CO226: Database Systems

Data Modeling

Sampath Deegalla
dsdeegalla@pdn.ac.lk

26th June 2014

ER Model

- High-level conceptual data model can be used to create a conceptual schema for the database.
- Entity-Relationship (ER) model is one such data model that can be used for the conceptual design of database applications.
- The output of the ER modeling is known as ER diagrams.
- ER model describes data as entities, relationships and attributes.

ER Model

- High-level conceptual data model can be used to create a conceptual schema for the database.
- Entity-Relationship (ER) model is one such data model that can be used for the conceptual design of database applications.
- The output of the ER modeling is known as ER diagrams.
- ER model describes data as entities, relationships and attributes.

ER Model

- High-level conceptual data model can be used to create a conceptual schema for the database.
- Entity-Relationship (ER) model is one such data model that can be used for the conceptual design of database applications.
- The output of the ER modeling is known as ER diagrams.
- ER model describes data as entities, relationships and attributes.

ER Model

- High-level conceptual data model can be used to create a conceptual schema for the database.
- Entity-Relationship (ER) model is one such data model that can be used for the conceptual design of database applications.
- The output of the ER modeling is known as ER diagrams.
- ER model describes data as entities, relationships and attributes.

ER Model

- High-level conceptual data model can be used to create a conceptual schema for the database.
- Entity-Relationship (ER) model is one such data model that can be used for the conceptual design of database applications.
- The output of the ER modeling is known as ER diagrams.
- ER model describes data as entities, relationships and attributes.

An Example Database Application: COMPANY

- The company is organized into DEPARTMENTs. Each department has a unique name, a unique number, and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department controls a number of PROJECTs. Each project has a unique name, a unique number and is located at a single location.
- We store each EMPLOYEE's name, social security number, address, salary, sex, and birth date. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs. For each dependent, we keep their name, sex, birth date, and relationship to the employee.

An Example Database Application: COMPANY

- The company is organized into DEPARTMENTs. Each department has a unique name, a unique number, and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department controls a number of PROJECTs. Each project has a unique name, a unique number and is located at a single location.
- We store each EMPLOYEE's name, social security number, address, salary, sex, and birth date. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs. For each dependent, we keep their name, sex, birth date, and relationship to the employee.

An Example Database Application: COMPANY

- The company is organized into DEPARTMENTs. Each department has a unique name, a unique number, and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department controls a number of PROJECTs. Each project has a unique name, a unique number and is located at a single location.
- We store each EMPLOYEE's name, social security number, address, salary, sex, and birth date. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs. For each dependent, we keep their name, sex, birth date, and relationship to the employee.

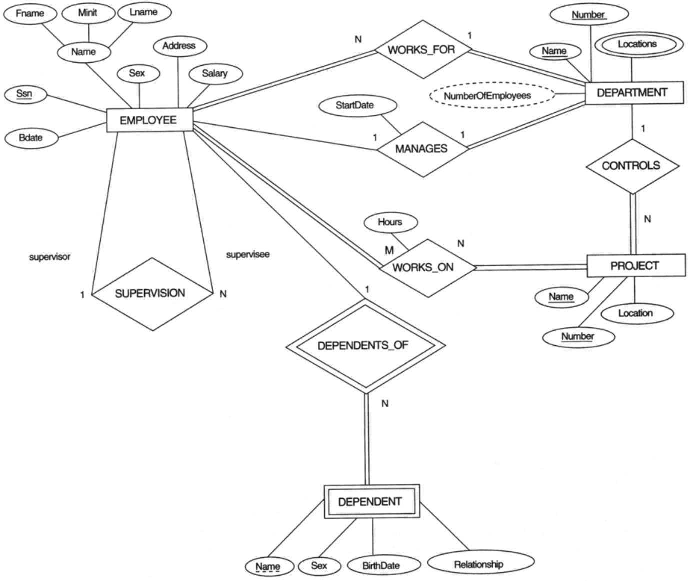
An Example Database Application: COMPANY

- The company is organized into DEPARTMENTs. Each department has a unique name, a unique number, and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department controls a number of PROJECTs. Each project has a unique name, a unique number and is located at a single location.
- We store each EMPLOYEE's name, social security number, address, salary, sex, and birth date. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs. For each dependent, we keep their name, sex, birth date, and relationship to the employee.

