
1 hour of study
6 hours of gaming

Databases - Tutorial 1
Conceptual Data Modeling

Hamza Salem - Innopolis University

Hamza Salem

Google

الكل | صور | فيديو | خرائط | الأخبار | المزيد | أدوات | المجموعات | البحث الآمن مفعّل

tweet | tweeted | linkedin | tweets | برمجة | twitter | حمزة السالم

enghamzasalem.com | حمزة سالم
enghamzasalem.com

Hamza Salem (@HamzatheH...
twitter.com

Hamza Salem - Researcher P...
jo.linkedin.com

Hamza Salem (@enghamzas...
twitter.com

Hamza SALEM | PhD Student ...
researchgate.net

Hamza Salem - YouTube
youtube.com

Who am i ?

- ❏ Phd Student
- ❏ Senior Lecturer
- ❏ Youtuber
- ❏ Udemy Instructor
- ❏ Blockchain Developer

Find me here Enghamzasalem.com

h.salem@innopolis.ru

Telegram @enghamzassalem

Requirements Engineering

- What it is Requirements Engineering
  - The process of **establishing the services that a customer requires** from a system and the constraints under which it operates and is developed.
  - The system requirements are the **descriptions of the system services and constraints** that are generated during the requirements engineering process.
- What is a requirement?
  - It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
  - **User requirements** - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
  - **System requirements** - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

System stakeholders

- Any person or organization who is **affected by the system** in some way and so who has a legitimate interest

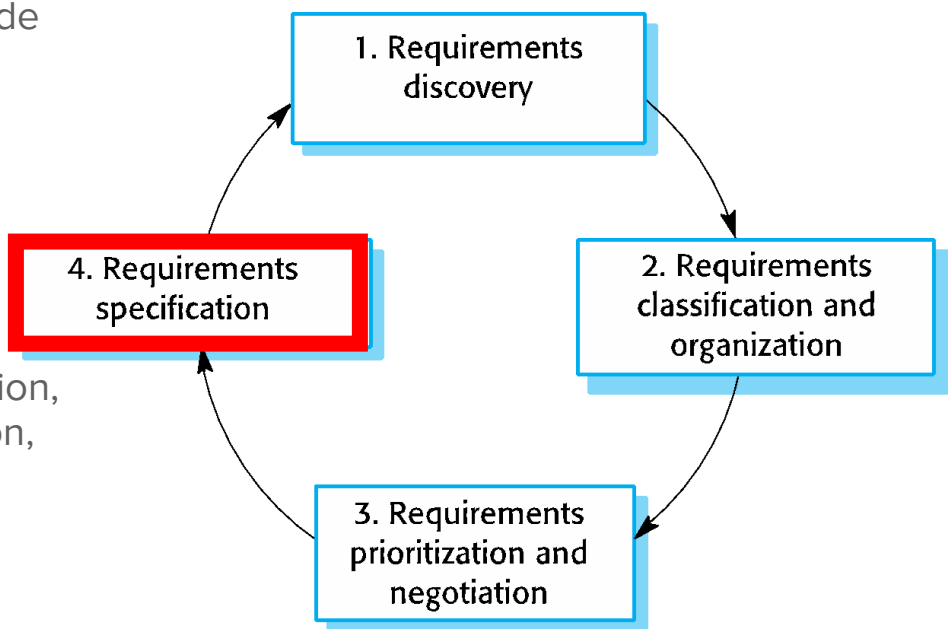
- Stakeholder **types**:
  - End users
  - System managers
  - System owners
  - External stakeholders

- Example - Stakeholders in a health care system
  - Patients whose information is recorded in the system.
  - Doctors who are responsible for assessing and treating patients.
  - Nurses who coordinate the consultations with doctors and administer some treatments.
  - IT staff who are responsible for installing and maintaining the system.

