The Entity-Relationship Model

The Entity-Relationship Model

last updated 6-sep-19

Database Design

Goal of design is to generate a formal specification of the database ***conceptual*** schema.

Methodology:

1. Use E-R model to get a high-level graphical, conceptual view of the essential components of the enterprise and how these components are related
2. We later then convert the E-R diagram to SQL DDL, or whatever database model you are using

E-R Model is not SQL-based. It's not tied to any particular logical implementation of a DBMS. It is a conceptual and semantic model, which attempts to capture meanings rather than an actual implementation.

DO NOT THINK OR START WITH TABLES--YOU WILL BE MISGUIDED ON RELATIONSHIPS AND SOME ATTRIBUTES.

The E-R Model:

The enterprise is viewed as set of

- Entities
- Relationships among entities

Symbols used in E-R Diagram

- Entity – rectangle
- Attribute – oval
- Relationship – diamond
- Link - line

We model the potential relationship. Not all entities from a set necessarily connect/relate to another entity in another set

Entities and Attributes

Entity: an **object** that is involved in the enterprise and that be distinguished from other objects. (not shown in the ER diagram--is an instance)

- Can be person, place, event, object, concept in the real world
- Can be physical object or abstraction
- Ex: "John", "CSE305"

Entity Type or Set: set of similar objects or a category of entities; they are well defined (akin to an OO class).

- A rectangle represents an entity set
- Ex: ***students***, ***courses***
- We often just say "entity" and mean "entity type"

Attribute: describes one aspect of an entity type; usually [and best as] a single value and indivisible (atomic)

- Represented by oval on E-R diagram
- Ex: *name, maximum enrollment*
- May be **multi-valued** – use double oval on E-R diagram
- May be **composite** – attribute has further structure; also use oval for composite attribute, with ovals for components connected to it by lines
- May be **derived** – a virtual attribute, one that is computable from existing data in the database, use dashed oval. This helps reduce redundancy.
- One or composite set may be designated a **key** (later)

Entity Types

An entity type is

named

and is described by

set of attributes

- Student

  :

  Id, Name, Address, Hobbies

Domain: possible values of an attribute.

- Note that the value for an attribute can be a set or list of values, sometimes called "multi-valued" attributes
- This is in contrast to the pure relational model which requires atomic values
- E.g.,

  (111111, John, 123 Main St, (stamps, coins))

Key

: subset of attributes that uniquely identifies an entity (candidate key)

Entity Schema:

The meta-information of

entity type name

,

attributes (and associated domain)

,

key constraints

Entity Types tend to correspond to **nouns**; *attributes* are also nouns albeit descriptions of the parts of entities

We may have

null

values for some entity attribute instances – no mapping to domain for those instances

Keys

Superkey: an attribute or set of attributes that uniquely identifies an entity--there can be many of these

Composite key: a key requiring more than one attribute

Candidate key: a superkey such that no proper subset of its attributes is also a superkey (minimal superkey – has no unnecessary attributes)

Primary key

:

the candidate key chosen to be used for identifying entities and accessing records. Unless otherwise noted "key" means "primary key"

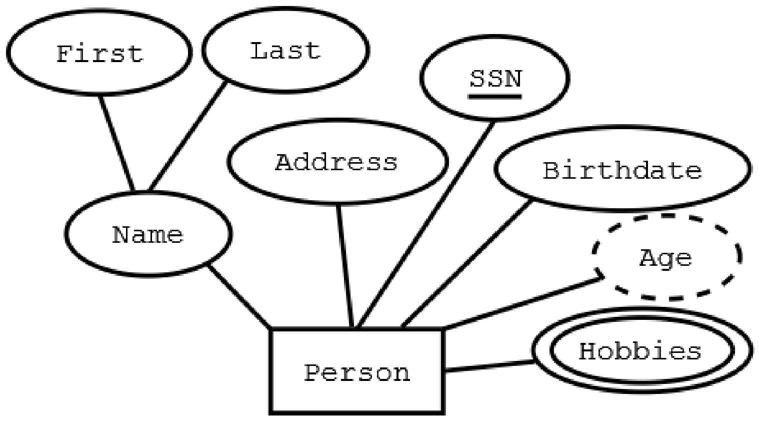
Alternate key: a candidate key not used for primary key

Secondary key: attribute or set of attributes commonly used for accessing records, but not necessarily unique

Foreign key: term used in relational databases **(but not in the E-R model)** for an attribute that is the primary key of another table and is used to establish a relationship with that table where it appears as an attribute also.

