Database Design Entity-Relationship (ER) Modeling

COSC 304 – Introduction to Database Systems

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
- Have you been on a project that failed? A) Yes B) No

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

Software and System Development

Software development follows an iterative process with steps:

- ***Specification*** – capturing user and system requirements, goals, and timelines
  - *Top-down design* by specifying entities, attributes, and relationships.
- ***Design*** – develops models for system architecture and behavior. Design database and the associated application. Special focus on transactions and UI.
  - Recommend: Solid DB design before prototyping application.
- ***Implementation*** – build database statements and program code.
- ***Testing*** - executing programs to determine errors and issues.
- ***Maintenance*** – monitoring and maintaining the system after installation.
  - DBMS maintenance includes monitoring performance, security, and upgrades.

Specification

Specification involves:

- understanding how the project fits into the organization
- project goals and success outcomes
- project timelines and deliverables
- measurement criteria for determining project success
- information on users and user requirements
- standards for database/application development that must be followed
- documentation of privacy and security concerns
- legal issues including copyright when dealing with outside developers
- information on how the project will interface with other systems

Specification: Mission Statements

The ***mission statement*** specifies the major project objectives with defined metrics to evaluate if successfully completed.

NASA's mission statement when going to the moon:

"I believe that this nation should commit itself to achieving the goal, before this decade is out, of landing a man on the moon and returning him safely to Earth." (John F. Kennedy May 25, 1961)

- NASA fulfilled that goal on July 20, 1969, when Apollo 11's lunar module *Eagle* touched down in the Sea of Tranquility, with Neil Armstrong and Buzz Aldrin aboard. A dozen men would walk on the moon before the Apollo program ended. The last of those men, Gene Cernan, left the desolate lunar surface with these words: "We leave as we came and, God willing, as we shall return, with peace and hope for all mankind."

Specification: The "Project Champion"

The "**Project Champion**" is a manager or senior IT person who is the project's promoter and backer.

Many projects fail because no one takes ownership of them.

- Consequently, they take too long, go over budget, and are never deployed effectively.
- When hiring outside consultants, make sure somebody in the organization is the Project Champion.
- For internal projects, a Project Champion is especially important as there are always conflicts over money, developer time, and political issues on making users work on new applications.

Bottom line: If no one is willing to be the champion for a project, it is likely that project will not achieve its goals.

Specification: Requirements Gathering

Requirements gathering collects details on organizational processes and user issues.

- Often the organization itself does not know this information, and it can only be determined by collecting it from user interviews.
- Through user interviews identify:
  - Who are the users? Group them into classes.
  - What do the users do now? (existing systems/processes)
  - What are the complaints and possibilities for improvement?
- Determine the data used by the organization, identify data relationships, and determine how data is used and generated.
  - Identify unique fields (keys)
  - Determine data dependencies, relationships, and constraints (high-level)
  - Estimate the data sizes and their growth rates

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***.
  - Select a DBMS based on features, performance, price, and interoperability.

Database People: DA and DBA

We have seen these two database people before:

- ***Database administrator*** (***DBA***) - responsible for installing, maintaining, and configuring the DBMS software.
- ***Data administrator*** (***DA***) - responsible for organizational policies on data creation, security, and planning.

The DA is involved in the early phases of design including planning the project and conceptual and logical design.

The DBA performs the physical design and actively manages deployed, production systems.

Another common position is a (data) ***architect*** that selects systems to use, makes design decisions, and evaluates current and future systems at the architectural level.

