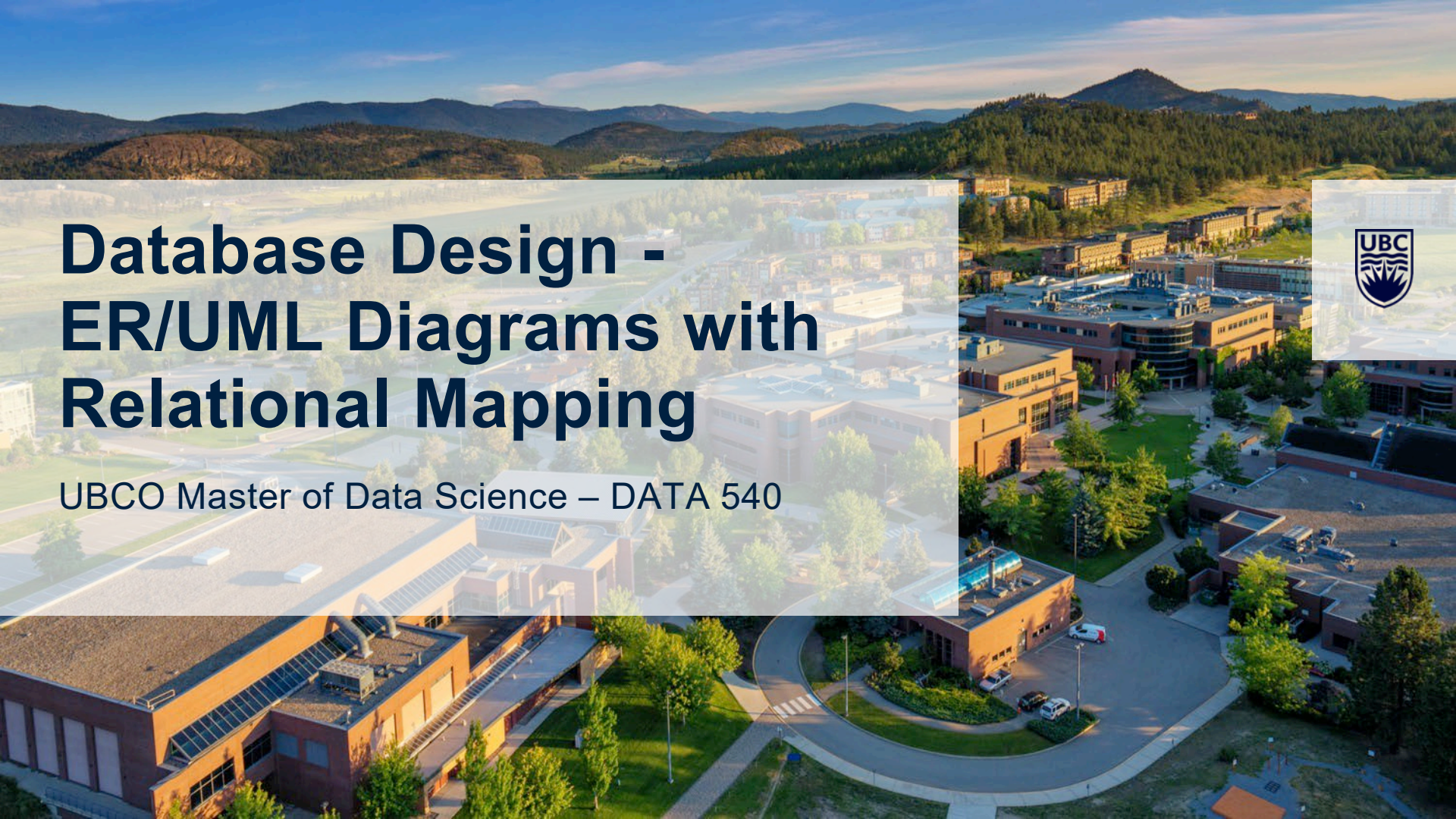


Database Design - ER/UML Diagrams with Relational Mapping

UBCO Master of Data Science – DATA 540

Objectives

- Describe the three steps in database design including the results of each step.
- Describe differences between conceptual, logical, and physical data models.
- Identify on an ER diagram: entity type, relationship type, degree of a relationship, recursive relationship, attribute, multi-valued attribute, derived attribute
- Identify on an ER diagram: primary key, partial primary key
- Identify on an ER diagram: cardinality and participation constraints
- Given an ER diagram, translate to the relational model with sufficient detail to be able to write queries using only an ER diagram in UML syntax.

Database Design

The ability to design databases and associated applications is critical to the success of the modern enterprise.

Database design requires understanding both the operational and business requirements of an organization as well as the ability to model and realize those requirements in a database.

Developing database and information systems is performed using a ***development lifecycle***, which consists of a series of steps. Data analysts may have access to design documents to help understand how to use the data properly.

The Importance of Database Design

Some statistics on software projects:

- 80 - 90% do not meet their performance goals
- 80% delivered late and over budget
- 40% fail or abandoned
- 10 - 20% meet all their criteria for success

The primary reasons for failure are improper requirements specifications, development methodologies, and design techniques.

Database Design

Database design is divided into three phases:

- ***Conceptual database design*** - models the collected information at a high-level of abstraction without using a particular data model or DBMS.
  - *Top-down design* by specifying entities, attributes, and relationships.
- ***Logical database design*** - constructs a model of the information in the domain using a particular data model, but independent of the DBMS.
  - Typically use relational model but may also use object-oriented, graphs, JSON, or XML.
  - Since logical design selects a data model, it is now possible to model the information using the features of that model (e.g. keys and foreign keys in relational model).
- ***Physical database design*** - constructs a physical model of information in a given data model for a particular DBMS. Selects a database system and determines how to represent the logical model on that DBMS.
  - E.g. creating tables, indexes, security, data partitioning
  - Physical database design is ***how***, and logical database design is the ***what***.

Entity-Relationship Modeling

Entity-relationship modeling is a top-down approach to conceptual database design that models the data as entities, attributes, and relationships.

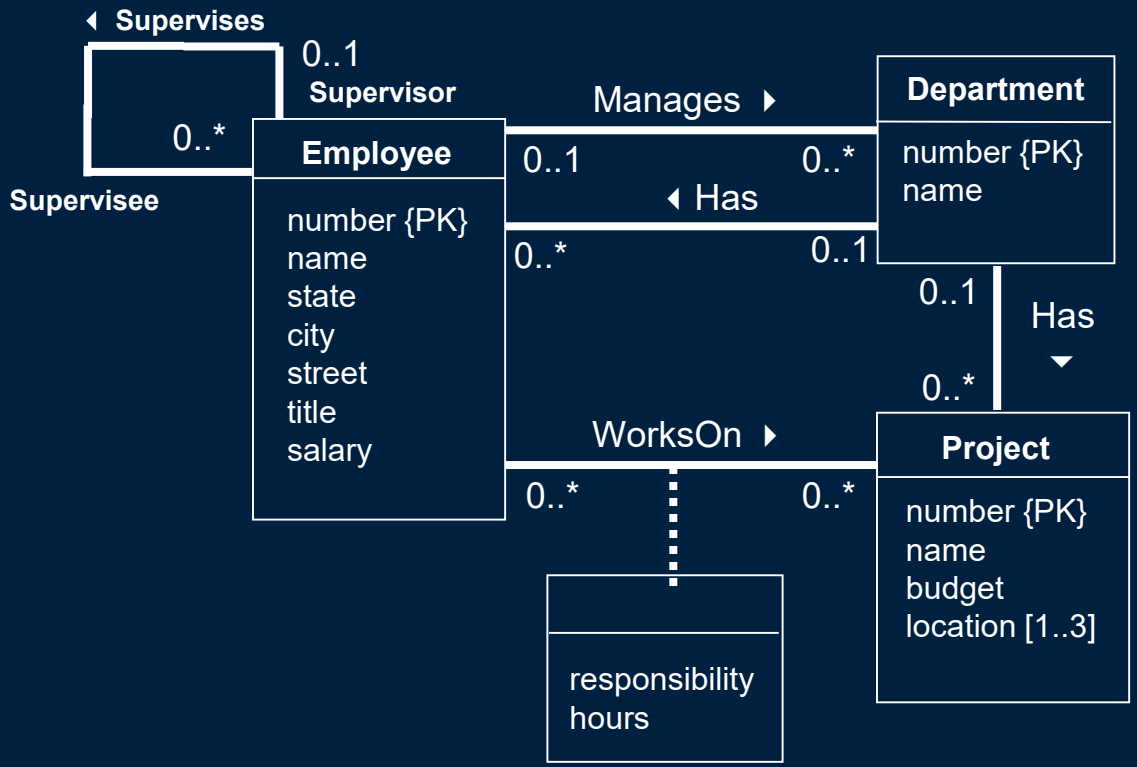
- The entity-relationship (ER) model was proposed by Peter Chen in 1976. We will perform ER modeling using Unified Modeling Language (UML) syntax.