An Example Database Application: COMPANY

- The company is organized into DEPARTMENTs. Each department has a unique name, a unique number, and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department controls a number of PROJECTs. Each project has a unique name, a unique number and is located at a single location.
- We store each EMPLOYEE's name, social security number, address, salary, sex, and birth date. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs. For each dependent, we keep their name, sex, birth date, and relationship to the employee.

An ER schema diagram for the COMPANY database



Entities and Attributes

- Entities are specific objects or things in the mini-world that are represented in the database.
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT.
- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have a Name, SSN, Address, Sex and BirthDate.
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M' and BirthDate='09-JAN-55.'
- Each attribute has a value set(or data type) associated with it
  - e.g. integer, string, subrange, enumerated type, ...

Entities and Attributes

- Entities are specific objects or things in the mini-world that are represented in the database.
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT.
- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have a Name, SSN, Address, Sex and BirthDate.
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M' and BirthDate='09-JAN-55.'
- Each attribute has a value set(or data type) associated with it
  - e.g. integer, string, subrange, enumerated type, ...

Entities and Attributes

- Entities are specific objects or things in the mini-world that are represented in the database.
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT.
- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have a Name, SSN, Address, Sex and BirthDate.
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M' and BirthDate='09-JAN-55.'
- Each attribute has a value set(or data type) associated with it
  - e.g. integer, string, subrange, enumerated type, ...

Entities and Attributes

- Entities are specific objects or things in the mini-world that are represented in the database.
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT.
- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have a Name, SSN, Address, Sex and BirthDate.
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M' and BirthDate='09-JAN-55.'
- Each attribute has a value set(or data type) associated with it
  - e.g. integer, string, subrange, enumerated type, . . .

Entities and Attributes

- Entities are specific objects or things in the mini-world that are represented in the database.
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT.
- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have a Name, SSN, Address, Sex and BirthDate.
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M' and BirthDate='09-JAN-55.'
- Each attribute has a value set(or data type) associated with it
  - e.g. integer, string, subrange, enumerated type, . . .

Types of Attributes

- Simple
  - Each entity has a single atomic value for the attribute.
    - For example, SSN or Sex.
- Composite
  - The attribute may be composed of several components.
    - For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName).
    - Composition may form a hierarchy where some components are themselves composite.

Types of Attributes

- Simple
  - Each entity has a single atomic value for the attribute.
    - For example, SSN or Sex.
- Composite
  - The attribute may be composed of several components.
    - For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName).
    - Composition may form a hierarchy where some components are themselves composite.

Types of Attributes

- Simple
  - Each entity has a single atomic value for the attribute.
    - For example, SSN or Sex.
- Composite
  - The attribute may be composed of several components.
    - For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName).
    - Composition may form a hierarchy where some components are themselves composite.

Types of Attributes

- Multi-valued
  - An entity may have multiple values for that attribute.
    - For example, Color of a CAR or PreviousDegreesof a STUDENT. Denoted as {Color} or {PreviousDegrees}.
- Derived
  - Attribute which could be derived from another attribute.
    - For example, Age could be derived from DateofBirth attribute.
- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.

  - For example, PreviousDegreesof a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees(College, Year, Degree, Field)}.

Types of Attributes

- Multi-valued
  - An entity may have multiple values for that attribute.
    - For example, Color of a CAR or PreviousDegreesof a STUDENT. Denoted as {Color} or {PreviousDegrees}.
- Derived
  - Attribute which could be derived from another attribute.
    - For example, Age could be derived from DateofBirth attribute.
- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.
  - For example, PreviousDegreesof a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees(College, Year, Degree, Field)}.

Types of Attributes

- Multi-valued
  - An entity may have multiple values for that attribute.
    - For example, Color of a CAR or PreviousDegreesof a STUDENT. Denoted as {Color} or {PreviousDegrees}.
- Derived
  - Attribute which could be derived from another attribute.
    - For example, Age could be derived from DateofBirth attribute.
- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.
  - For example, PreviousDegreesof a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees(College, Year, Degree, Field)}.

Types of Attributes

- Multi-valued
  - An entity may have multiple values for that attribute.
    - For example, Color of a CAR or PreviousDegreesof a STUDENT. Denoted as {Color} or {PreviousDegrees}.
- Derived
  - Attribute which could be derived from another attribute.
    - For example, Age could be derived from DateofBirth attribute.
- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.