Entity-Relationship Modeling

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

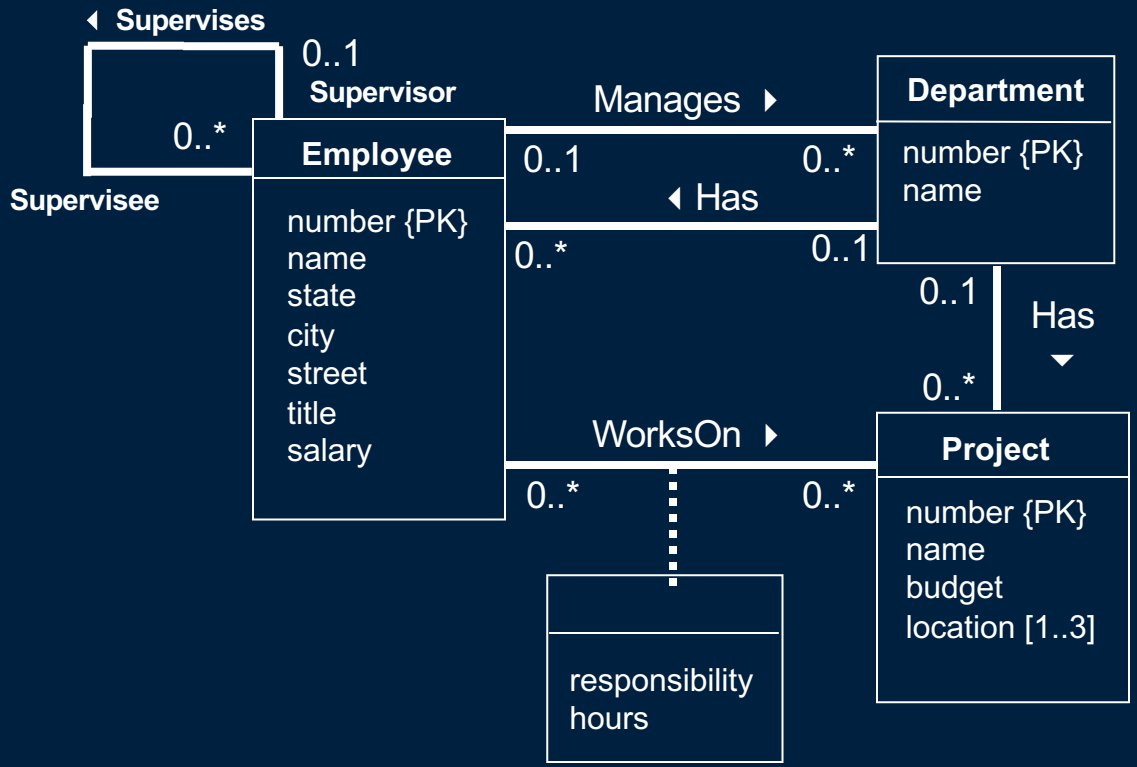
- The entity-relationship (ER) model was proposed by Peter Chen in 1976. We will perform ER modeling using Unified Modeling Language (UML) syntax.

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. An 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: How many of the following statements are ***true***?

1) Entity types are represented using a rectangle box.

2) An entity is always a physical object.

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

4) Employee number is an entity.

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

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.

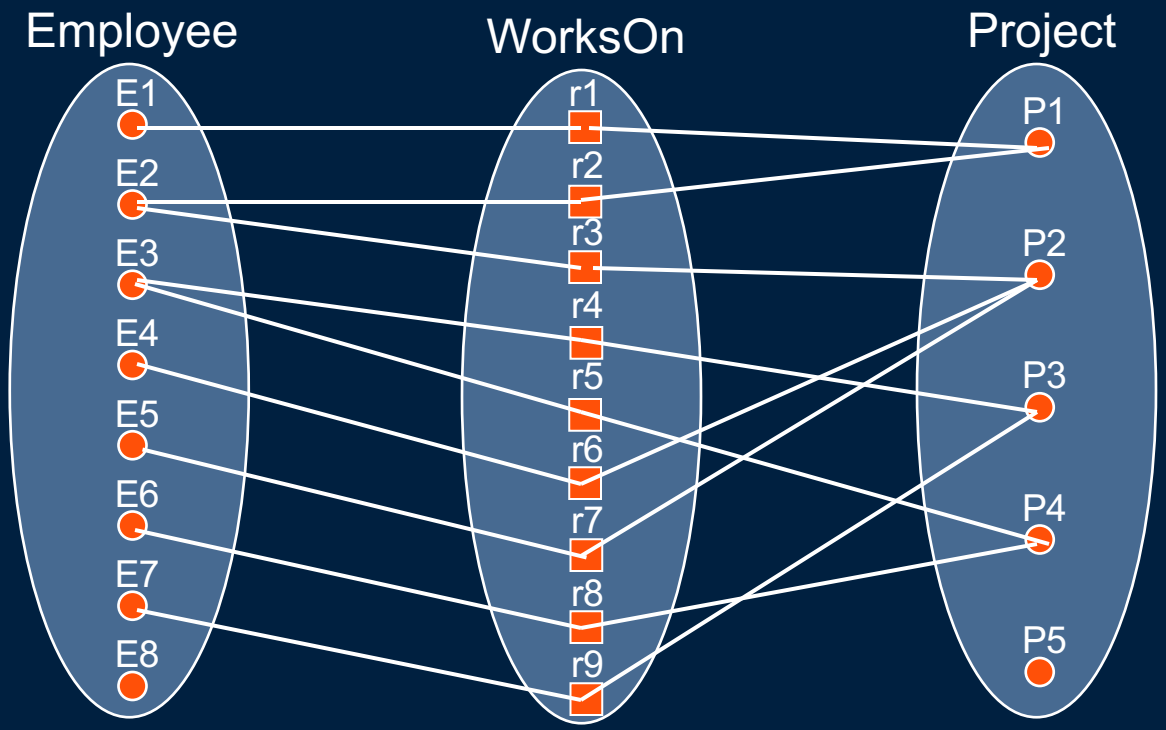
There can be more than one relationship between two entity types.