The ER model refines entities and relationships by including properties of entities and relationships called *attributes*, and by defining *constraints* on entities, relationships, and attributes.

The ER model conveys knowledge at a high-level (conceptual level) which is suitable for interaction with technical and non-technical users.

Since the ER model is data model independent, it can later be converted into the desired logical model (e.g. relational model).

ER Model Example in UML notation



Entity Types

An ***entity type*** is a group of objects with the same properties which are identified as having an independent existence.

- An entity type does not always have to be a physical real-world object such as a person or department. It can be an abstract concept such as a project or job.

An ***entity instance*** is a particular example or occurrence of an entity type.

- For example, an entity type is Employee. A entity instance is 'E1 - John Doe'.

Representing Entity Types

Entity types are represented by rectangles with the name of the entity type in the rectangle.

Examples:

- An entity type name is normally a singular noun.
  - That is, use Person instead of People, Project instead of Projects, etc.
- The first letter of each word in the entity name is capitalized by convention.

Entities Question

Question: Which one of the following statements is ***true***?

A) Entity types are represented using a rectangle box.

B) An entity type is always a physical object.

C) An entity type is named using a plural noun.

D) Employee number is an entity type.

Relationships

A ***relationship type*** is a set of associations among entity types. Each relationship type has a name that describes it.

A ***relationship instance*** is a particular occurrence of a relationship type that relates entity instances.

There can be more than one relationship between two entity types.

Relationship Degree

The ***degree of a relationship type*** is the number of entity types participating in the relationship.

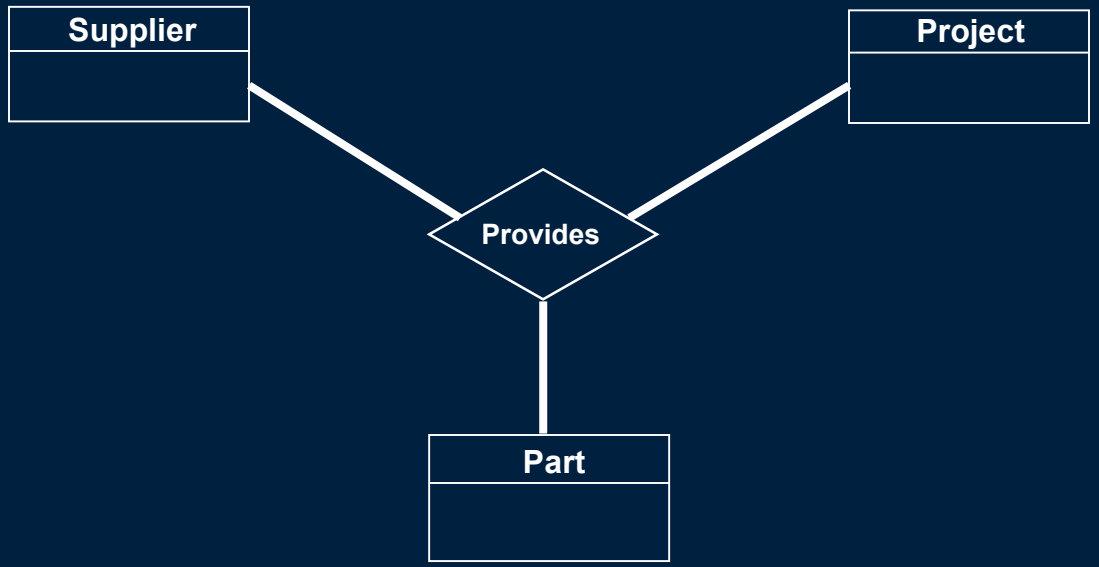
- For example, WorksOn is a relationship type of degree two as the two participating entity types are Employee and Project.
  - Note: This is not the same as degree of a relation which was the number of attributes in a relation.

Relationships of degree two are *binary*, of degree three are *ternary*, and of degree four are *quaternary*.

- Relationships of arbitrary degree N are called *n*-ary.

Use a diamond to represent relationships of degree higher than two.

Ternary Relationship Type Example



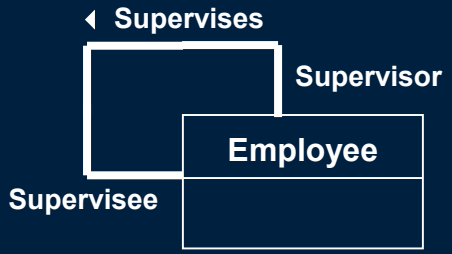
A project may require a part from multiple different suppliers.

Recursive Relationships

A ***recursive relationship*** is a relationship type where the same entity type participates more than once in different roles.

- For example, an employee has a supervisor. The supervisor is also an employee. Each *role* has a *role name*.

Example:



- The degree of a recursive relationship is two as the same entity type participates twice in the relationship.
  - It is possible for an entity type to be in a relationship more than twice.

Relationship Question

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

1) Relationships are represented using a directed edge (with arrows).

2) A relationship is typically named using a verb.

3) It is not possible to have a relationship of degree 1.

4) The degree of a relationship is the number of attributes it has.

5) A diamond is used to represent a relationship of degree larger than two.

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

Attributes

An ***attribute*** is a property of an entity type or a relationship type.

- For example, entity type Employee has attributes name, salary, title, etc.

Some rules:

- By convention, attribute names begin with a lower case letter.
- Attribute names are typically nouns.
- Each attribute has a *domain* which is the set of allowable values for the attribute (data type).
- An attribute may be single valued or have multi-values.

Attribute Question

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

1) Attributes are properties of either entities or relationships.

2) An attribute may be multi-valued.

3) Each attribute has a domain representing its data type.

4) Attribute names are typically verbs.

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

Representing Attributes and Keys

A ***primary key*** is a minimal set of attributes that identifies each instance of an entity type. Attributes labeled with `{PK}` in diagram.

- Note: No foreign keys in ER model but may see `{FK}` notation in logical diagram.

Other attribute notations:

- multi-valued attribute: *attributeName* [*minVals*..*maxVals*]
  - e.g. phoneNumber [1..3]
- derived attribute: */attributeName* (e.g. /totalEmp)
  - Derived attribute is not stored in database (calculated on demand)
- partial primary key: {PPK} – for key field of weak entity
  - A ***weak entity type*** is an entity type whose existence depends on another entity type.

Attributes on Relationships

An attribute may be associated with a relationship type.

For example, the WorksOn relationship type has two attributes: responsibility and hours.

Note that these two attributes belong to the relationship and cannot belong to either of the two entities individually (as they would not exist without the relationship).

Relationship attributes are represented as a separate box connected to the relationship using a dotted line.

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.

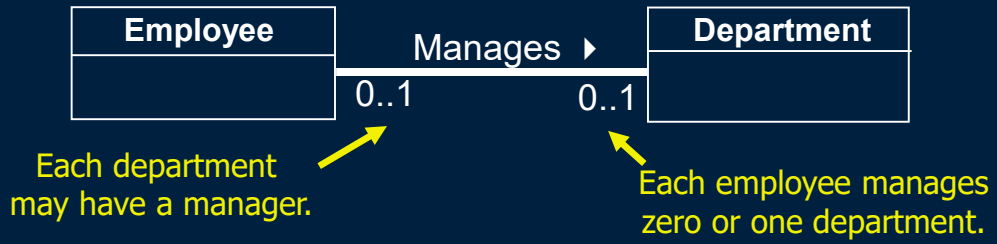
For binary relationships, there are three common types:

- 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-Many Relationships

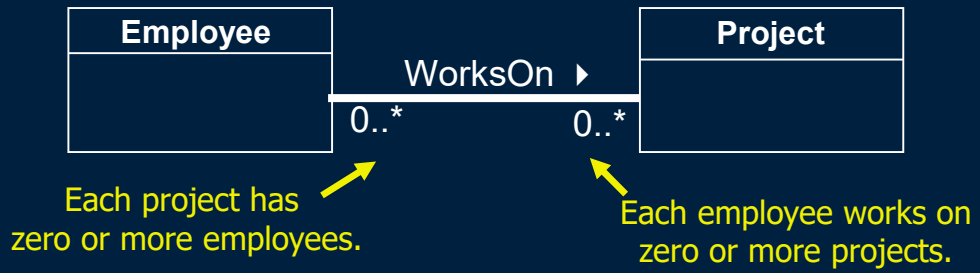
In a one-to-many relationship, each instance of an entity class E1 can be associated with **more than one** instance of an entity class E2. However, E2 can only be associated with **at most one** instance of entity class E1.

