Database Design Entity-Relationship (ER) Modeling: Relationships

COSC 304 – Introduction to Database Systems

ER Modeling
Relationship Cardinalities

Relationship cardinalities or ***multiplicities*** are used to restrict how entity types participate in relationships in order to model real-world constraints.

The ***multiplicity*** is the number of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship.

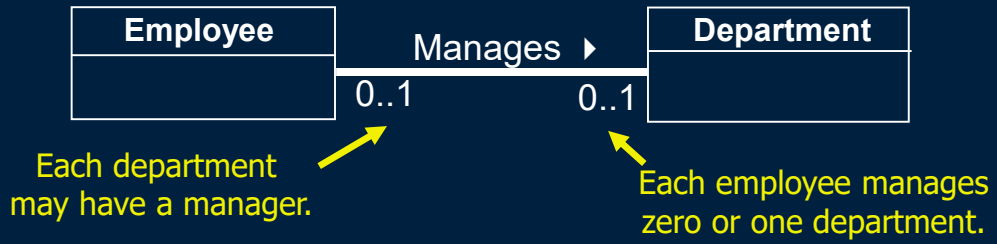
For binary relationships, there are three common types:

- one-to-one (1:1)
- one-to-many (1:* or 1:N)
- many-to-many (*:* or N:M)

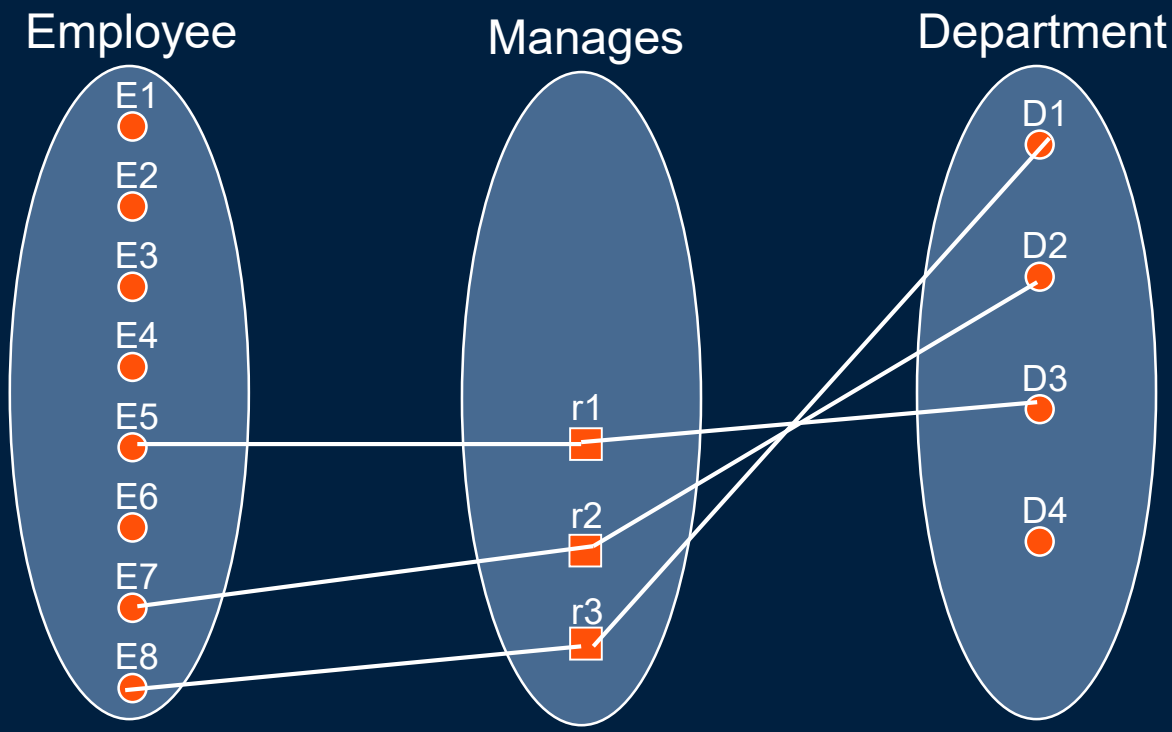
One-to-One Relationships

In an one-to-one relationship, each instance of an entity class E1 can be associated with **at most one** instance of an entity class E2 and vice versa.

Example: A department may have only one manager, and a manager may manage only one department.