Requirements elicitation

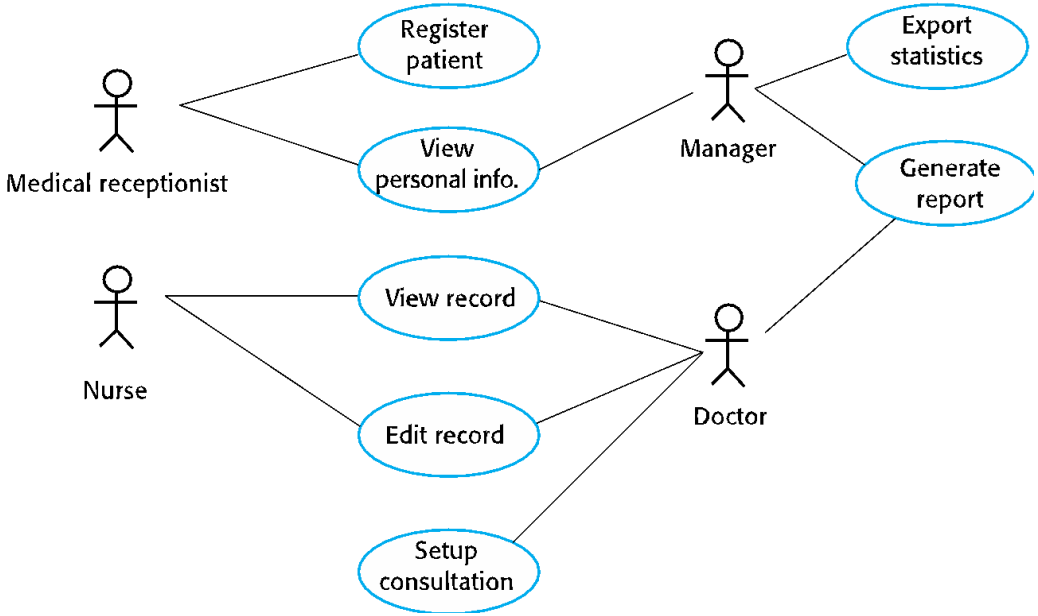
- Software engineers work with a range of system stakeholders to **find out** about:
  - The application domain
  - The services that the system should provide
  - The required system performance
  - Hardware constraints
  - Involved systems
  - etc.

- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.



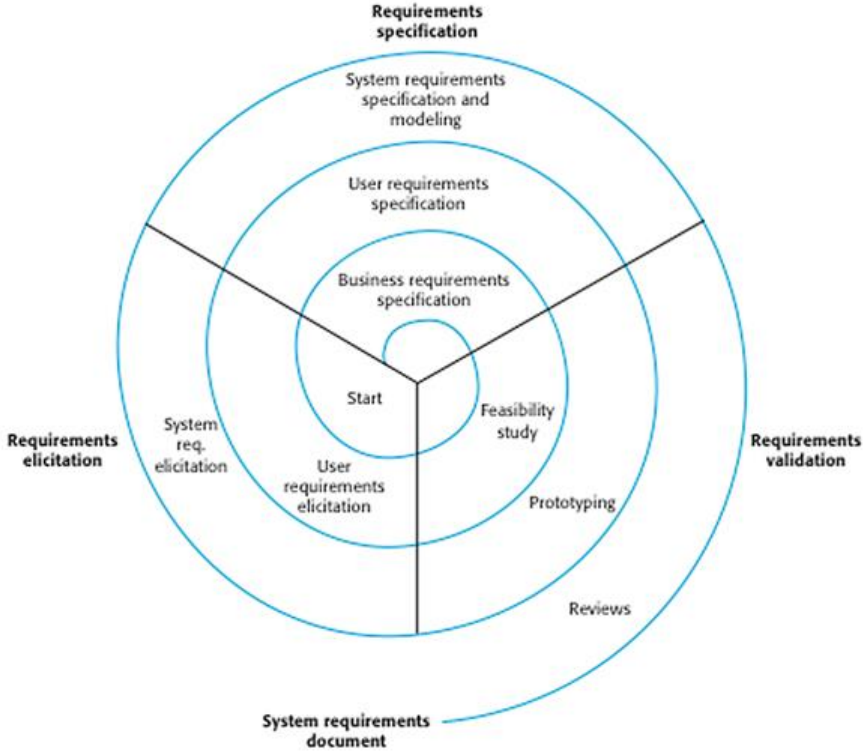
Use cases

- Use-cases are **used for requirement specification** and are included in the UML.
- Use cases identify the **actors** in an interaction and which describe the **interaction** itself.
- A set of use cases should describe all possible interactions with the system.
- UML sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.