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

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.



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.

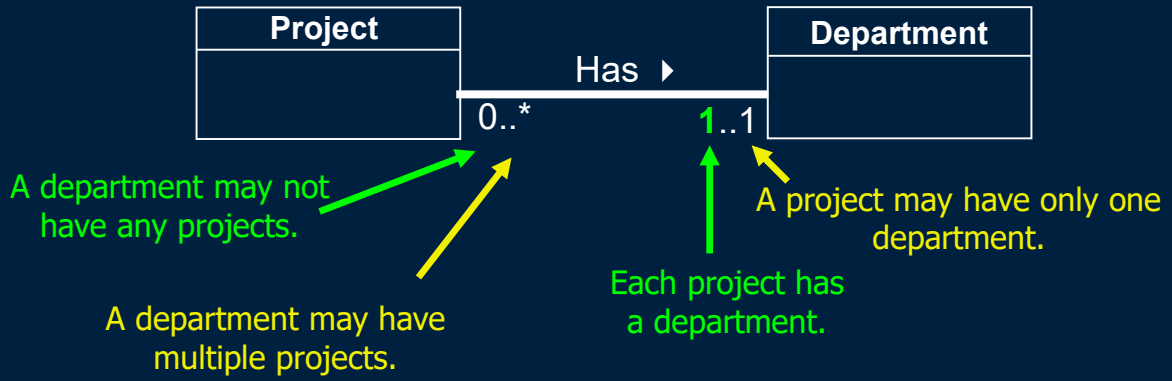
- Participation can be *optional* (zero) or *mandatory (1 or more)*.

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*.

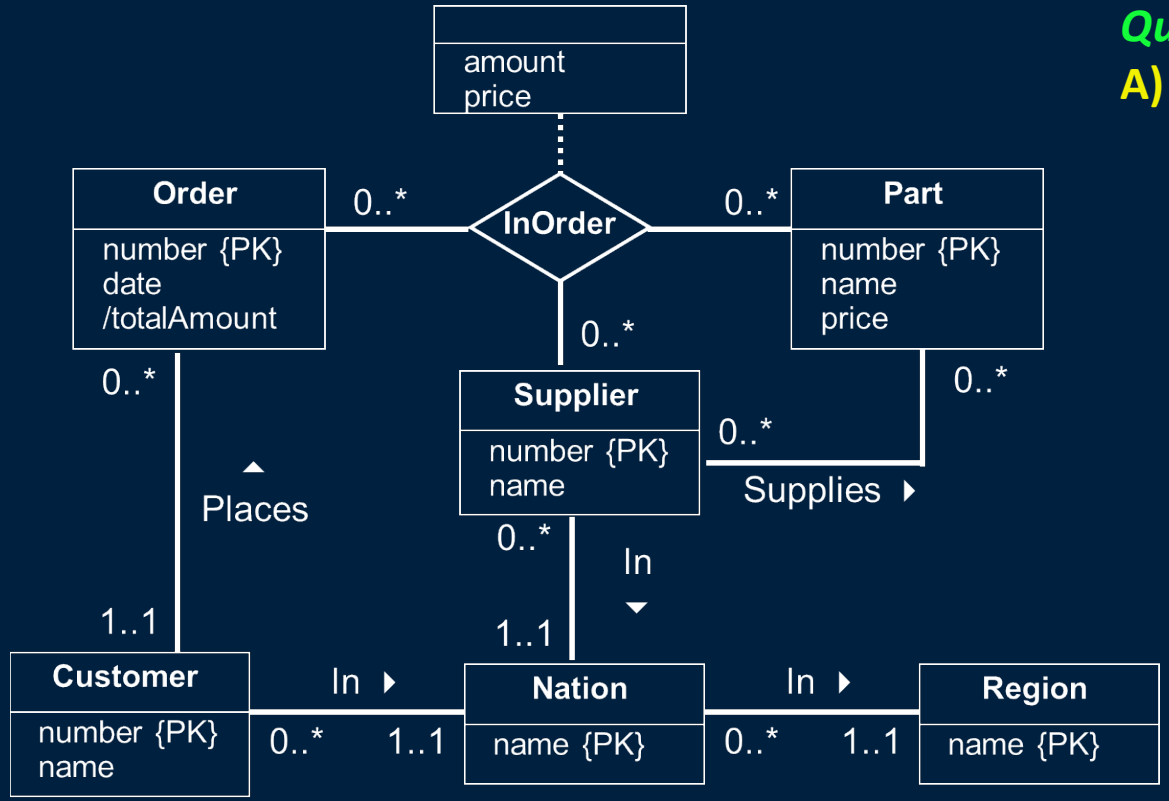
Participation Constraints Example

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).

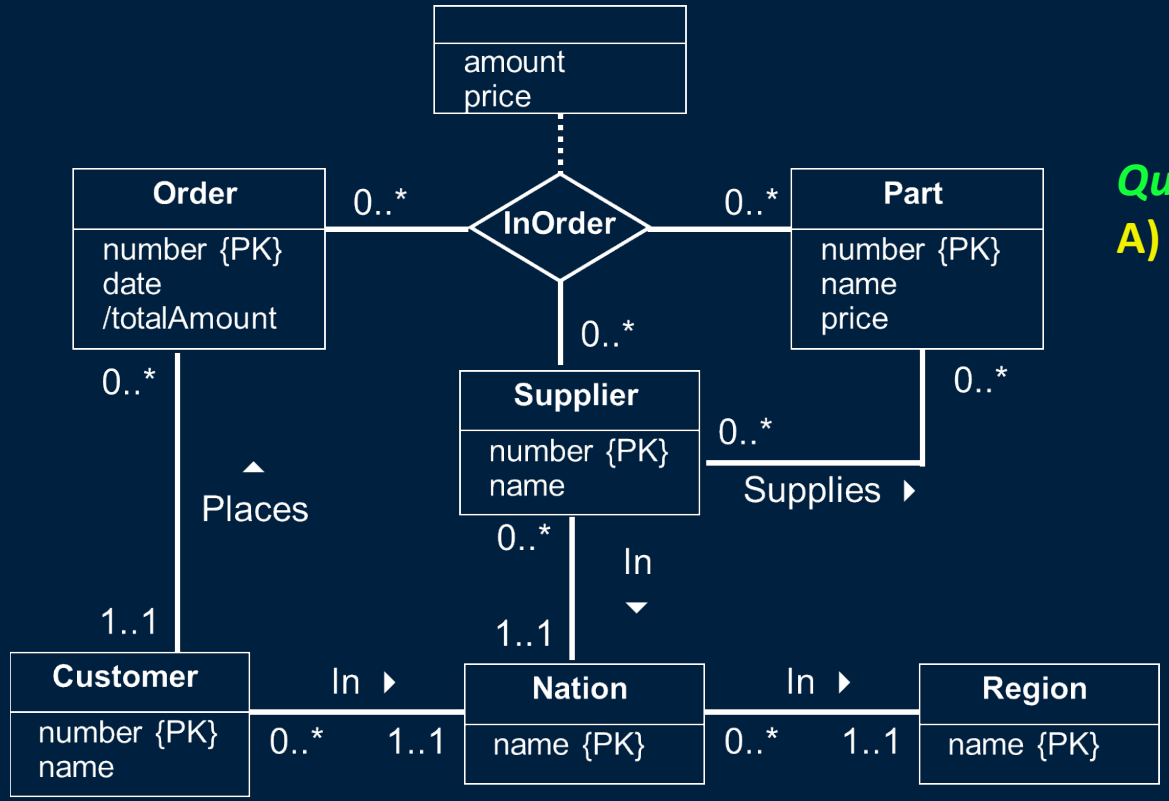
TPC-H ER Diagram Questions



Question 1: How many entity types?