  - For example, PreviousDegreesof a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees(College, Year, Degree, Field)}.

Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the EMPLOYEE entity type or the PROJECT entity type.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute (uniqueness constraint) of the entity type.
  - For example, SSN of EMPLOYEE.
- A key attribute may be composite.
  - For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key. For example, the CAR entity type may have two keys:
  - VehicleIdentificationNumber(VIN) and
  - VehicleTagNumber(Number, State), also known as license_plate number.

Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the EMPLOYEE entity type or the PROJECT entity type.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute (uniqueness constraint) of the entity type.
  - For example, SSN of EMPLOYEE.
- A key attribute may be composite.
  - For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key. For example, the CAR entity type may have two keys:
  - VehicleIdentificationNumber(VIN) and
  - VehicleTagNumber(Number, State), also known as license_plate number.

Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the EMPLOYEE entity type or the PROJECT entity type.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute (uniqueness constraint) of the entity type.
  - For example, SSN of EMPLOYEE.
- A key attribute may be composite.
  - For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key. For example, the CAR entity type may have two keys:
  - VehicleIdentificationNumber(VIN) and
  - VehicleTagNumber(Number, State), also known as license_plate number.

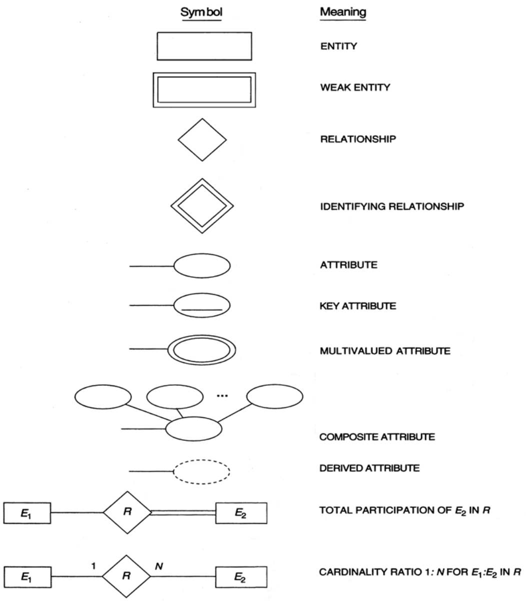
Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the EMPLOYEE entity type or the PROJECT entity type.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute (uniqueness constraint) of the entity type.
  - For example, SSN of EMPLOYEE.
- A key attribute may be composite.
  - For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key. For example, the CAR entity type may have two keys:
  - VehicleIdentificationNumber(VIN) and
  - VehicleTagNumber(Number, State), also known as license_plate number.

Entity Types and Key Attributes

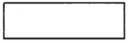
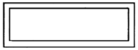
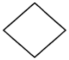
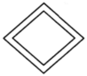
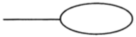
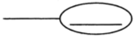
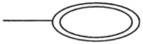
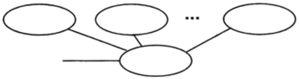
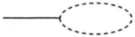
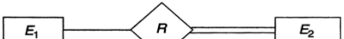
- Entities with the same basic attributes are grouped or typed into an entity type.
    - For example, the EMPLOYEE entity type or the PROJECT entity type.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute (uniqueness constraint) of the entity type.
    - For example, SSN of EMPLOYEE.
- A key attribute may be composite.
    - For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key. For example, the CAR entity type may have two keys:
    - VehicleIdentificationNumber(VIN) and
    - VehicleTagNumber(Number, State), also known as license_plate number.