Representing Relationship Types

The relationship type is represented as a labeled edge between the two entity types. The label is applied only in one direction so an arrow indicates the correct way to read it.

- A relationship type name is normally a verb or verb phrase.
- The first letter of each word in the name is capitalized.
- **Do not put arrows on either end of the line.**

Visualizing Relationships



Note: This is an example of a many-to-many relationship. A project can have more than one employee, and an employee can work on more than one project.

Relationship Degree

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

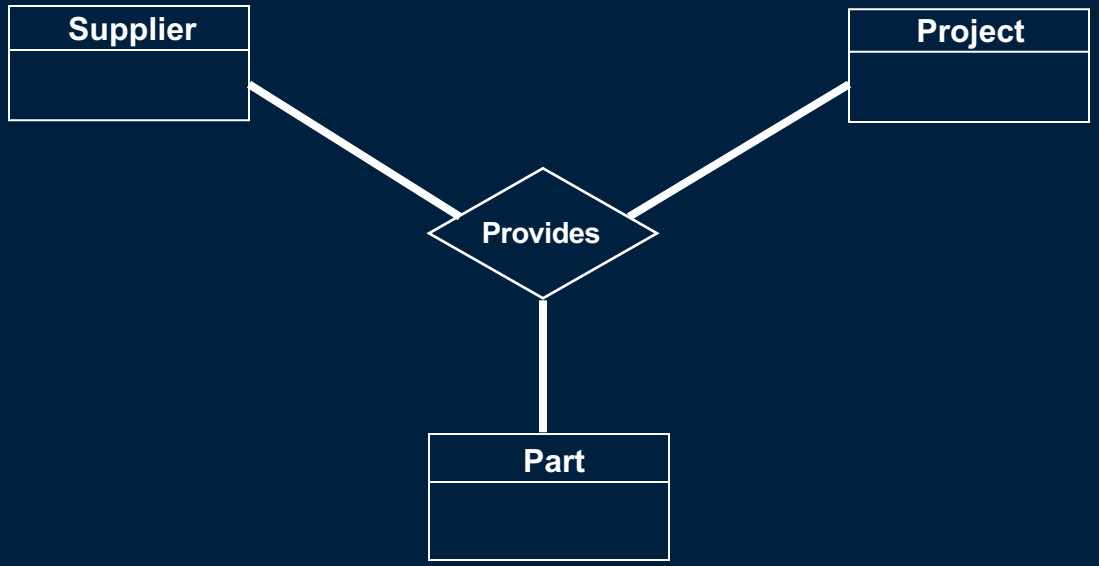
- For example, WorksOn is a relationship type of degree two as the two participating entity types are Employee and Project.
  - Note: This is not the same as degree of a relation which was the number of attributes in a relation.

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



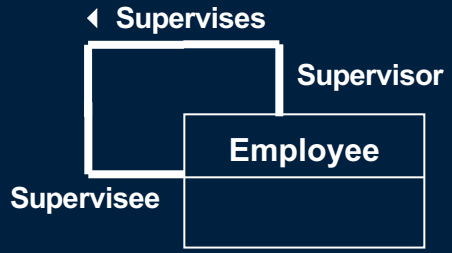
A project may require a part from multiple different suppliers.

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 adjectives.
- 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.
- An attribute may be simple if it contains a single component (e.g. salary) or composite if it contains multiple components (e.g. address).
  - Question: Is the name attribute of Employee simple or composite?
- A ***derived attribute*** is an attribute whose value is calculated from other attributes but is not physically stored.

Representing Attributes

In UML attributes are listed in the rectangle for their entity. Tags are used to denote any special features of the attributes.

- 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.

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) A composite attribute contains two or more components.

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

5) Attribute names are typically verbs.

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

Representing Attributes and Keys

A ***candidate key*** is a minimal set of attributes that uniquely identifies each instance of an entity type.

A ***primary key*** is a candidate key that is selected by the designer to identify 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.

A ***composite key*** is a key that consists of two or more attributes.

Key Question

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

1) It is possible to have two candidate keys with different numbers of attributes.

2) A composite key has more than 1 attribute.

3) The computer picks the primary key used in the design.

4) A relationship has a primary key.

5) An attribute has a primary key.