A) 5 **B)** 6 **C)** 7 **D)** 8 **E)** 9

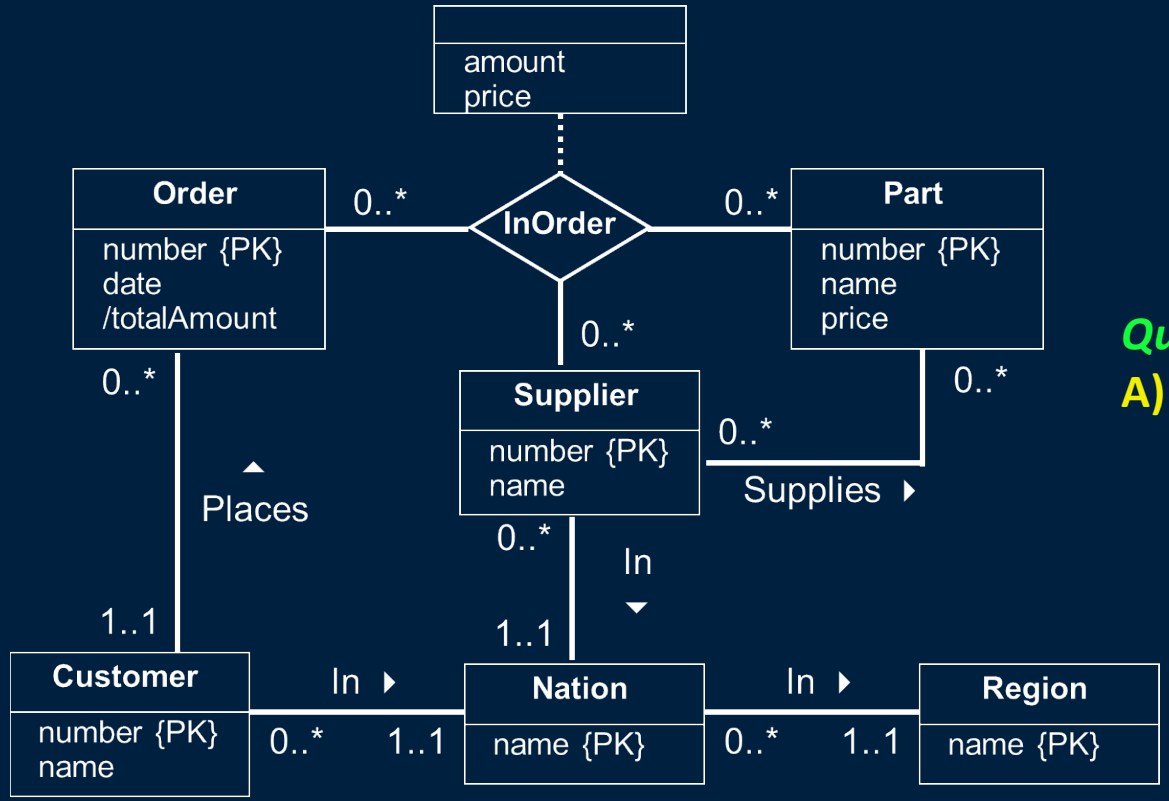
TPC-H ER Diagram Questions



Question 2: How many relationships?

A) 4 **B)** 5 **C)** 6 **D)** 7 **E)** 8

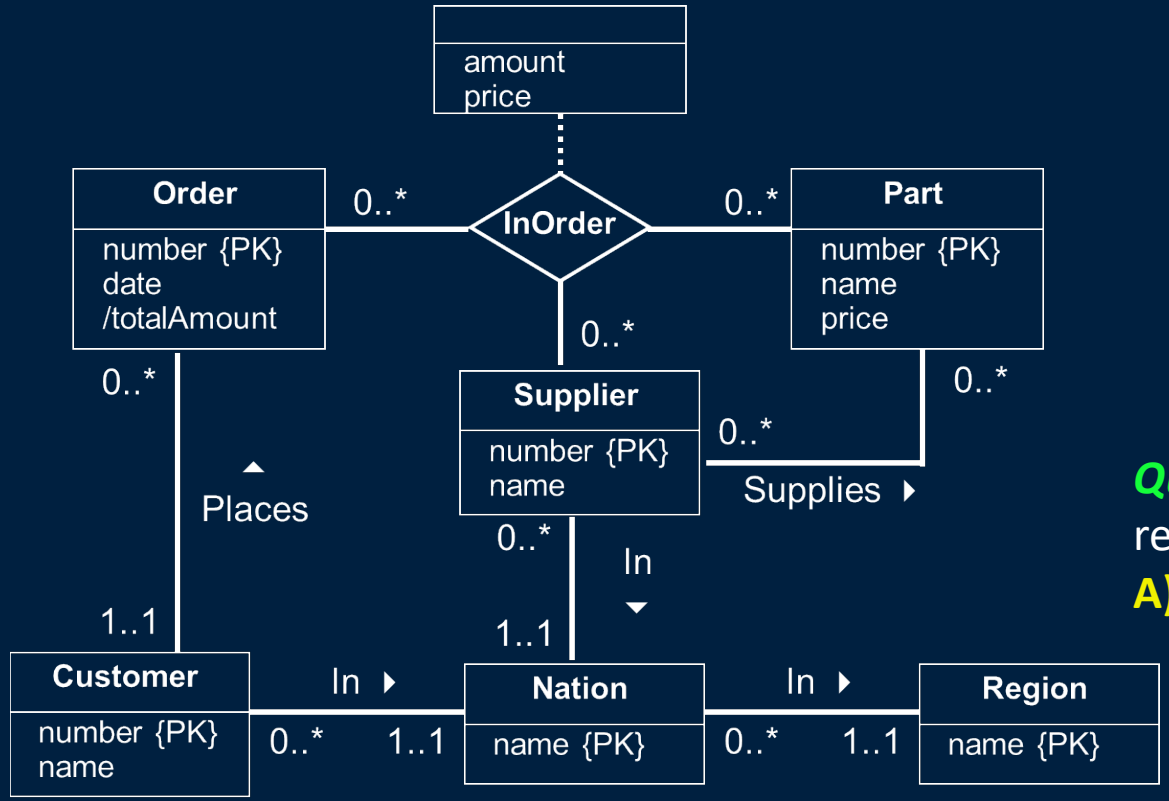
TPC-H ER Diagram Questions



Question 3: How many primary keys?

A) 4 **B)** 5 **C)** 6 **D)** 7 **E)** 8

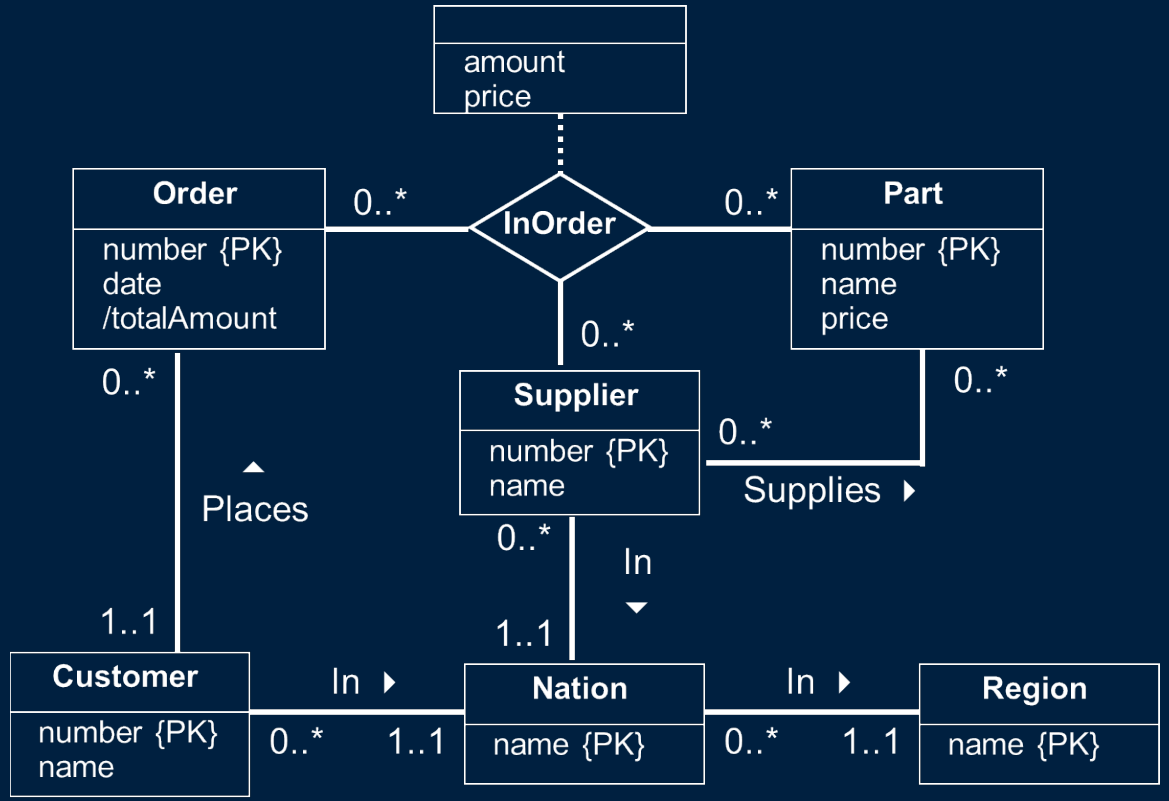
TPC-H ER Diagram Questions



Question 4: How many 1-N relationships?