So a foreign key value occurs in the table and again in the other table. This conflicts with the idea that a **value** is stored only once; however, the idea that a **fact** is stored once is not undermined.

Graphical Representation in E-R diagram



Rectangle -- Entity

Ellipses -- Attribute (underlined attributes are [part of] the primary key)

Double ellipses -- multi-valued attribute

Dashed ellipses-- derived attribute, e.g. age is derivable from birthdate and current date.

[Drawing notes: *keep all attributes above the entity*. Lines have *no arrows*. Use straight lines only]

Relationships

Relationship: connects two or more entities into an association/relationship

- "John" majors in "Computer Science"

Relationship Type: set of similar relationships

- ***Student***

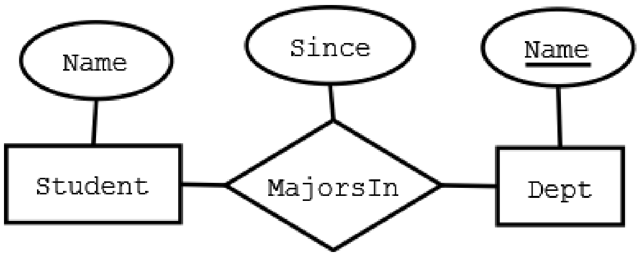
  (entity type) is related to

  Department

  (entity type) by

  MajorsIn

  (relationship type).



Relationship Types may also have attributes in the E-R model. When they are mapped to the relational model, the attributes become part of the relation. Represented by a diamond on E-R diagram.

Relationship types can have descriptive attributes like entity sets

Relationships tend to be **verbs or verb phrases;** attributes of relationships are again nouns, or prepositional phrases

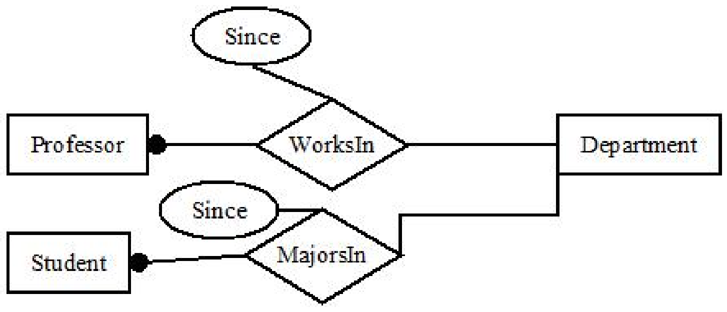
An **attribute** of a relationship type adds additional information to the relationship

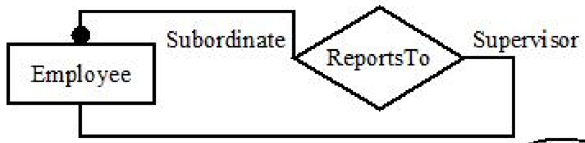
- e.g., "John" majors in "CS" ***since*** **2000**
- John and CS are related
- 2000 describes the relationship - it's the value of the ***since*** attribute of ***MajorsIn*** relationship type
- time stamps of updates or establishment of a relationship between two entities can be attributed here rather than with the entities.

[Drawing tips: relationship diamonds should connect off the left and right points; Dia can label those points with cardinality; use Manhattan connecting line (horizontal/vertical zigzag)]

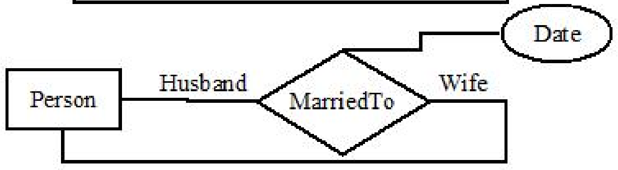
Roles

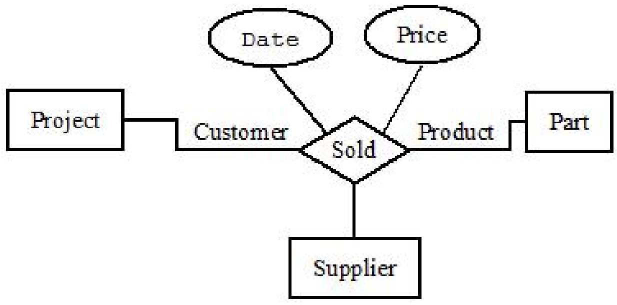
The **role** of a relationship type may additionally names the purpose of the entity in the relationship. Commonly the name of the entity serves asthe role name.