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

Attributes on Relationships

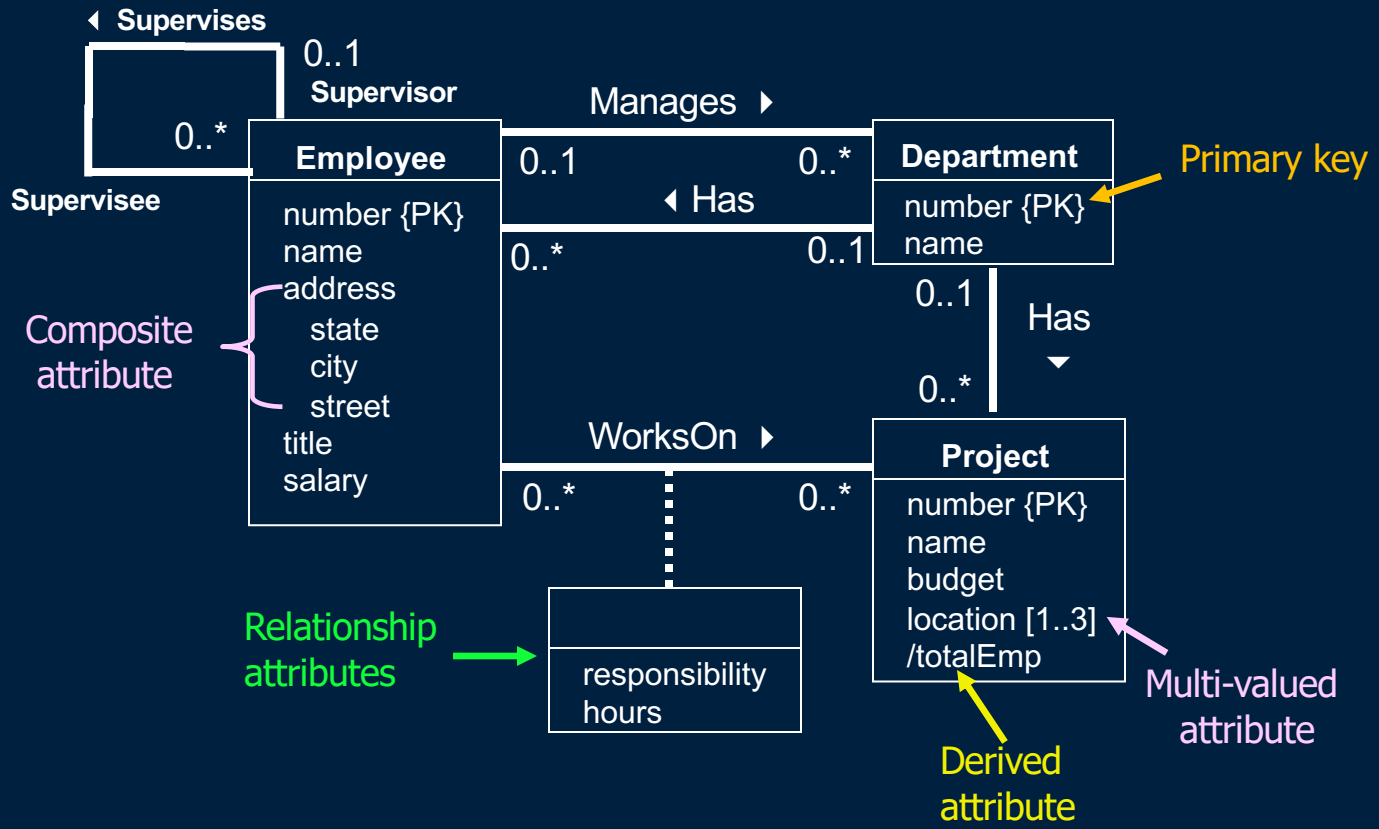
An attribute may be associated with a relationship type.

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.

Attributes in UML Notation



ER Design Question #1

Construct a university database where:

- Each student has an id, name, sex, birth date, and GPA.
- Each professor has a name and is in a department.
- Each department offers courses and has professors. A department has a name and a building location.
- Each course has a name and number and may have multiple sections.
- Each section is taught by a professor and has a section number.
- Students enroll in sections of courses. They may only enroll in a course once (and in a single section). A student receives a grade for each of their course sections.

Conclusion

Database design is divided into three phases:

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

Effective requirements gathering is an important skill to master. Projects should have a well-defined mission statement and project champion to increase their probability of success.

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.

Objectives

- Describe the three steps in database design including the results of each step.
- Describe differences between conceptual, logical, and physical data models.
- Describe how the roles of DBA and DA fit into database design. What do these people do?
- 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

Be able to model a domain explained in an English paragraph in an ER diagram using UML notation. 


THE UNIVERSITY OF BRITISH COLUMBIA