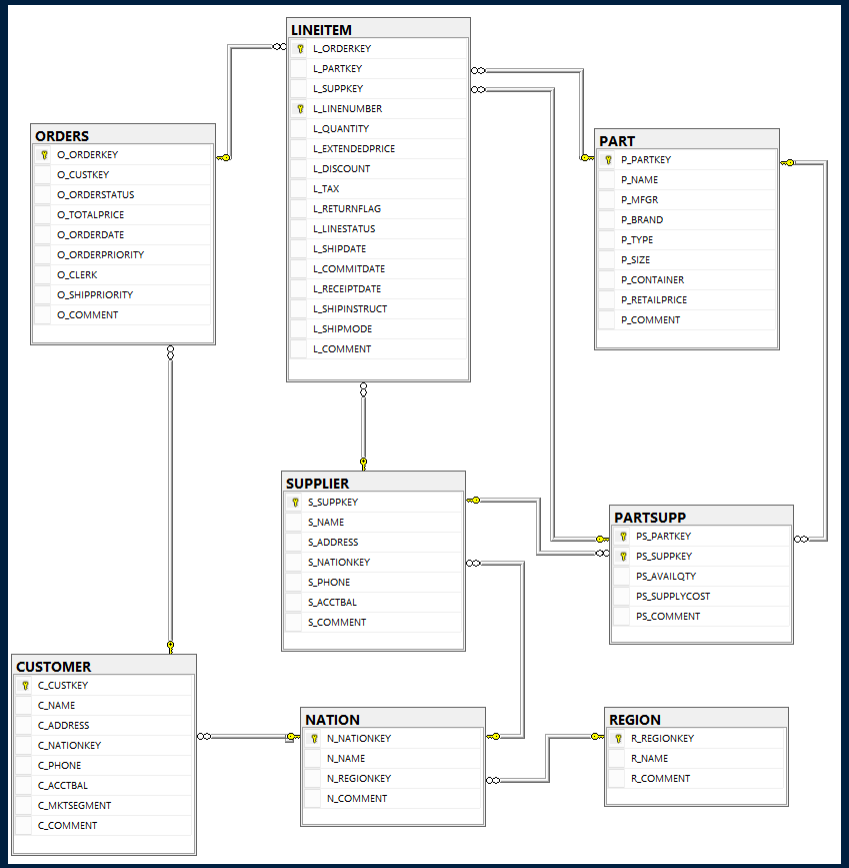
A) 4 **B)** 5 **C)** 6 **D)** 7 **E)** 8

TPC-H ER Diagram Questions



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

Other Notations – Logical/Relational Diagram

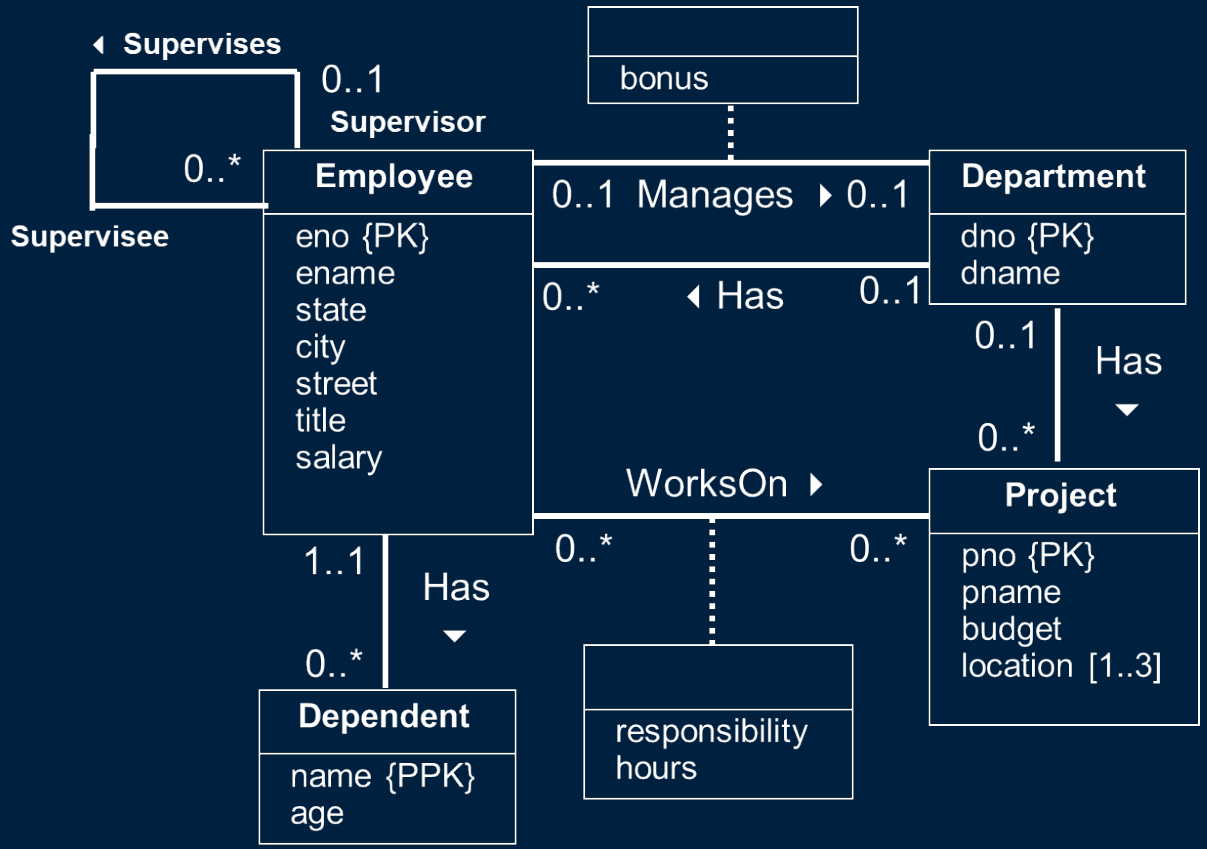


ER Model to Relational Schemas

Converting an ER model to a relational database schema involves 7 steps.

In general, these steps convert entities to relations and ER relationships to relations. For 1:1 and 1:N relationships, foreign keys are used instead of separate relations.

ER to Relational Example



ER to Relational Mapping
Step #1: Convert Strong Entities

Step #1: Convert each strong entity to a relation.

Employee (<u>eno</u>, ename, state, city, street, title, salary)

- Notes:
  - Attributes of the entity type become attributes of the relation.
  - Multi-valued attributes are handled separately (in step #6).
  - The primary key of the relation is the key attributes for the entity.

ER to Relational Mapping
Current Relational Schema - Step #1

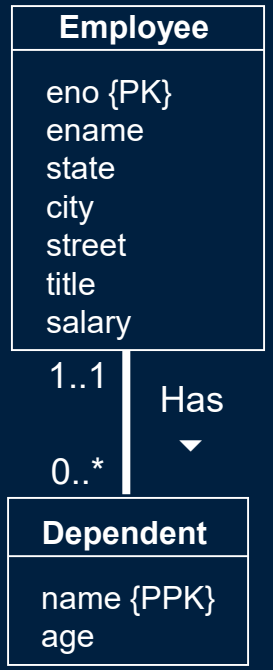
Employee (eno, ename, state, city, street, title, salary)

Project (pno, pname, budget)

Department (dno, dname)

ER to Relational Mapping
Step #2: Convert Weak Entities

Step #2: Convert each weak entity into a relation with foreign keys to its identifying relations (entities). Example:



Employee (eno, ename, state, city, street, title, salary)

Dependent (eno, name, age)

Dependent.eno is a FK to Employee.eno

ER to Relational Mapping
Current Relational Schema - Step #2

Dependent (eno, name, age)

Employee (eno, ename, state, city, street, title, salary)

Project (pno, pname, budget)

Department (dno, dname)

ER to Relational Mapping
Steps #3-5: Convert Relationships

Steps 3 to 5 convert *binary* relationships of cardinality:

- 1:1 - Step #3
- 1:N - Step #4
- M:N - Step #5

M:N relationships are the most general case, and the conversion algorithm for these relationships can be applied to 1:1 and 1:N as well.