One-to-One Relationship Example



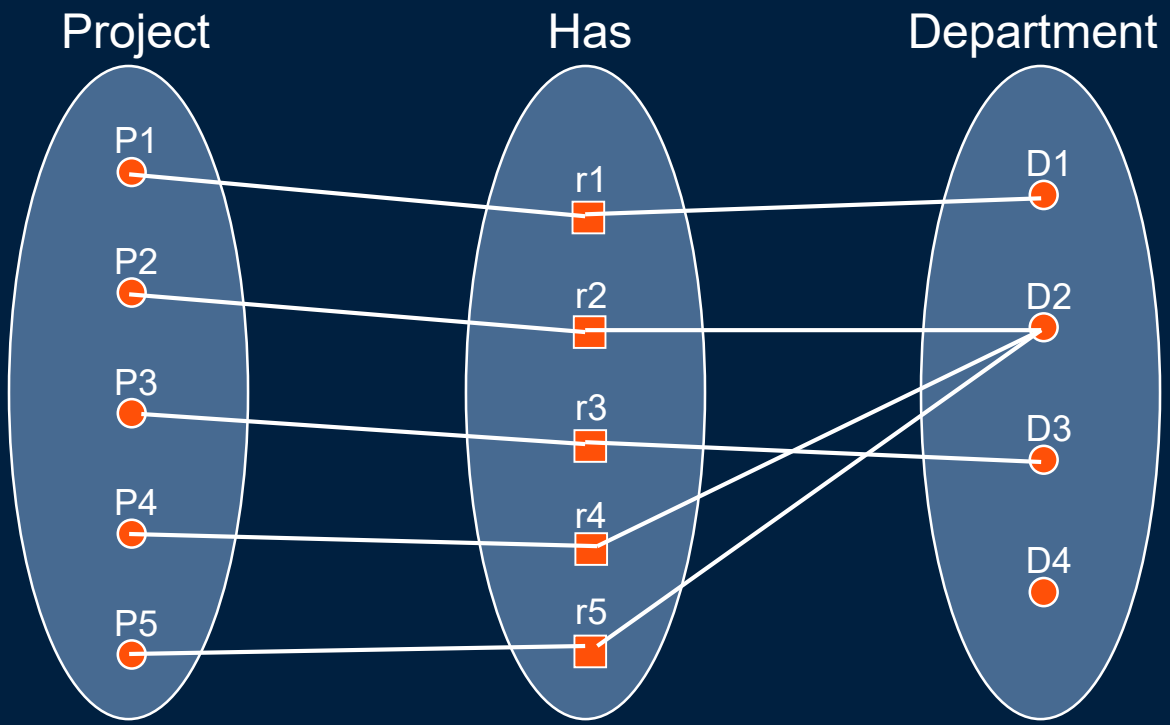
Relationship explanation: A department may have only one manager. A manager (employee) may manage only one department.

One-to-Many Relationships

In a one-to-many relationship, each instance of an entity class E1 can be associated with **more than one** instance of an entity class E2. However, E2 can only be associated with **at most one** instance of entity class E1.

Example: A department may have multiple projects, but a project may have only one department.

One-to-Many Relationship Example

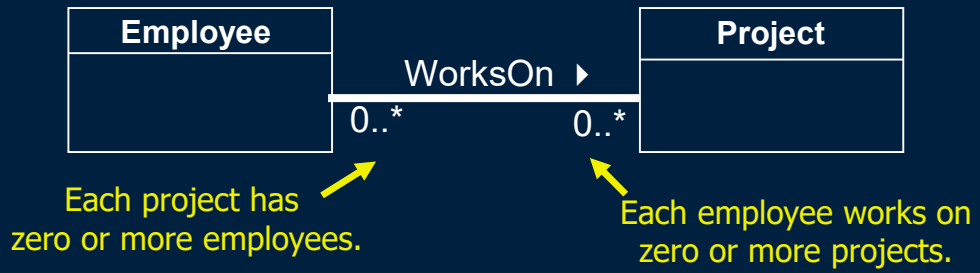


Relationship: One-to-many relationship between department and project.