> e.g., "John" is value of ***Student*** role, "CS" value of ***Department*** role of ***MajorsIn*** relationship type





> The tuple (John, CS, 2000) describes a relationship

Interesting situation: relationships can relate elements of same entity type

> e.g., ReportsTo relationship type relates two elements of Employee entity type:
>
> - Bob reports to Mary since 2000
>
> We do not have distinct names for the roles. It is not clear who reports to whom.

Solution: the role name of relationship type need not be same as name of entity type from which participants are drawn

- *ReportsTo* has roles *Subordinate* and *Supervisor* and attribute *Since*
- Values of *Subordinate* and *Supervisor* both drawn from entity type *Employee*

It is optional to name role of each entity-relationship, but helpful in cases of

- Recursive relationship – entity set relates to itself
- Multiple relationships between same entity sets

Roles are edges labeled with role names (omitted if role name = name of entity set). Most attributes have been omitted.

Relationship Type

Relationship types are described by the set of roles (entities) and [optional] attributes

- e.g., **MajorsIn**: Student, Department, Since

Think that entities are nouns; relationship types are often verbs

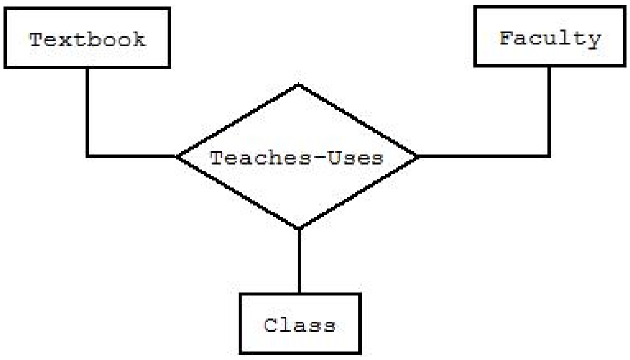
- students and departments are the entities (nouns) and roles in relationship types
- majors is the relationship type (verb)
- i.e., "student" "majors in " "department"

Here we have equate the role name (Student) the name of the entity type (Student) of the participant in the relationship.

Degree of relationship

The **number** of roles (entity participants) in the relationship

Binary – links two entity sets; set of ordered pairs (most common)

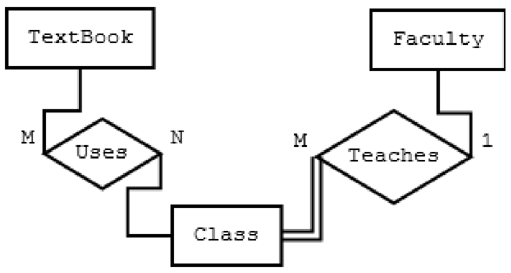


Ternary – links three entity sets; ordered triples (rare). If a relationship exists among the three entities, all three must be present. This is rare.

NOTE: Ternary relationships are rare.

N-ary – links n entity sets; ordered n-tuples (extremely rare). If a relationship exists among the entities, then all must be present. Cannot represesnt subsets.

Note: ternary relationships may sometimes be replaced by two or more binary relationships (see book Figures 3.5 and 3.13). Semantic equivalence between ternary relationships and two binary ones are not necessarily the same.



Why are non binary relationship rare?

> If you have a ternary relationship, there must be 3 entities that relate simultaneously--a triple, not just a pair.
>
> A four-way relationship would require a quadruple--all four, together represent one relationship.