- However, for performance reasons, it is more efficient to perform different conversions for each relationship type.
- In general, each ER relationship can be mapped to a relation. However, for 1:1 and 1:N relationships, **it is more efficient to combine the relationship with an existing relation instead of creating a new one**.

Relationships that are not binary are handled in step #7.

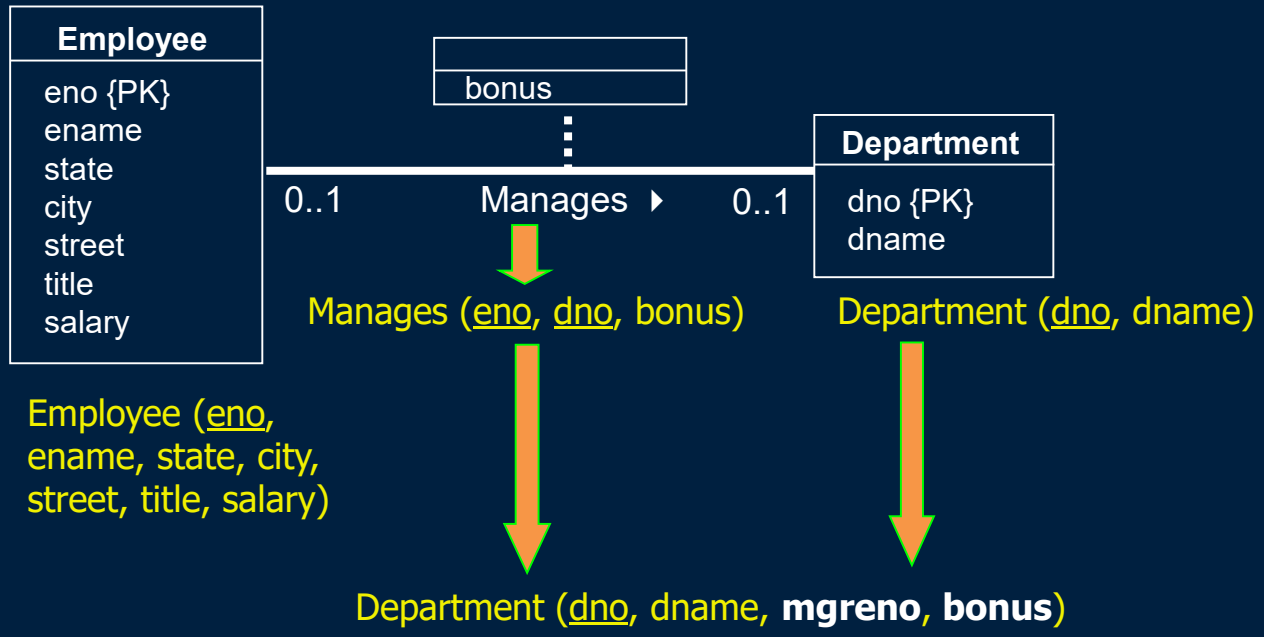
ER to Relational Mapping
Step #3: Convert 1:1 Relationships

Step #3: Convert binary 1:1 relationships into a `UNIQUE` foreign key reference from one relation to the other.

Choose one of the relations, say *R*, and:

- Add the attributes of the relationship to *R*.
- Add the primary key attributes of the other relation to *R*, and create a foreign key reference to the other relation.
- Declare these added primary key attributes of *R* to be `UNIQUE`.
- Note: You can select either relation, but it is best to select the relation that is guaranteed to always participate in the relationship or the one that will participate the most in the relationship.

ER to Relational Mapping
Step #3: Convert 1:1 Relationships (2)



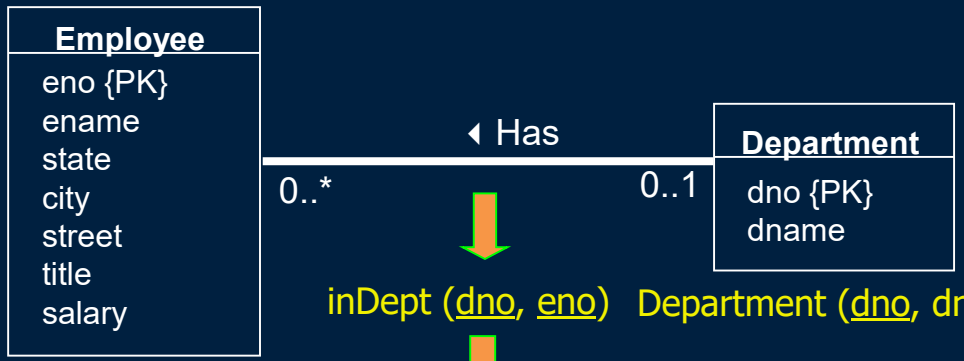
Note: Renamed eno to mgreno for clarity.

ER to Relational Mapping
Step #4: Convert 1:N Relationships

Step #4: Convert binary 1:N relationships into a foreign key reference from the N-side relation to the 1-side relation.

Note: Unlike 1:1 relationships, you must select the N-side of the relationship as the relation containing the foreign key and relationship attributes.

ER to Relational Mapping
Step #4: Convert 1:N Relationships



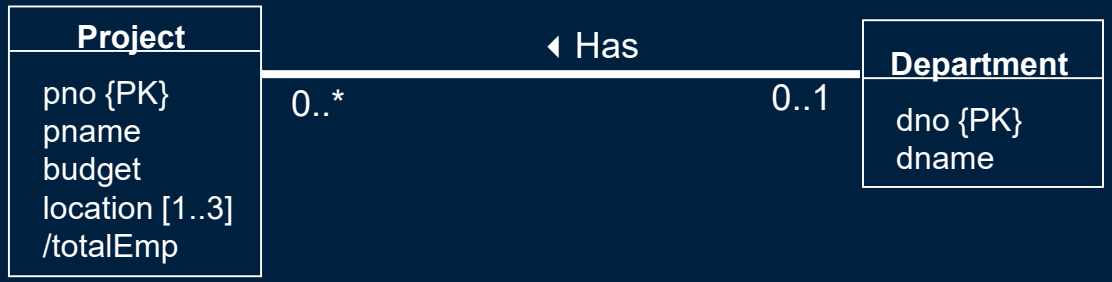
inDept (<u>dno</u>, <u>eno</u>)

Department (<u>dno</u>, dname, mgreno, bonus)

Employee (<u>eno</u>, ename, state, city, street, title, salary)

Employee (<u>eno</u>, ename, state, city, street, title, salary, **dno**)

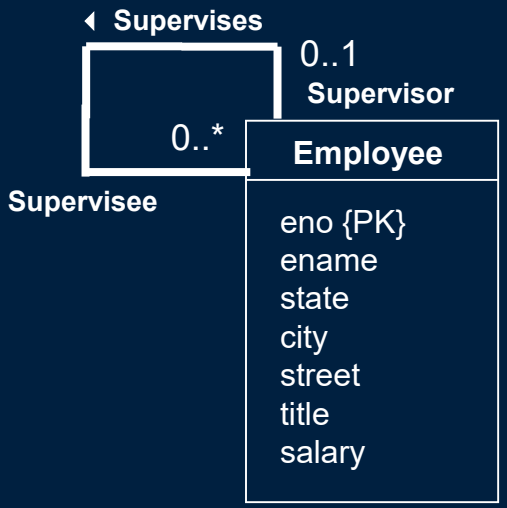
ER to Relational Mapping
Step #4: Convert 1:N Relationships



Question: What table should contain the foreign key field?