Use case diagram for a health care system

The process of requirements engineering



Content

- Entity-Relationship (ER) Model
- High-Level Conceptual Data Models for Database Design
- ER model concepts
  - Entity Types, Entity Sets, Attributes, and Keys
  - Relationship Types, Relationship Sets, Roles, and Structural Constraints

Entity-Relationship (ER) Model

- The ER model is a popular high-level conceptual data model
  - It is frequently used for the conceptual design of database applications
  - Many database design tools still employ its concepts
- The diagrammatic notation associated with the ER model is known as ER diagram
  - Class diagrams, which are part of the Unified Modeling Language (UML), are becoming increasingly popular in both database and software design
  - Class diagrams are similar in many ways to the ER diagrams
- Next, the basic data-structuring concepts and constraints of the ER model and their use in the design of conceptual schemas

High-Level Conceptual Data Models for Database Design

A simplified overview of the database design process consists of:

1. Requirements collection and analysis
   - The database designers interview prospective database users to understand and document their data requirements.
   - The result of this step is a concisely written set of users' requirements.
   - These requirements should be specified in as detailed and complete a form as possible.
   - It is useful to specify the known functional requirements of the application. These consist of the user-defined operations (or transactions) that will be applied to the database, including both retrievals and updates. In software design, it is common to use data flow diagrams, sequence diagrams, scenarios, and other techniques to specify functional requirements.

High-Level Conceptual Data Models for Database Design

2. Conceptual design
   - Once the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using a high-level conceptual data model
   - The conceptual schema is a concise description of the data requirements of the users and includes detailed descriptions of the entity types, relationships, and constraints; these are expressed using the concepts provided by the high-level data model
   - They are usually easier to understand and can be used to communicate with nontechnical users
   - The high-level conceptual schema can also be used as a reference to ensure that all users' data requirements are met and that the requirements do not conflict.
   - This approach enables database designers to concentrate on specifying the properties of the data, without being concerned with storage and implementation details, which makes it is easier to create a good conceptual database design.

High-Level Conceptual Data Models for Database Design

3. Implementation of the database using a (commercial) DBMS
   - Most current commercial DBMSs use an implementation data model - such as the relational (SQL) model - so the conceptual schema is transformed from the high-level data model into the implementation data model.
   - This step is called logical design or data model mapping
   - The result of this phase is a database schema in the implementation data model of the DBMS. Data model mapping is often automated or semi automated within the database design tools.
4. Physical design phase
   - Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files are specified.
   - In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

