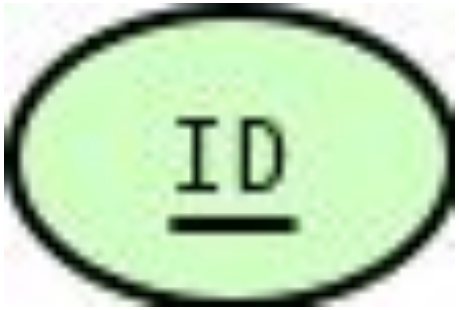
- An entity with an attribute

E-R Diagrams



- A relationship with an attribute

E-R Diagrams



- Primary key

Roles

- Entity sets of a relationship need not be distinct
- The labels “manager” and “worker” are called roles; they specify how employee entities interact via the works-for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship

Structural Constraints

- Structural constraints of a relationship type:
  - **Cardinality ratio:** Limits the number of relationship instances an entity can participate in, eg. **1:1, 1:N, M:N**
  - **Participation constraint:** If each entity of an entity type is **required** to participate in some instance of a relationship type, then that participation is **total**; otherwise, it is **partial**.

Structural Constraint Min, Max

- A more complete specification of the structural constraint on a relationship type can be given by the integer pair (min, max), which means an entity must participate in at least min and at most max relationship instances.

Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.
- E.g.: One-to-one relationship:
  - A customer is associated with at most one loan via the relationship *borrower*
  - A loan is associated with at most one customer via *borrower*

One-To-Many Relationship

- In the one-to-many relationship (e.g. a loan is associated with at most one customer via *borrower*, a customer is associated with several (including 0) loans via *borrower)*

Many-To-One Relationships

- In a many-to-one relationship a loan is associated with several (including 0) customers via *borrower*, a customer is associated with at most one loan via *borrower*

Many-To-Many Relationship

- A customer is associated with several (possibly 0) loans via borrower
- A loan is associated with several (possibly 0) customers via borrower

Cardinality of Relationships

- One-to-One
  - Each entity in the relationship will have exactly one related entity
- One-to-Many
  - An entity on one side of the relationship can have many related entities, but an entity on the other side will have a maximum of one related entity
- Many-to-Many
  - Entities on both sides of the relationship can have many related entities on the other side

Strong vs. Weak Entities, and Identifying Relationships

- Strong entity
  - exists independently of other types of entities
  - has its own unique identifier
    - identifier underlined with single line
- Weak entity
  - dependent on a strong entity (identifying owner)…cannot exist on its own
  - does not have a unique identifier (only a partial identifier)
  - entity box and partial identifier have double lines
- Identifying relationship
  - links strong entities to weak entities

Participation of an Entity Set in a Relationship Set

- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g. participation of *loan* in *borrower* is total
    - every loan must have a customer associated to it via borrower
- Partial participation: some entities may not participate in any relationship in the relationship set
  - E.g. participation of *customer* in *borrower* is partial

Keys

- A *super key* of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A *candidate key* of an entity set is a minimal super key
  - *Customer-id* is candidate key of *customer*
  - *account-number* is candidate key of *account*
- Although several candidate keys may exist, one of the candidate keys is selected to be the *primary key*.

Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (*customer-id, account-number*) is the super key of *depositor*
  - *NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.*
    - E.g. if we wish to track all access-dates to each account by each customer, we cannot assume a relationship for each access. We can use a multivalued attribute though
- Must consider the mapping cardinality of the relationship set when deciding the what are the candidate keys
- Need to consider semantics of relationship set in selecting the *primary key* in case of more than one candidate key

Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g. an arrow from *works-on* to *job* indicates each employee works on at most one job at any branch.