Summary of the notation for ER diagrams

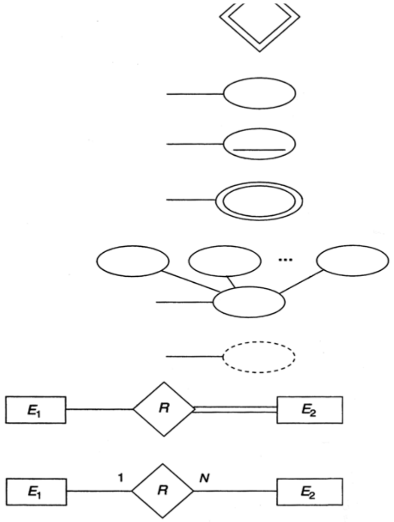


Symbol	Meaning
	ENTITY
	WEAK ENTITY
	RELATIONSHIP
	IDENTIFYING RELATIONSHIP
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
...	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
E_1 R E_2	TOTAL PARTICIPATION OF E_2 IN R
E_1 1 R N E_2	CARDINALITY RATIO 1: N FOR E_1:E_2 IN R

Summary of the notation for ER diagrams

Symbol	Meaning
	ENTITY
	WEAK ENTITY
	RELATIONSHIP
	IDENTIFYING RELATIONSHIP
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	TOTAL PARTICIPATION OF E_2 IN R

Summary of the notation for ER diagrams



IDENTIFYING RELATIONSHIP

ATTRIBUTE

KEY ATTRIBUTE

MULTIVALUED ATTRIBUTE

COMPOSITE ATTRIBUTE

DERIVED ATTRIBUTE

TOTAL PARTICIPATION OF E_2 IN R

CARDINALITY RATIO 1: N FOR E_1:E_2 IN R

Exercise

Using COMPANY database specification, define entity types and attributes using ER notation.

Relationships and Relationship Types

- A relationship relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
  - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships.

Relationships and Relationship Types

- A relationship relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
  - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships.

Relationships and Relationship Types

- A relationship relates two or more distinct entities with a specific meaning.
    - For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
    - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships.

Relationships and Relationship Types

- A relationship relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
  - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships.

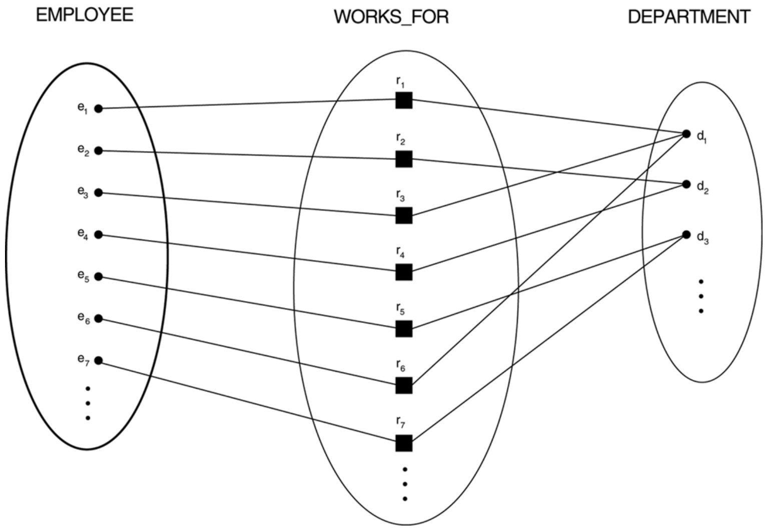
Relationships and Relationship Types

- A relationship relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
  - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships.

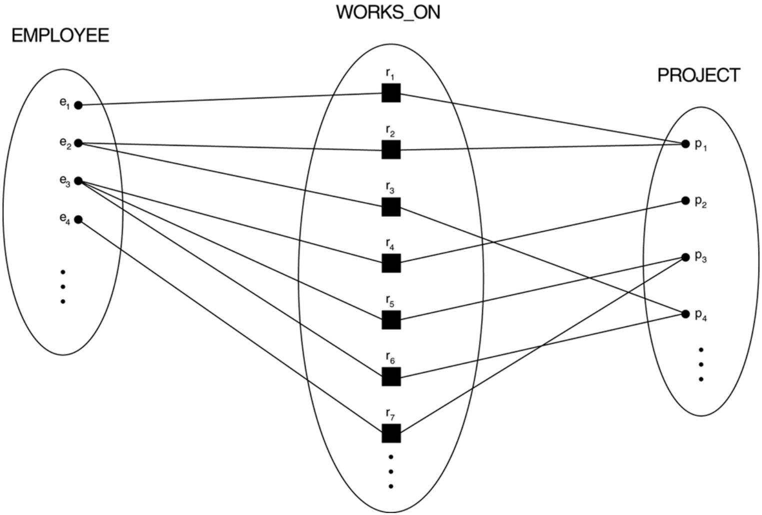
Relationships and Relationship Types

- A relationship relates two or more distinct entities with a specific meaning.
    - For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
    - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships.