A) Project **B)** Department **C)** None

ER to Relational Mapping
Step #4: Convert 1:N Relationships

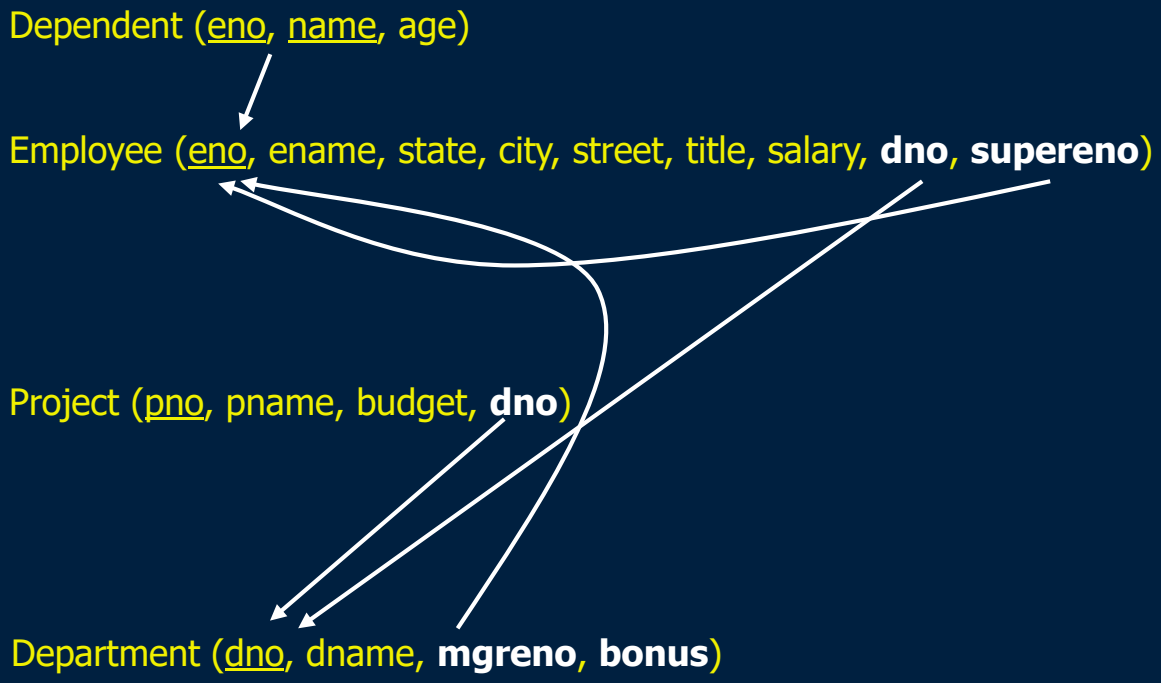


Supervises (<u>supereno</u>, <u>eno</u>)

Employee (<u>eno</u>, ename, state, city, street, title, salary, dno)

Employee (<u>eno</u>, ename, state, city, street, title, salary, dno, **supereno**)

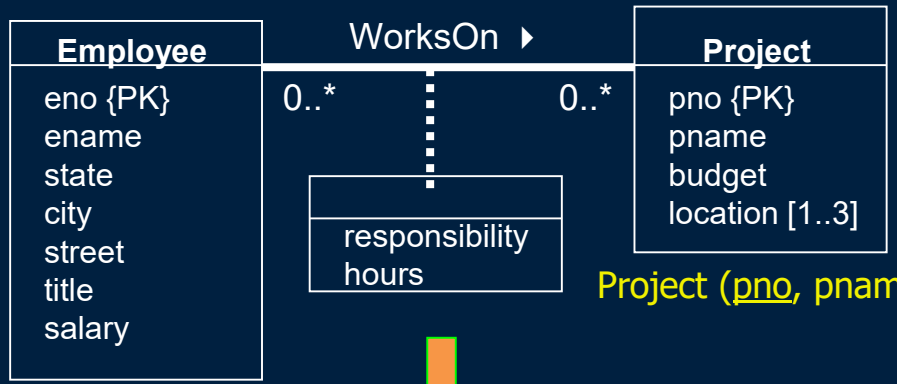
ER to Relational Mapping
Current Relational Schema - Step #4



ER to Relational Mapping
Step #5: Convert M:N Relationships

Step #5: Convert binary M:N relationships into a **new relation** with foreign keys to the two participating entities.

- The primary key consists of the primary keys of the two relations.

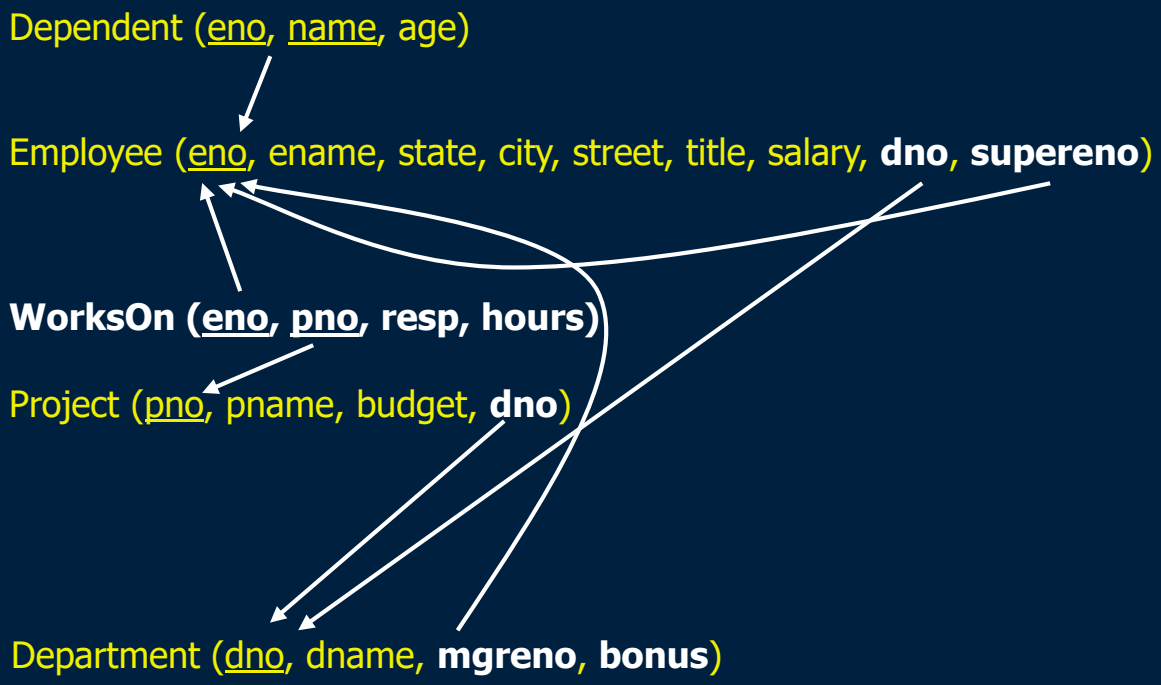


Project (pno, pname, budget, dno)

Employee (eno, ename, state, city, street, title, salary, dno, supereno)

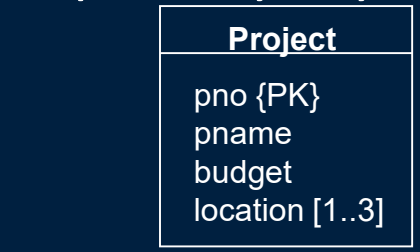
WorksOn (eno, pno, resp, hours)

ER to Relational Mapping
Current Relational Schema - Step #5



ER to Relational Mapping
Step #6: Convert Multi-Valued Attributes

Step #6: Convert a multi-valued attribute into a relation with composite (two or more attributes) primary key consisting of the attribute value plus the primary key of the attribute's entity.