Cardinality Constraints on Ternary Relationship

- If there is more than one arrow, there are two ways of defining the meaning.
  - E.g a ternary relationship *R* between *A*, *B* and *C* with arrows to *B* and *C* could mean
  - 1. each *A* entity is associated with a unique entity from *B* and *C* or
  - 2. each pair of entities from (*A, B*) is associated with a unique *C* entity, and each pair (*A, C*) is associated with a unique *B*
  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow

Binary Vs. Non-Binary Relationships

- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g. A ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g. only mother being know)
  - But there are some relationships that are naturally non-binary
    - E.g. *works-on*

Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:

    1. R_A, relating *E* and *A*
    2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*

  - Create a special identifying attribute for *E*
  - Add any attributes of *R* to *E*
  - For each relationship (a_i , b_i , c_i) in *R,* create

    1. a new entity e_i in the entity set *E*
    2. add (e_i , a_i) to R_A
    3. add (e_i , b_i) to R_B
    4. add (e_i , c_i) to R_C

Converting Non-Binary Relationships

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of *R*
    - *Exercise: add constraints to the relationships* R_A, R_B *and* R_C to ensure that a newly created entity corresponds to exactly one entity in each of entity sets *A, B* and *C*
  - We can avoid creating an identifying attribute by making E a weak entity set (described shortly) identified by the three relationship sets

Design Issues

- Use of entity sets vs. attributes
  Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.
- Use of entity sets vs. relationship sets
  Possible guideline is to designate a relationship set to describe an action that occurs between entities
- Binary versus *n*-ary relationship sets
  Although it is possible to replace any nonbinary (*n*-ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- Placement of relationship attributes

Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.

Weak Entity Sets

- An entity set that does not have a primary key is referred to as a *weak entity set*.
- The existence of a weak entity set depends on the existence of a *identifying entity set*
  - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - Identifying relationship depicted using a double diamond
- The *discriminator (or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

Weak Entity Set Examples

- In a university, a *course* is a strong entity and a *course-offering* can be modeled as a weak entity
- The discriminator of *course-offering* would be *semester* (including year) and *section-number* (if there is more than one section)
- If we model *course-offering* as a strong entity we would model *course-number* as an attribute.

  Then the relationship with *course* would be implicit in the *course-number* attribute

Specialization

- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (E.g. *customer* "is a" *person*).
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

Entity vs. Attribute

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
    - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
    - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).

E-R Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

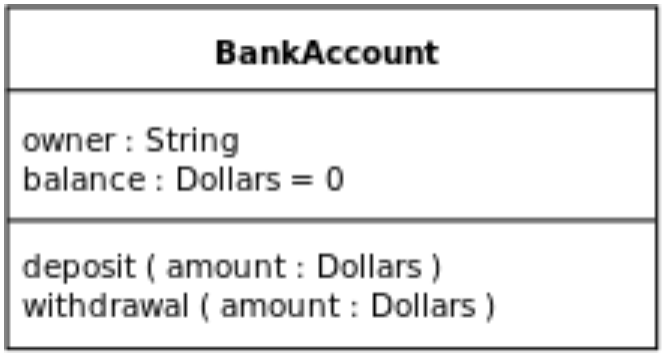
UML

- UML: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.
- It was created and developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software during 1994–95 with further development led by them through 1996.

UML Class Diagrams

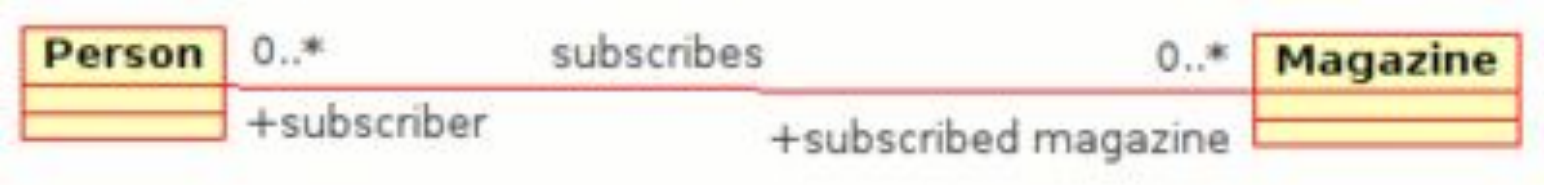
- Entity sets are shown as boxes, and attributes are shown within the box, rather than as separate ellipses in E-R diagrams.
- Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.
- The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.
- The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.
- Non-binary relationships drawn using diamonds, just as in ER diagrams