Example relationship instances of the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT



Example relationship instances of the WORKS_ON relationship between EMPLOYEE and PROJECT



Relationships and Relationship Types

- More than one relationship type can exist with the same participating entity types. For example, MANAGES and WORKS_FOR are distinct relationships between EMPLOYEE and DEPARTMENT, but with different meanings and different relationship instances.

Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

Example: Suppose that a DEPENDENT entity is identified by the dependent's first name and birhtdate, and the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

Example: Suppose that a DEPENDENT entity is identified by the dependent's first name and birhtdate, and the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

Example: Suppose that a DEPENDENT entity is identified by the dependent's first name and birhtdate, and the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

Example: Suppose that a DEPENDENT entity is identified by the dependent's first name and birhtdate, and the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

  Example: Suppose that a DEPENDENT entity is identified by the dependent's first name and birhtdate, and the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

Exercise

Using COMPANY database specification, define relationship types using ER notation.

Constraints on Relationship Types

- Maximum Cardinality (Cardinality Ratios)
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N) ER diagram
- Minimum Cardinality (Participation Constraints and Existence Dependencies)
  - zero (optional participation, not existence-dependent)
  - one or more (mandatory, existence-dependent)

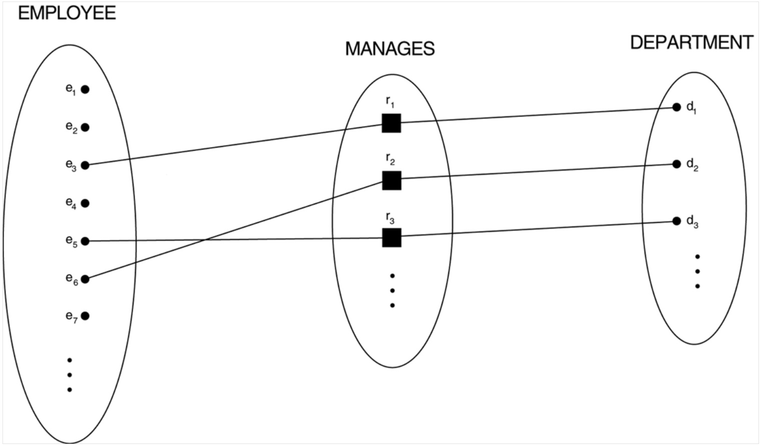
Constraints on Relationship Types

- Maximum Cardinality (Cardinality Ratios)
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N) ER diagram
- Minimum Cardinality (Participation Constraints and Existence Dependencies)
  - zero (optional participation, not existence-dependent)
  - one or more (mandatory, existence-dependent)

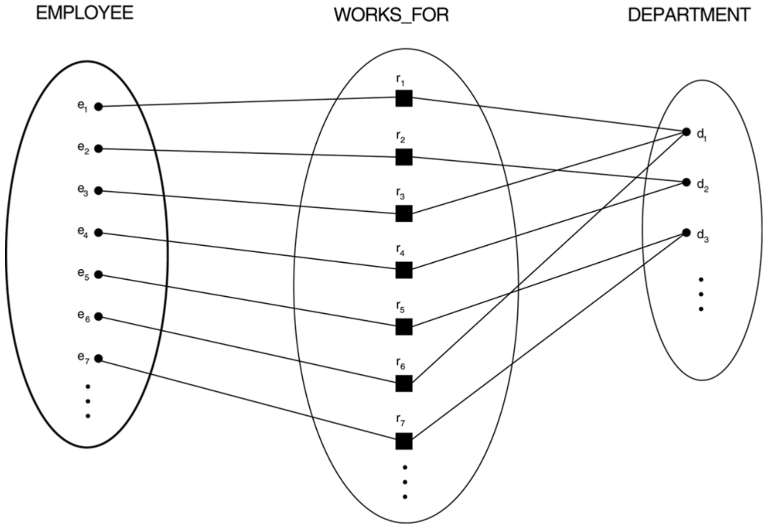
Constraints on Relationship Types

- Maximum Cardinality (Cardinality Ratios)
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N) ER diagram
- Minimum Cardinality (Participation Constraints and Existence Dependencies)
  - zero (optional participation, not existence-dependent)
  - one or more (mandatory, existence-dependent)