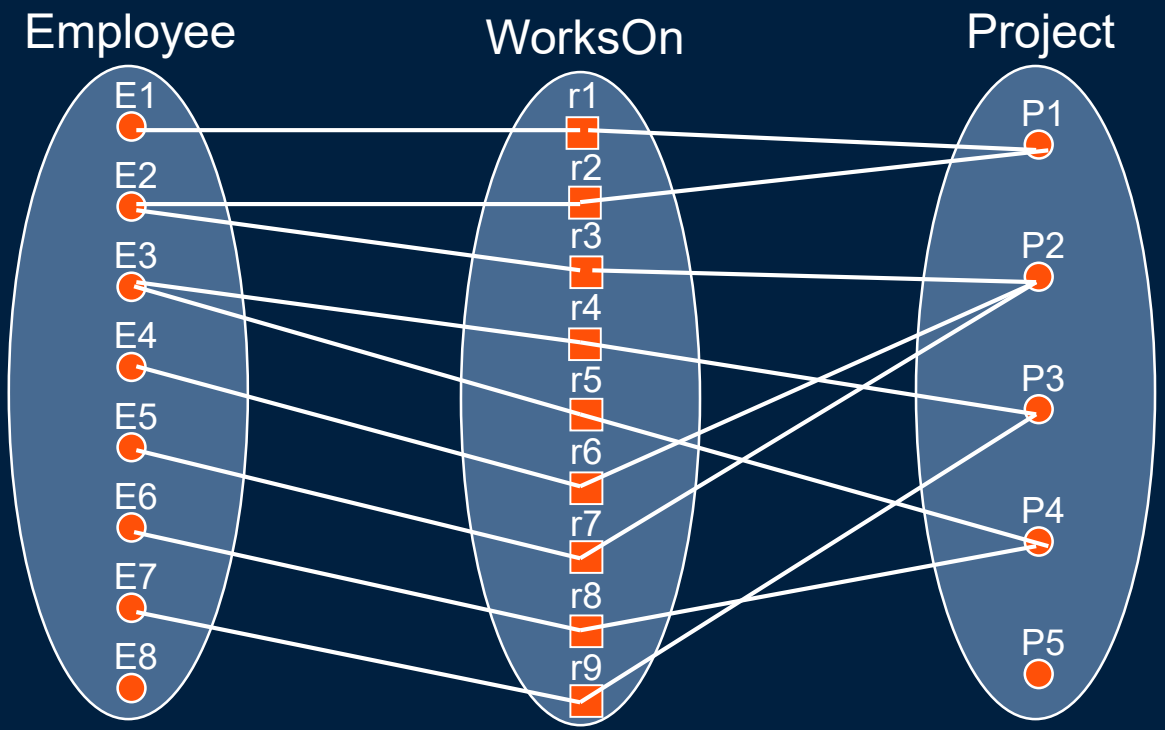
Many-to-Many Relationships

In a many-to-many relationship, each instance of an entity class E1 can be associated with **more than one** instance of an entity class E2 and vice versa.

Example: An employee may work on multiple projects, and a project may have multiple employees working on it.



Many-to-Many Relationship Example



Participation Constraints

Cardinality is the *maximum* number of relationship instances for an entity participating in a relationship type.

Participation is the *minimum* number of relationship instances for an entity participating in a relationship type.

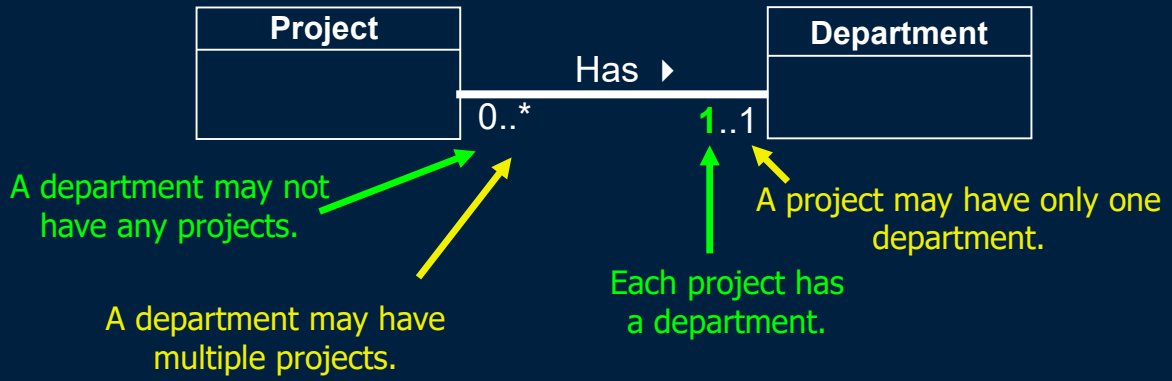
- Participation can be *optional* (zero) or *mandatory (1 or more)*.

If an entity's participation in a relationship is mandatory (also called *total* participation), then the entity's existence depends on the relationship.

- Called an *existence dependency*.

Participation Constraints Example

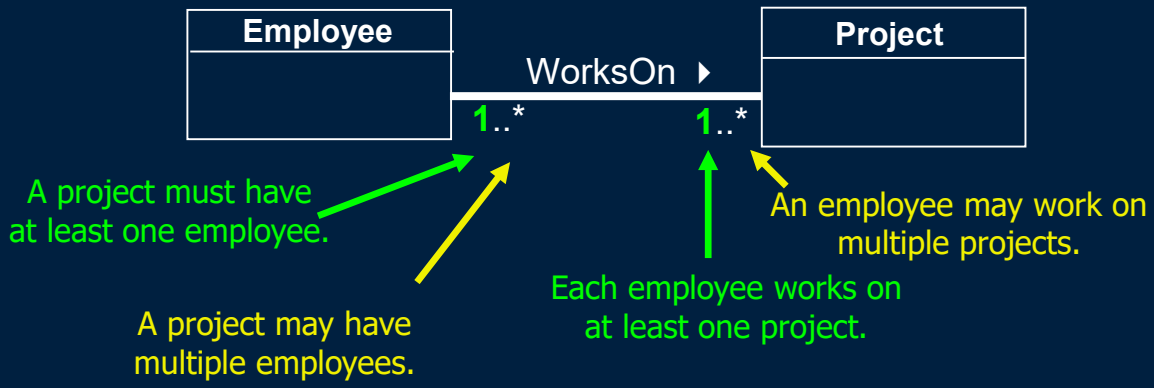
Example: A project is associated with one department, and a department may have zero or more projects.