Cardinality of Relationships

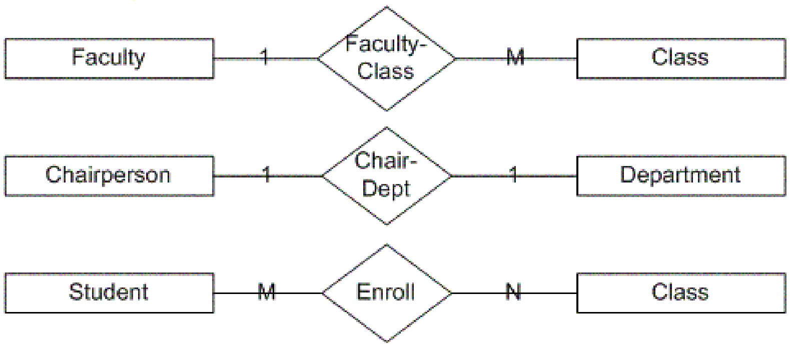
Cardinality is the number of entity instances to which another entity set can map under the relationship. This does not reflect a requirement that an entity has to participate in a relationship. Participation is another concept.

One-to-one: X-Y is 1:1 when each entity in X is associated with at most one entity in Y, and each entity in Y is associated with at most one entity in X.

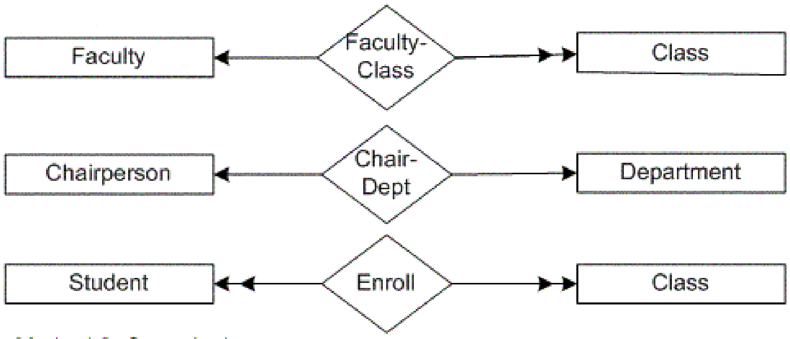
One-to-many: X-Y is 1:M when each entity in X can be associated with many entities in Y, but each entity in Y is associated with at most one entity in X.

Many-to-many: X:Y is M:M if each entity in X can be associated with many entities in Y, and each entity in Y is associated with many entities in X ("many" =>one or more and sometimes zero)

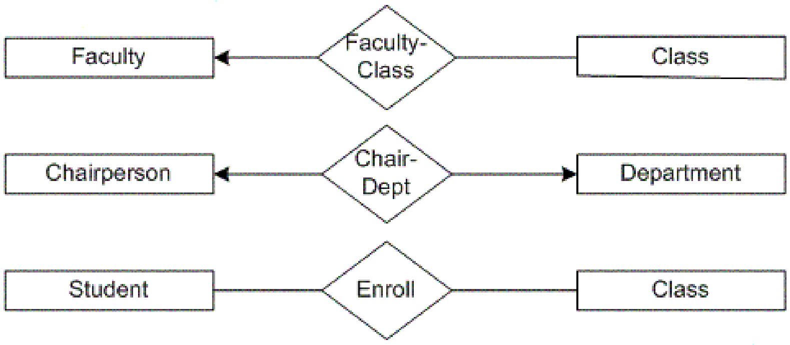
From all of these choices, please use the first method!



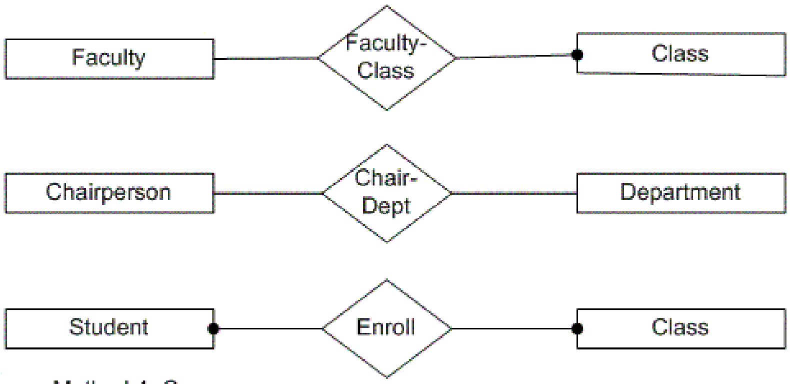
Method 1: One: 1
Many M,N



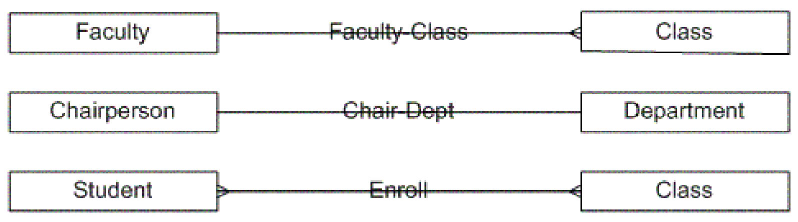
Method 2: One: single arrow
Many: double arrow