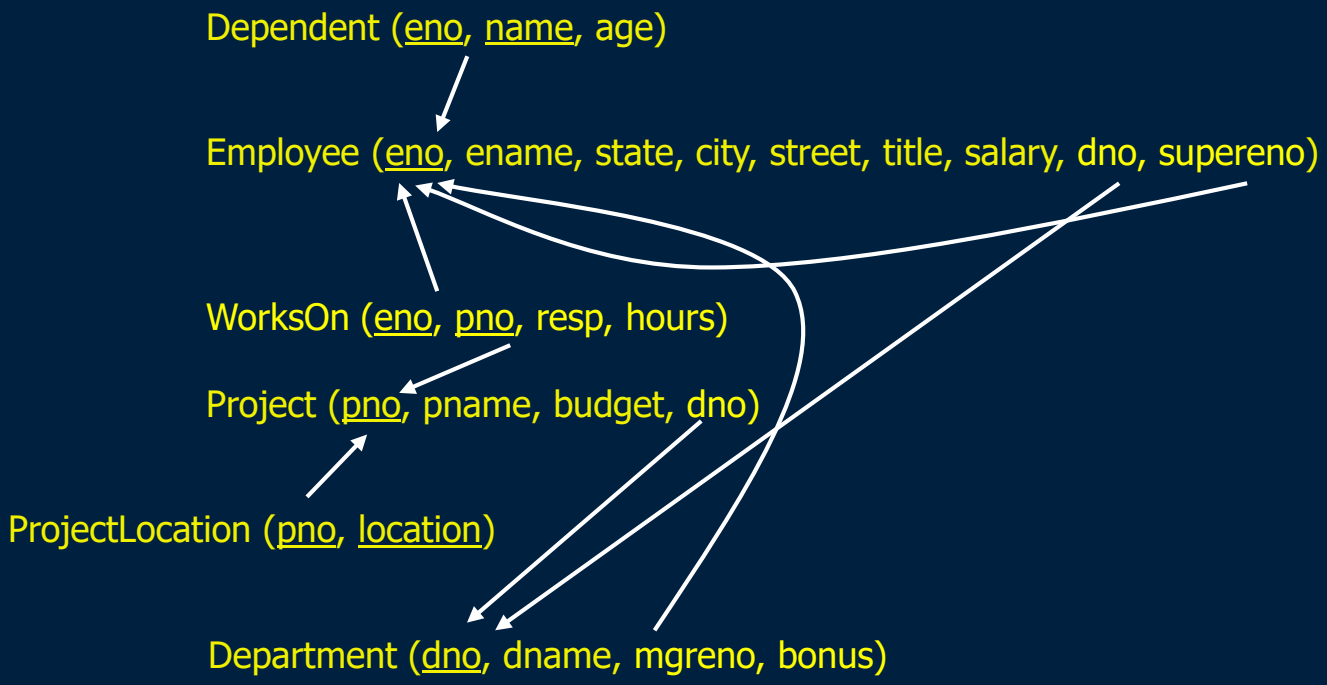


Project (pno, pname, budget, dno)

ProjectLocation (pno, location)

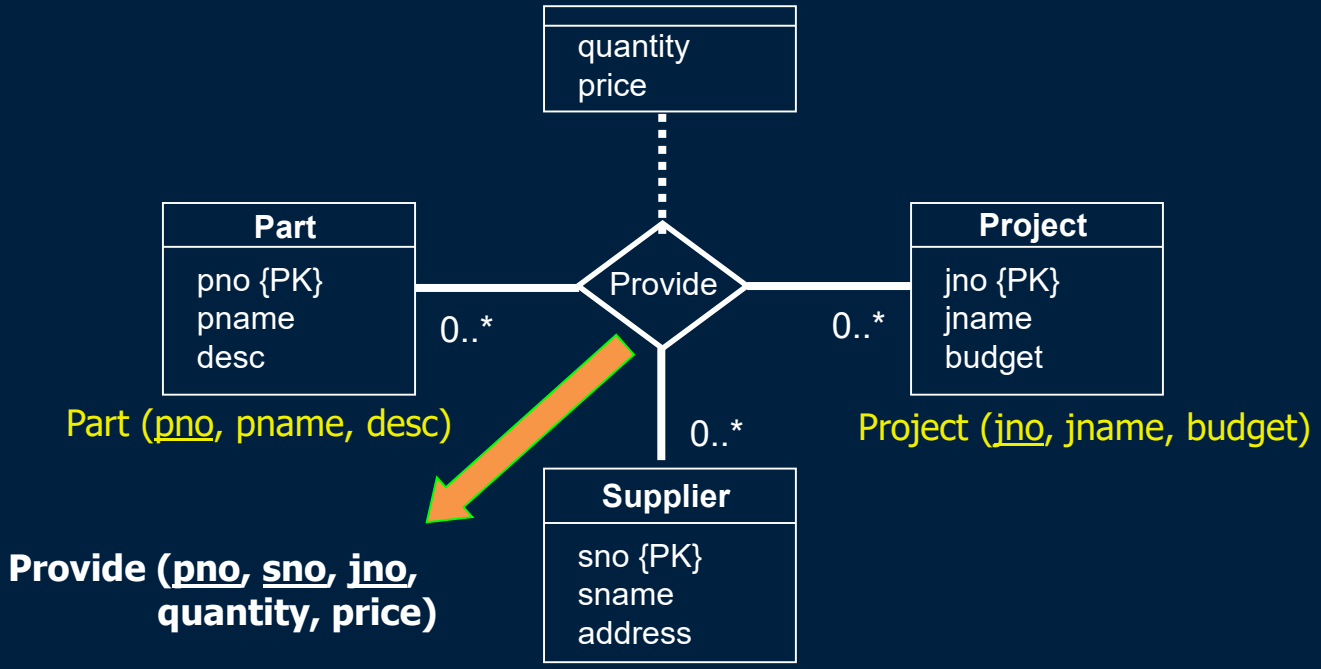
Project (pno, pname, budget, dno)

ER to Relational Mapping
Final Relational Schema



ER to Relational Mapping
Step #7: Convert n-ary Relationships

Step #7: Convert *n*-ary relationships by creating a new relation with primary keys from all entities and foreign keys back to them.



Summary of ER to Relational Mapping

ER Model	Relational Model
Entity Type	Relation
1:1 or 1:N Relationship Type	Foreign key (from N-side to 1-side)
M:N Relationship Type	"Relationship" relation and 2 foreign keys
n-ary Relationship Type	"Relationship" relation and *n* foreign keys
Multi-valued attribute	Relation and foreign key
Key attribute	Primary key attribute

ER to Relational Mapping Question

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

1) The M:N relationship mapping rule could be applied to 1:1 and 1:N relationships, as it is more general.

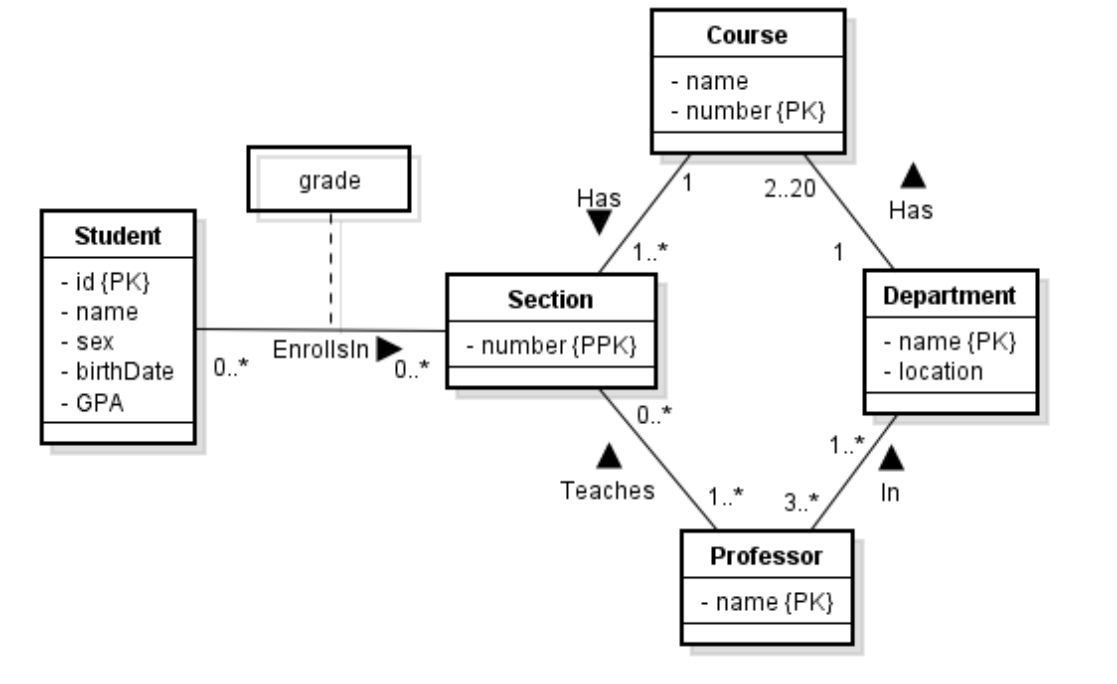
2) A M:N relationship is mapped to a relation in the relational model.

3) The designer has a choice on which side to put the foreign key when mapping a 1:N relationship.

4) When mapping a multi-value attribute, the new table containing the multi-value attribute will have a composite(two or more attributes) primary key.

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

ER to Relational Mapping University Question



Question: Convert to the relational model.

Conclusion

A data analyst often must understand database design documentation.

Database design is divided into three phases:

- Conceptual database design
- Logical database design
- Physical database design

ER (conceptual) design is performed at a high-level of abstraction involving entities, relationships, and attributes.

- There are a variety of different diagram syntax. We used UML syntax.
- Watch out for difference between ER (conceptual) and relational (logical) diagrams.

The algorithm for converting ER models to relational schemas involves 7 steps. A data analyst can write queries using only a database design.


UBC
THE UNIVERSITY OF BRITISH COLUMBIA