UML Class Diagrams



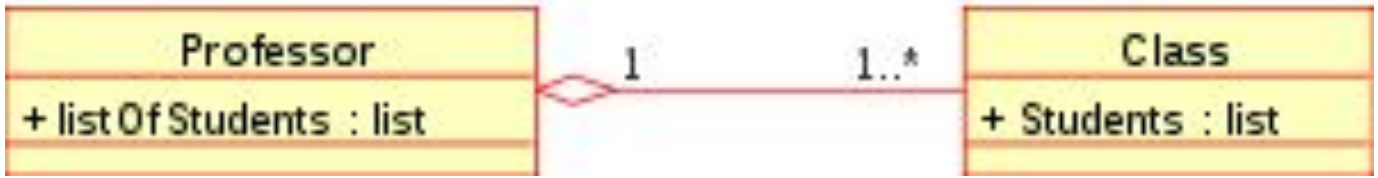
- In the diagram, classes are represented with boxes which contain three parts:
- The top part contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle part contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom part contains the methods the class can execute. They are also left-aligned and the first letter is lowercase.

UML Class Diagrams - Relationships



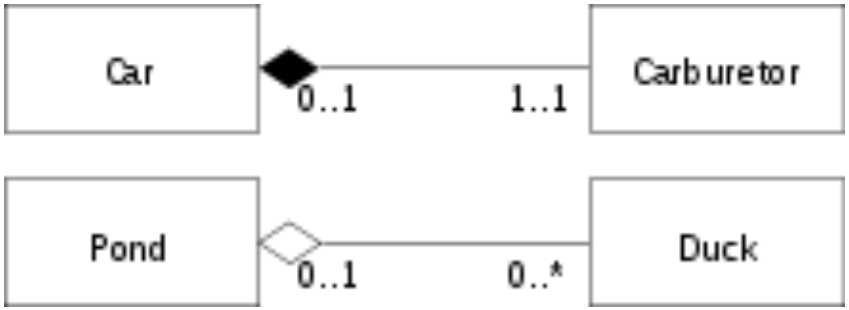
- Class diagram example of association between two classes

UML Class Diagrams - Aggregation



- Class diagram showing Aggregation between two classes
- Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship.

UML Class Diagrams - Composition



- Class diagram showing Composition between two classes at top and Aggregation between two classes at bottom
- Composition is a stronger variant of the "has a" association relationship; composition is more specific than aggregation.

UML Class Diagrams

- Cardinality constraints are specified in the form *l..h*, where *l* denotes the minimum and *h* the maximum number of relationships an entity can participate in.
- Beware: the positioning of the constraints is exactly the reverse of the positioning of constraints in E-R diagrams.
- The constraint 0..* on the *E*2 side and 0..1 on the *E*1 side means that each *E*2 entity can participate in at most one relationship, whereas each *E*1 entity can participate in many relationships; in other words, the relationship is many to one from *E*2 to *E*1.
- Single values, such as 1 or * may be written on edges; The single value 1 on an edge is treated as equivalent to 1..1, while * is equivalent to 0..*.

Reduction of an E-R Schema to Tables

- Primary keys allow entity sets and relationship sets to be expressed uniformly as *tables* which represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has a number of columns (generally corresponding to attributes), which have unique names.
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.

Composite and Multivalued Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - E.g. given entity set *customer* with composite attribute *name* with component attributes *first-name* and *last-name* the table corresponding to the entity set has two attributes

    name.first-name and *name.last-name*
- A multivalued attribute M of an entity E is represented by a separate table EM
  - Table EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M
  - E.g. Multivalued attribute *dependent-names* of *employee* is represented by a table *employee-dependent-names*(*employee-id, dname*)
  - Each value of the multivalued attribute maps to a separate row of the table EM
    - E.g., an employee entity with primary key John and dependents Johnson and Johndotir maps to two rows: (John, Johnson) and (John, Johndotir)

Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA hierarchies*, and *aggregation*.
- Note: There are many variations on ER model.

Summary of Conceptual Design

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.

Summary of Conceptual Design

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.