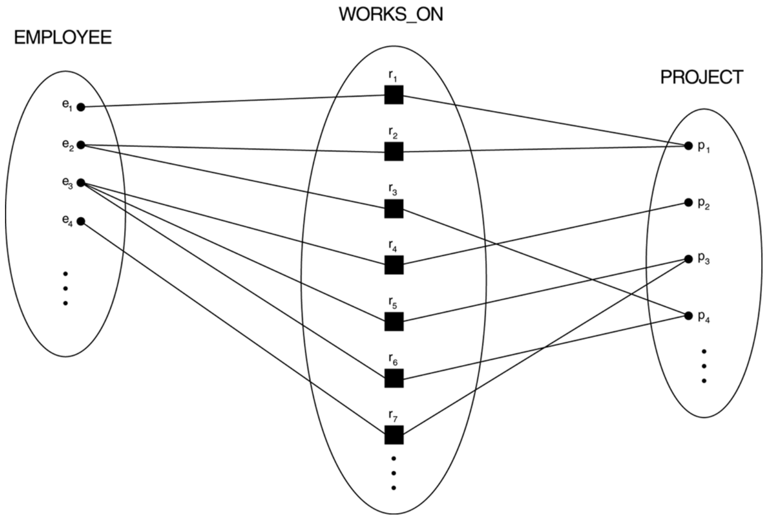
One-to-one (1:1) Relationship



Many-to-one (N:1) Relationship



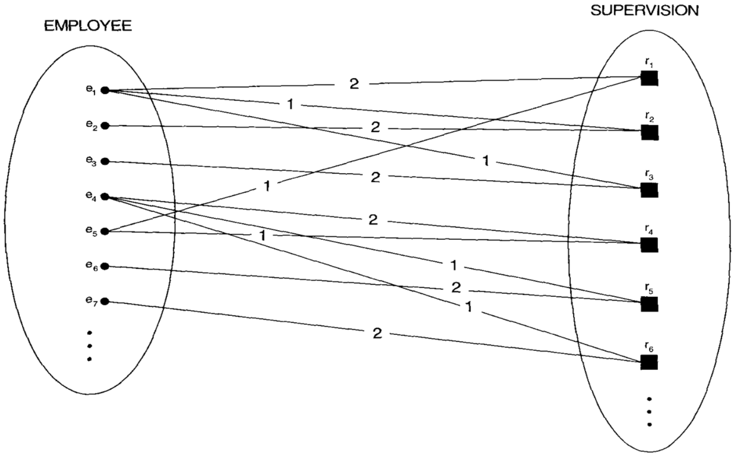
Many-to-many (M:N) Relationship



Recursive Relationship Type

- We can also have a recursive relationship type.
- Both participations are same entity type in different roles.
  - For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).
- In following figure, first role participation labeled with 1 and second role participation labeled with 2.
- In ER diagram, one needs to display role names to distinguish participations. ER diagram

A recursive relationship Supervision



Attributes of Relationship types

A relationship type can have attributes; for example, HoursPerWeek of WORKS_ON; its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.

Structural Constraints

- Structural constraints on relationships
  - Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
    SHOWN BY PLACING APPROPRIATE NUMBER ON THE LINK.
  - Participation constraint (on each participating entity type): total (called existence dependency) or partial.
    SHOWN BY DOUBLE LINING THE LINK

Relationships of Higher Degree

- Relationship types of degree 2 are called binary
- Relationship types of degree 3 are called ternary and of degree n are called n-ary
- In general, an n-ary relationship is not equivalent to n binary relationships

Guidelines for ER diagrams

- ER diagrams represent schemas rather than instances
- Use singular names and UPPERCASE letters for entity types and relationship types.
  e.g. `EMPLOYEE`, `DEPARTMENT`, `WORKS_FOR`, `MANAGES`
- Attributes names should be capitalized
  e.g. `Name`, `Salary`, `Bdate`
- Role names in lowercase letters e.g. `supervisor`
- Cardinality ratio of the relationship on each participation edge
  e.g. `1:1`, `1:N`, `N:1`, `M:N`
- Participation constraint represents using single line (partial) and double line (total)
- If the structural constraints cannot be determined from the requirements, user should make necessary assumptions.