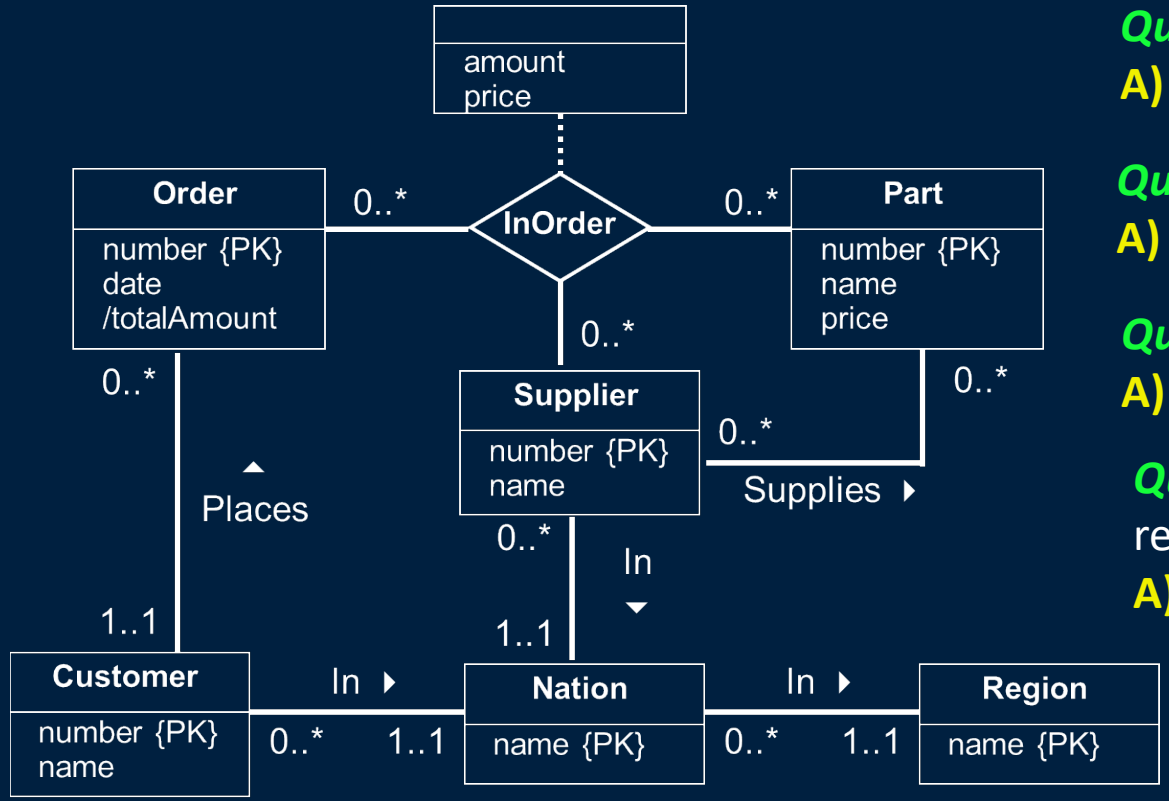
Note: Every project must participate in the relationship (mandatory).

Participation Constraints Example 2

Example: A project must have one or more employees, and an employee must work on one or more projects.



TPC-H ER Diagram Questions



Question 1: How many entity types?
A) 5 **B)** 6 **C)** 7 **D)** 8 **E)** 9

Question 2: How many relationships?
A) 4 **B)** 5 **C)** 6 **D)** 7 **E)** 8

Question 3: How many primary keys?
A) 4 **B)** 5 **C)** 6 **D)** 7 **E)** 8

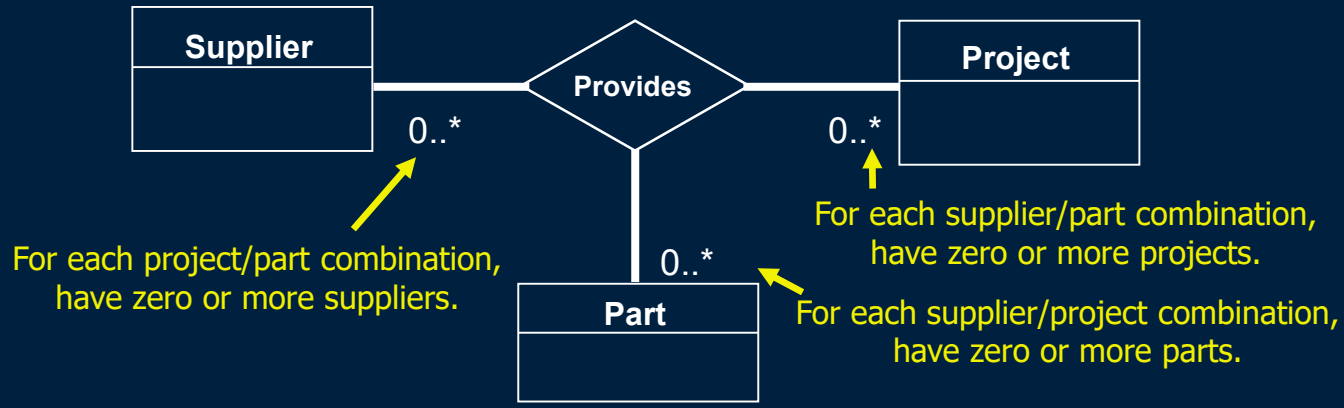
Question 4: How many 1-N relationships?
A) 4 **B)** 5 **C)** 6 **D)** 7 **E)** 8