Method 3: One: single arrow
Many: no arrow

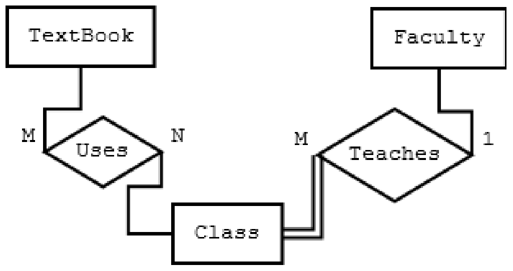


Method 4: One: no arrow
Many: big dot



Method 5: One: no arrow
Many: crow's feet

Relationship Participation Constraints



Total participation

- Every member of the entity set must participate in the relationship
- Represented by ***double line*** from entity rectangle to relationship diamond
- E.g., A ***Class*** entity cannot exist unless related to a ***Faculty*** member entity in this example, not necessarily at Juniata.
- You can set this double line in Dia
- In a relational model we will use the ***references*** clause.

Key constraint

- If every entity participates in *exactly* one relationship, both a total participation and a key constraint hold
- E.g., a class is taught by only one faculty member.

Partial participation

- Not every entity instance must participate
- Represented by single line from entity rectangle to relationship diamond
- E.g., A ***Textbook*** entity can exist without being related to a C***lass*** or vice versa.

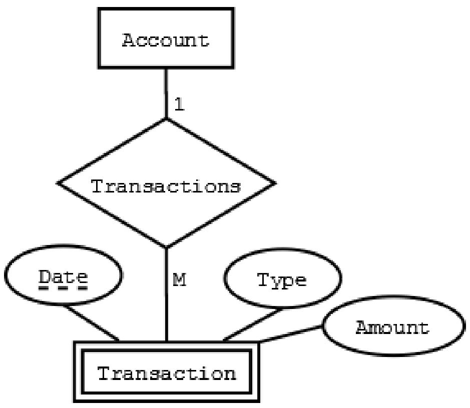
Existence Dependency and Weak Entities

Existence dependency: Entity Y is existence dependent on entity X is each instance of Y must have a corresponding instance of X

In that case, Y must have total participation in its relationship with X

If Y does not have its own candidate key, Y is called a **weak entity**, and X is **strong entity**

Weak entity may have a partial key, called a discriminator, that distinguishes instances of the weak entity that are related to the same strong entity



Use double rectangle for weak entity, with double diamond for relationship connecting it to its associated strong entity

Note: not all existence dependent entities are weak – the lack of a key is essential to definition