Guidelines for ER diagrams

- ER diagrams represent schemas rather than instances
- Use singular names and UPPERCASE letters for entity types and relationship types.
  e.g. `EMPLOYEE, DEPARTMENT, WORKS_FOR, MANAGES`
- Attributes names should be capitalized
  e.g. `Name, Salary, Bdate`
- Role names in lowercase letters e.g. `supervisor`
- Cardinality ratio of the relationship on each participation edge
  e.g. `1:1, 1:N, N:1, M:N`
- Participation constraint represents using single line (partial) and double line (total)
- If the structural constraints cannot be determined from the requirements, user should make necessary assumptions.

Guidelines for ER diagrams

- ER diagrams represent schemas rather than instances
- Use singular names and UPPERCASE letters for entity types and relationship types.
  e.g. `EMPLOYEE, DEPARTMENT, WORKS_FOR, MANAGES`
- Attributes names should be capitalized
  e.g. `Name, Salary, Bdate`
- Role names in lowercase letters e.g. `supervisor`
- Cardinality ratio of the relationship on each participation edge
  e.g. `1:1, 1:N, N:1, M:N`
- Participation constraint represents using single line (partial) and double line (total)
- If the structural constraints cannot be determined from the requirements, user should make necessary assumptions.

Guidelines for ER diagrams

- ER diagrams represent schemas rather than instances
- Use singular names and UPPERCASE letters for entity types and relationship types.
  e.g. `EMPLOYEE, DEPARTMENT, WORKS_FOR, MANAGES`
- Attributes names should be capitalized
  e.g. `Name, Salary, Bdate`
- Role names in lowercase letters e.g. `supervisor`
- Cardinality ratio of the relationship on each participation edge
  e.g. `1:1, 1:N, N:1, M:N`
- Participation constraint represents using single line (partial) and double line (total)
- If the structural constraints cannot be determined from the requirements, user should make necessary assumptions.

Guidelines for ER diagrams

- ER diagrams represent schemas rather than instances
- Use singular names and UPPERCASE letters for entity types and relationship types.
  e.g. `EMPLOYEE, DEPARTMENT, WORKS_FOR, MANAGES`
- Attributes names should be capitalized
  e.g. `Name, Salary, Bdate`
- Role names in lowercase letters e.g. `supervisor`
- Cardinality ratio of the relationship on each participation edge
  e.g. `1:1, 1:N, N:1, M:N`
- Participation constraint represents using single line (partial) and double line (total)
- If the structural constraints cannot be determined from the requirements, user should make necessary assumptions.

Guidelines for ER diagrams

- ER diagrams represent schemas rather than instances
- Use singular names and UPPERCASE letters for entity types and relationship types.
  e.g. `EMPLOYEE, DEPARTMENT, WORKS_FOR, MANAGES`
- Attributes names should be capitalized
  e.g. `Name, Salary, Bdate`
- Role names in lowercase letters e.g. `supervisor`
- Cardinality ratio of the relationship on each participation edge
  e.g. `1:1, 1:N, N:1, M:N`
- Participation constraint represents using single line (partial) and double line (total)
- If the structural constraints cannot be determined from the requirements, user should make necessary assumptions.

Guidelines for ER diagrams

- ER diagrams represent schemas rather than instances
- Use singular names and UPPERCASE letters for entity types and relationship types.
  e.g. `EMPLOYEE, DEPARTMENT, WORKS_FOR, MANAGES`
- Attributes names should be capitalized
  e.g. `Name, Salary, Bdate`
- Role names in lowercase letters e.g. `supervisor`
- Cardinality ratio of the relationship on each participation edge
  e.g. `1:1, 1:N, N:1, M:N`
- Participation constraint represents using single line (partial) and double line (total)
- If the structural constraints cannot be determined from the requirements, user should make necessary assumptions.

Guidelines for ER diagrams

- ER diagrams represent schemas rather than instances
- Use singular names and UPPERCASE letters for entity types and relationship types.
  e.g. `EMPLOYEE, DEPARTMENT, WORKS_FOR, MANAGES`
- Attributes names should be capitalized
  e.g. `Name, Salary, Bdate`
- Role names in lowercase letters e.g. `supervisor`
- Cardinality ratio of the relationship on each participation edge
  e.g. `1:1, 1:N, N:1, M:N`
- Participation constraint represents using single line (partial) and double line (total)
- If the structural constraints cannot be determined from the requirements, user should make necessary assumptions.