Question 5: How many foreign keys? **A)** 3 **B)** 4 **C)** 5 **D)** 9 **E)** 0

Multiplicity of Non-Binary Relationships

The multiplicity in a complex relationship of an entity type is the number of possible occurrences of that entity-type in the *n*-ary relationship when the other (*n*-1) values are fixed.

Example: A supplier may provide zero or more parts to a project. A project may have zero or more suppliers, and each supplier may provide to the project zero or more parts.



Relationship Cardinality Question

Question: How many of the following statements are ***true***?

1) An entity of type A must be related to an entity of type B.

2) An entity of type A may be related to more than one entity B.

3) This is a 1-to-many relationship between A and B.

4) An entity of type B must be related to an entity of type A.

5) An entity of type B must be related to more than one entity of type A.

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

Multiplicity Practice Question

Consider the university database developed before. Write multiplicities into the ER diagram given that:

- A department must offer at least 2 courses and no more than 20 courses. Courses are offered by only one department.
- A course may have multiple sections, but always has at least one section.
- A student may enroll for courses (but does not have to).
- A professor may be in multiple departments (at least 1), and a department must have at least 3 professors.
- A section is taught by at least one professor, but may be taught by more than one. A professor does not have to teach.

Strong and Weak Entity Types

A ***strong entity type*** is an entity type whose existence is not dependent on another entity type.

- A strong entity type always has a primary key of its own attributes that uniquely identifies its instances.

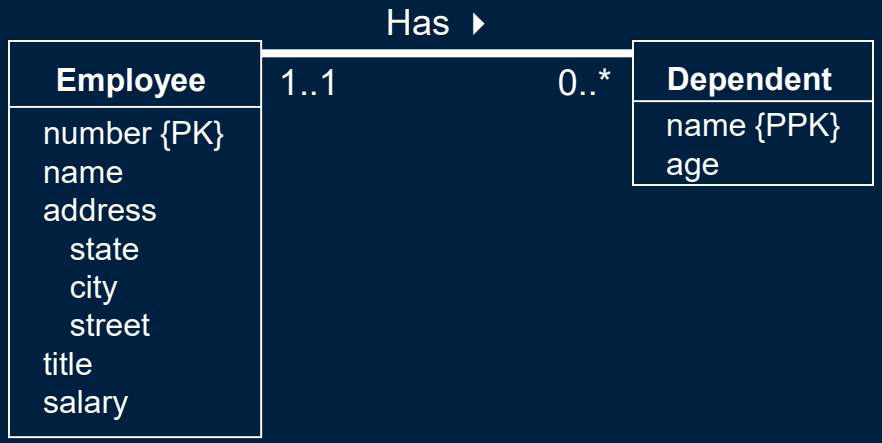
A ***weak entity type*** is an entity type whose existence is dependent on another entity type.

- A weak entity type does not have a set of its own attributes that uniquely identifies its instances.

A common example of strong and weak entity types are employees and their dependents:

- An employee is a strong entity because it has an employee number to identify its instances.
- A dependent (child) is a weak entity because the database does not store a key for each child, but rather they are identified by the parent's employee number and their name.

Weak Entities in UML



Strong and Weak Entity Question

Question: How many of the following statements are ***true***?

1) A weak entity has its own primary key.

2) A strong entity has its own primary key.

3) A weak entity must be associated (identified) by a strong entity.

4) A weak entity can have a relationship with another entity besides its identifying strong entity.

5) The attribute(s) of a weak entity used to identify it with its associated strong entity are noted as `{PPK}` in the UML model.

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

ER Modeling – Entity, Relationship, or Attribute?