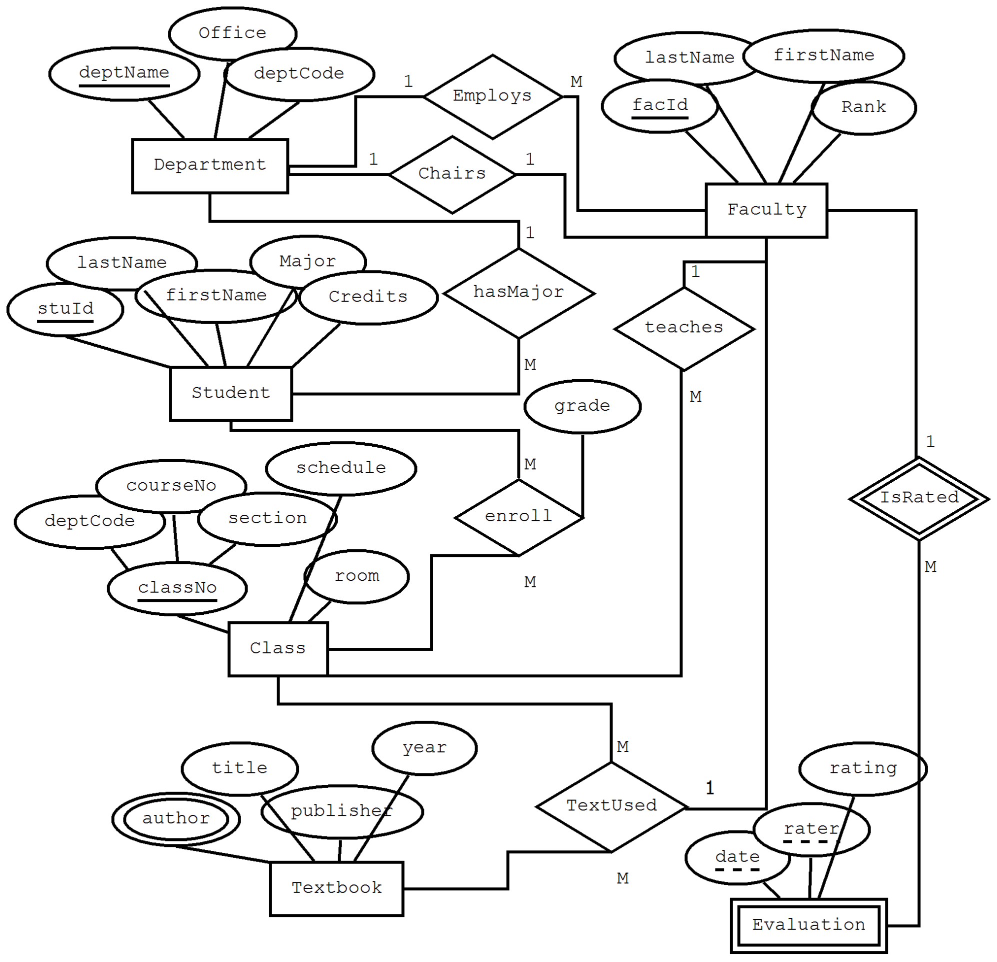
Schema of a Relationship Type

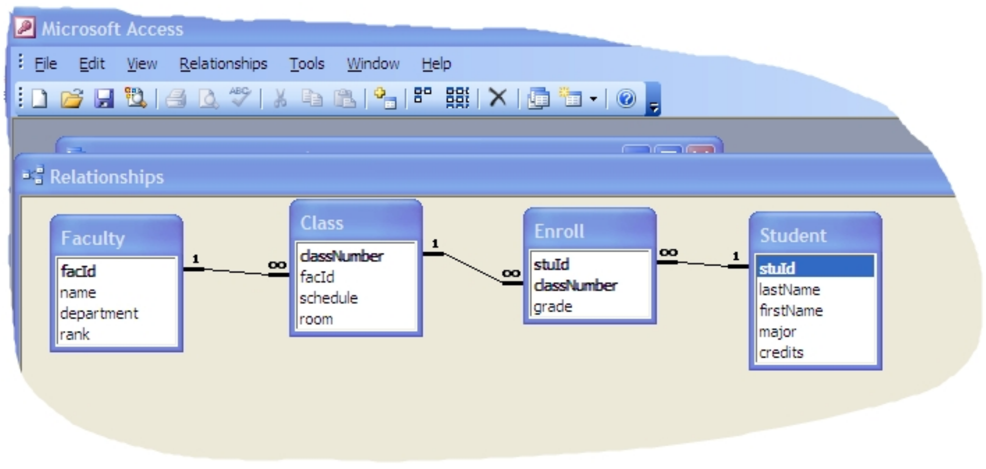
Contains the following features:

> ***Role names***, R_i, and their corresponding entity sets. Roles must be single valued (the number of roles is called its **degree**)
>
> ***Attribute names***, A_j, and their corresponding domains. Attributes in the E-R model may be set or multi-valued.
>
> ***Key***: Minimum set of roles and attributes that uniquely identify a relationship
>
> ***Relationship***: $<e_1, ...e_n; a_1, ...a_k>$
>
> - e_i is an entity, a value from R_i's entity set
> - a_j is a set of attribute values with elements from domain of A_j

ER Diagram Example

This was produced with Dia. It is the same as the figure in the book using instructor's preferred style.





Faculty: facId, name, department, rank

Class: classNumber, facId, schedule, room

Enroll: stuId, classNumber, grade

Student: stuId, lastName, firstName, major, credits