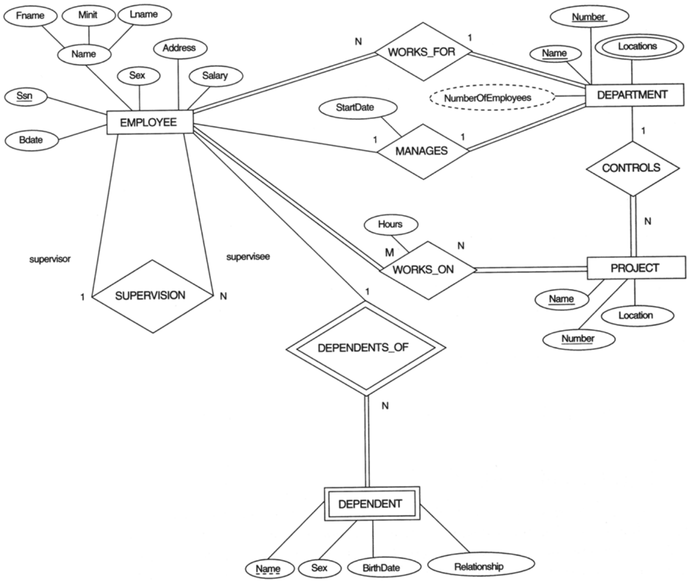
Exercise

Using COMPANY database specification, complete the ER diagram.

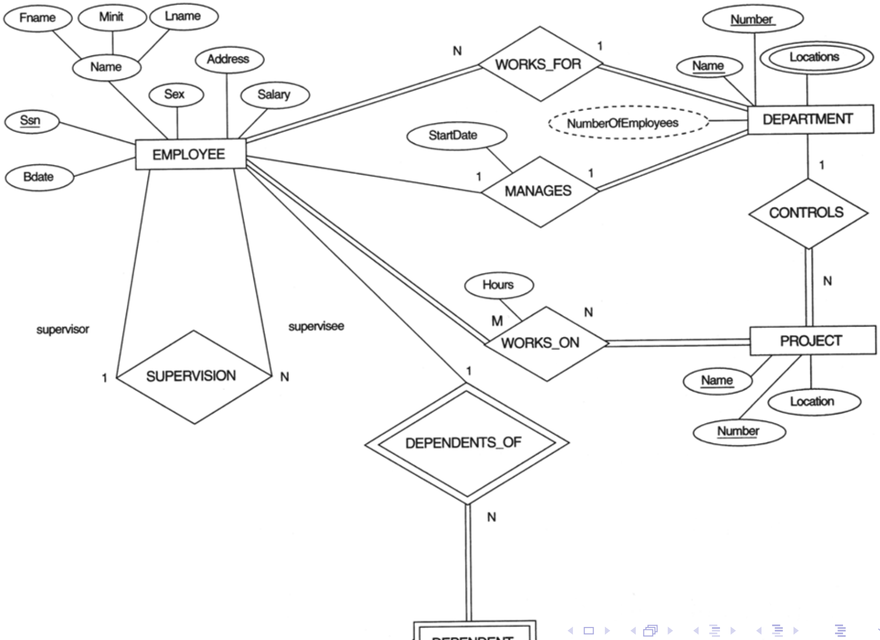
An Example Database Application: COMPANY

- Requirements for the COMPANY Database Back: Lec 5
- The company is organized into DEPARTMENTs. Each department has a unique name, a unique number, and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department controls a number of PROJECTs. Each project has a unique name, a unique number and is located at a single location.
- We store each EMPLOYEE's name, social security number, address, salary, sex, and birth date. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs. For each dependent, we keep their name, sex, birth date, and relationship to the employee.

An ER schema diagram for the COMPANY database



An ER schema diagram for the COMPANY database



Problems with the ER notation

- The entity relationship model in its original form did not support the specialization/ generalization abstractions
- Extended entity-relationship (EER) model
    - incorporates set-subset relationships
    - incorporates specialization/generalization hierarchies