Basic challenge is when to model a concept as an entity, a relationship, or an attribute. In general:

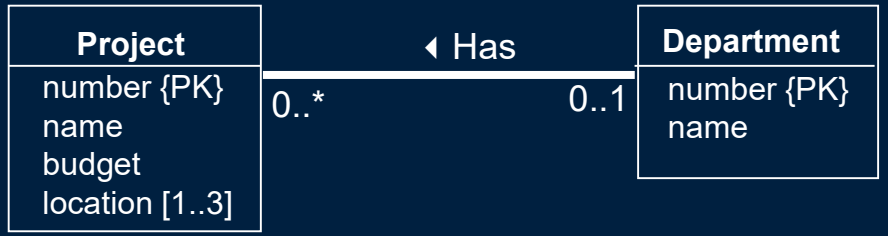
- Entities are nouns.
  - You should be able to identify a set of key attributes for an entity.
- Attributes are properties and may be nouns or adjectives.
  - Use an attribute if it relates to one entity and does not have its own key.
  - Use an entity if the concept may be shared by entities and has a key.
- Relationships should generally be binary.
  - Note that non-binary relationships can be modeled as an entity instead.

Good design will avoid redundancy and limit use of weak entities.

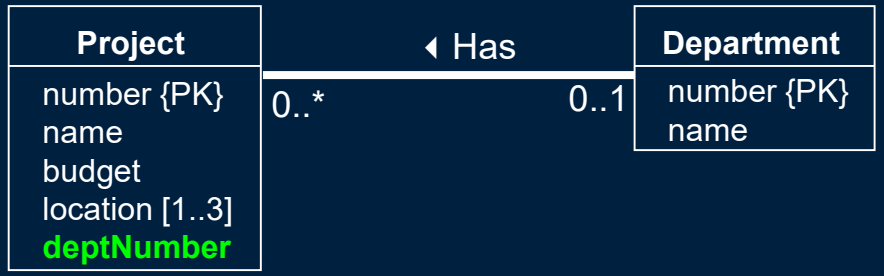
- Human-made (e.g. SIN) or auto-generated keys used instead of weak entities.

Good Design Practices
Avoiding Redundancy Example

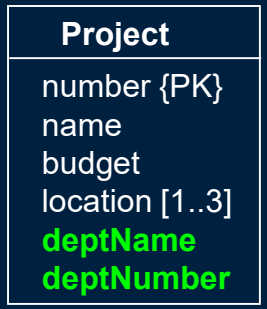
Good:



Wrong:



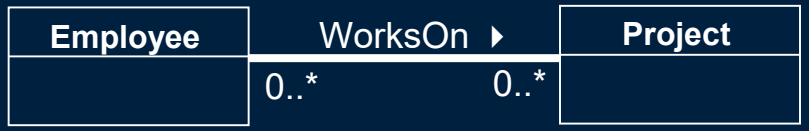
Bad:



Many-to-Many Relationship Simplification

A many-to-many relationship can be converted into one entity with two 1:N relationships between the new entity and the original entities participating in the relationship.

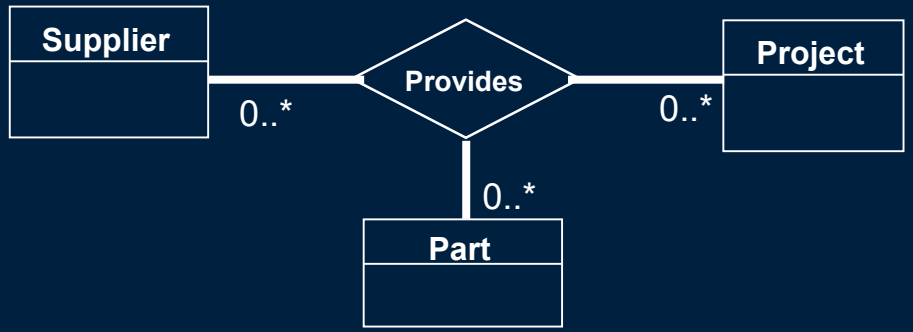
Original:



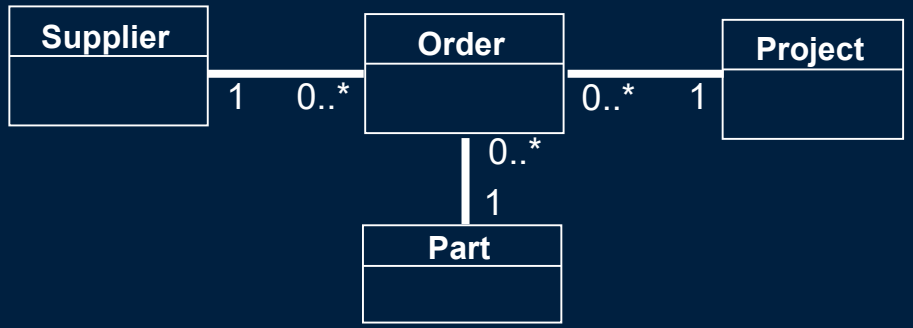
Simplified:

Higher Degree Relationships Simplified using a Weak Entity

Original:

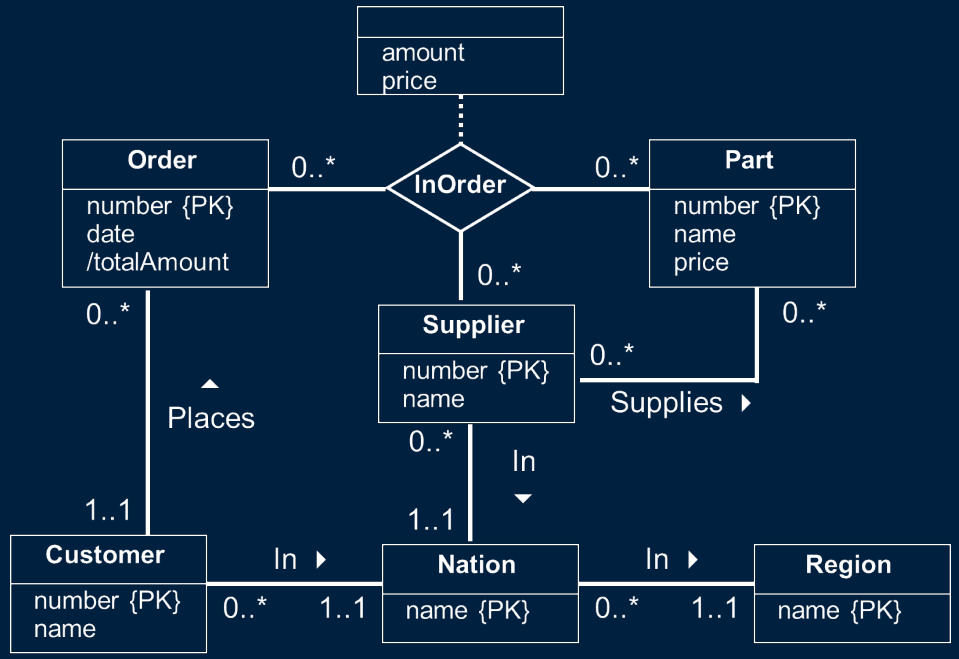


With weak entity:



ER Design Example
TPC-H Standard Schema

- Each order has a numeric key, a customer, a date, a total order amount, and a list of parts.
- A part in an order has an amount and a price paid and is supplied by a certain supplier.
- Each part has a numeric key, a name, and a price and may be supplied by multiple suppliers.
- A supplier has a key and a name and may supply multiple parts.
- A customer has a key and a name.
- Each supplier and customer is located in a nation.
- Each nation is located in a region (continent).



ER Design Question #2

Construct a fish store database where:

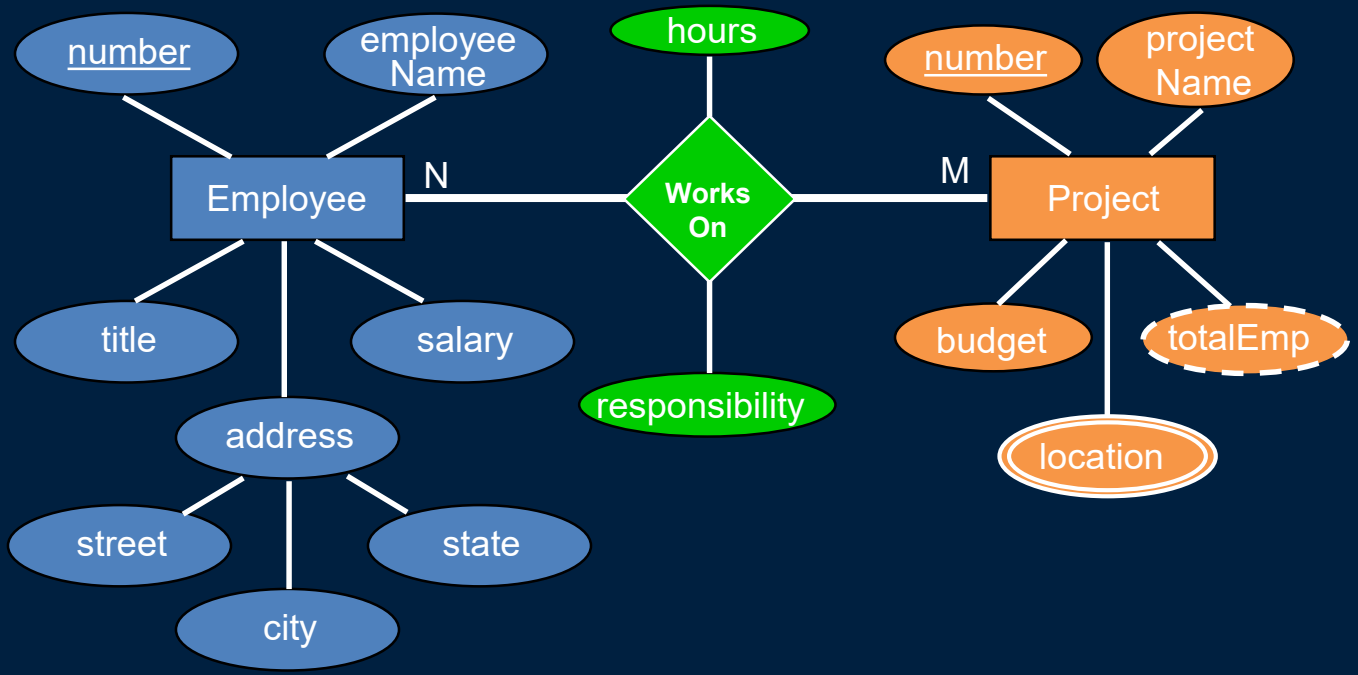
- A fish store maintains a number of aquaria (tanks), each with a number, name, volume and color.
- Each tank contains a number of fish, each with an id, name, color, and weight.
- Each fish is of a particular species, which has a id, name, and preferred food.
- Each individual fish has a number of events in its life, involving a date and a note relating to the event.

ER Design Question #3

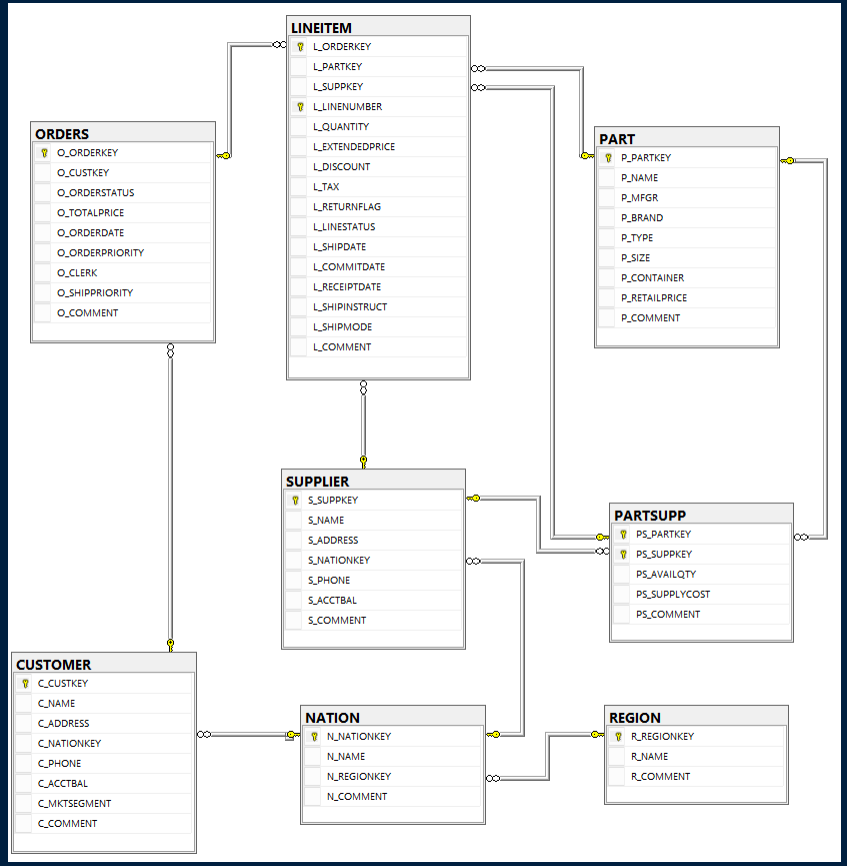
Construct an invoice database where:

- An invoice is written by a sales representative for a single customer and has a unique ID. An invoice has a date and total amount and is comprised of multiple detail lines, containing a product, price and quantity.
- Each sales representative has a name and can write many invoices, but any invoice is written by a single representative.
- Each customer has a unique id, name, and address and can request many invoices.
- Products have descriptions and weights and are supplied by vendors. Each product has a unique name for a particular vendor. A product is supplied by only one vendor.
- A vendor has an id and an address.

ER Model - Historical Notation



Other Notations – Logical/Relational Diagram



Conclusion

Conceptual design is performed at a high-level of abstraction involving entities, relationships, and attributes.

- An entity type is a group of entities with the same properties.
  - Entities may be strong (have unique key) or weak (no unique key).
- A relationship type is an association between entities.
  - A relationship may involve two or more entities and may be recursive.
  - A relationship has two types of constraints:
    - Participation - minimum # of times an entity must be involved in relationship
    - Cardinality - maximum # of times an entity can be involved in relationship
  - Common relationship multiplicities are: 1:1, 1:*, *:*.
- Attributes are properties of entities or relationships.

Good design requires practice.

Objectives

- Define and identify on an ER diagram: entity type, relationship type, degree of a relationship, recursive relationship, attribute, multi-valued attribute, derived attribute
- Define and identify on an ER diagram: primary key, partial primary key
- Define and identify on an ER diagram: cardinality and participation constraints
- Explain the difference between a strong entity type and a weak entity type.
- Explain multiplicity and participation and how they are used in modeling.

Model a domain explained in an English paragraph in an ER diagram using UML notation.


UBC
THE UNIVERSITY OF BRITISH COLUMBIA