High-Level Conceptual Data Models for Database Design

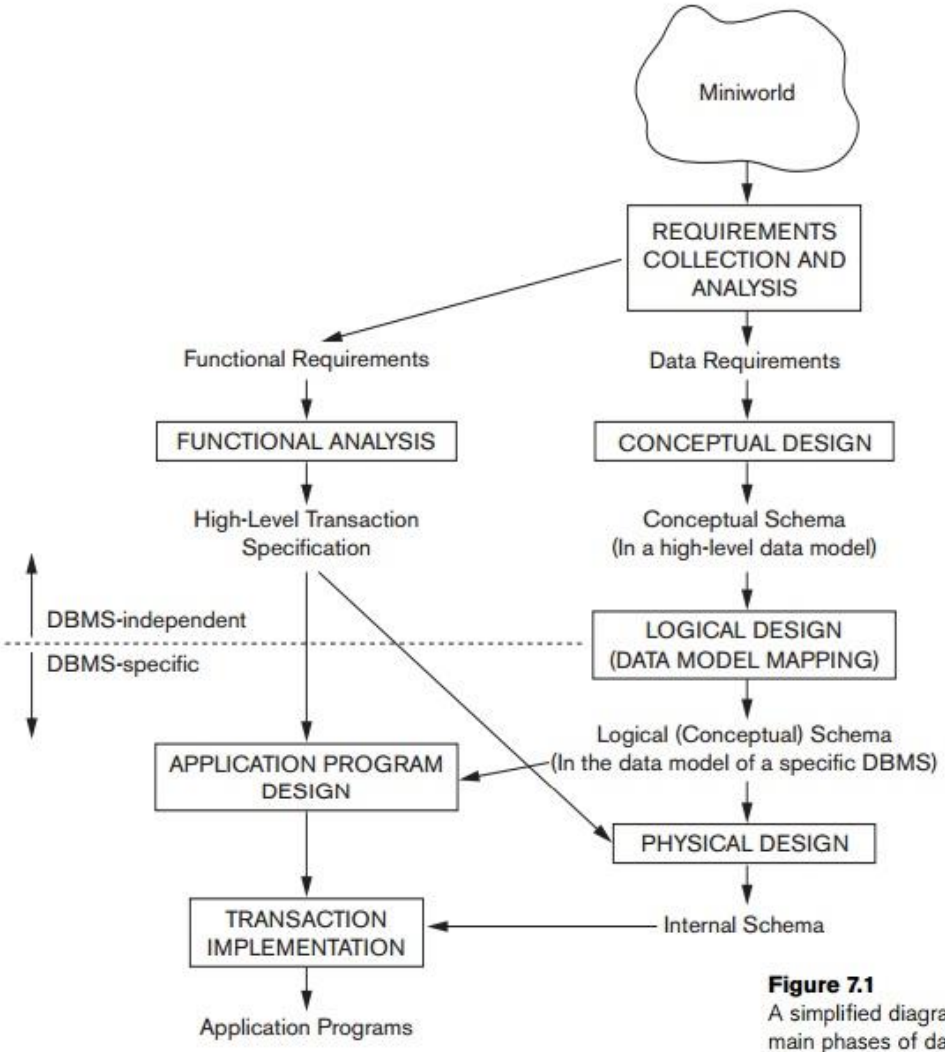


Figure 7.1
A simplified diagram to illustrate the main phases of database design.

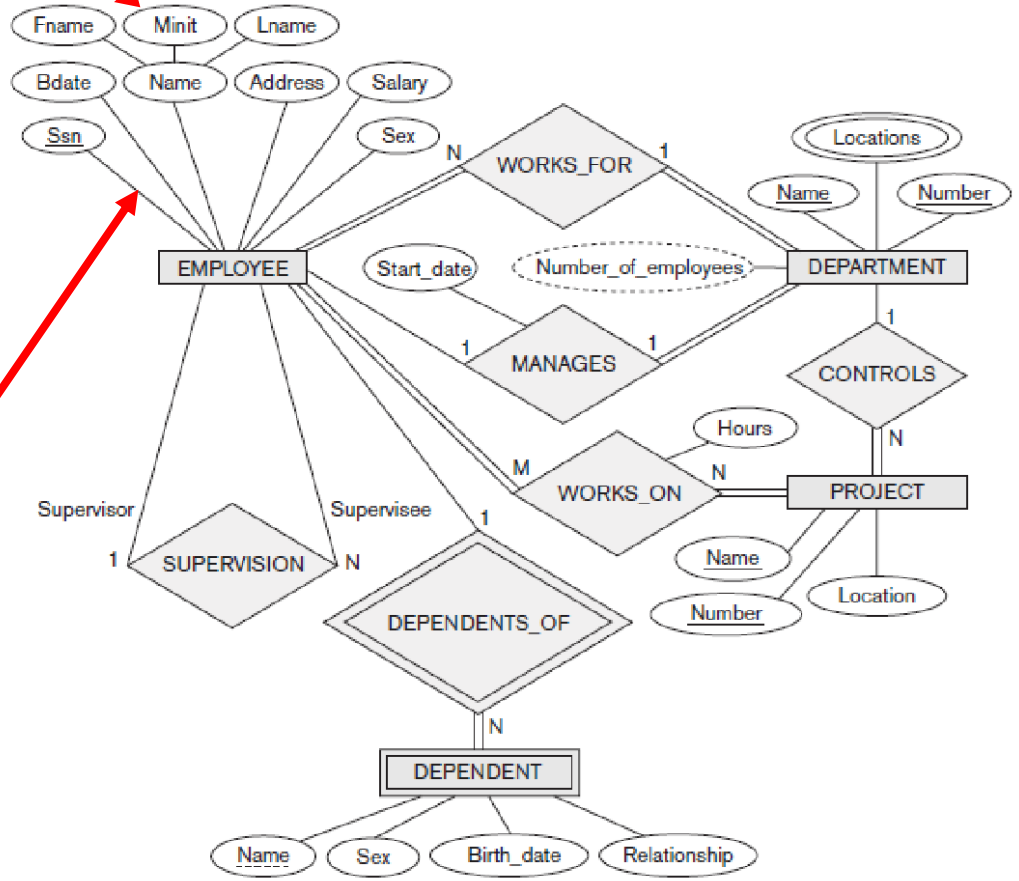
ER model concepts

- Let's describe a sample database application, called COMPANY, which serves to illustrate the basic ER model concepts and their use in schema design
- The COMPANY database keeps track of a company's employees, departments, and projects.
- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department.
- We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee's name, Social Security number, address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.

Entity Types, Entity Sets, Attributes, and Keys

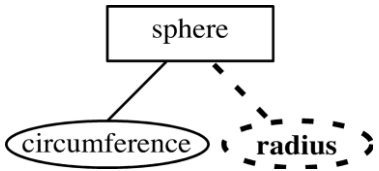
Entities and Attributes

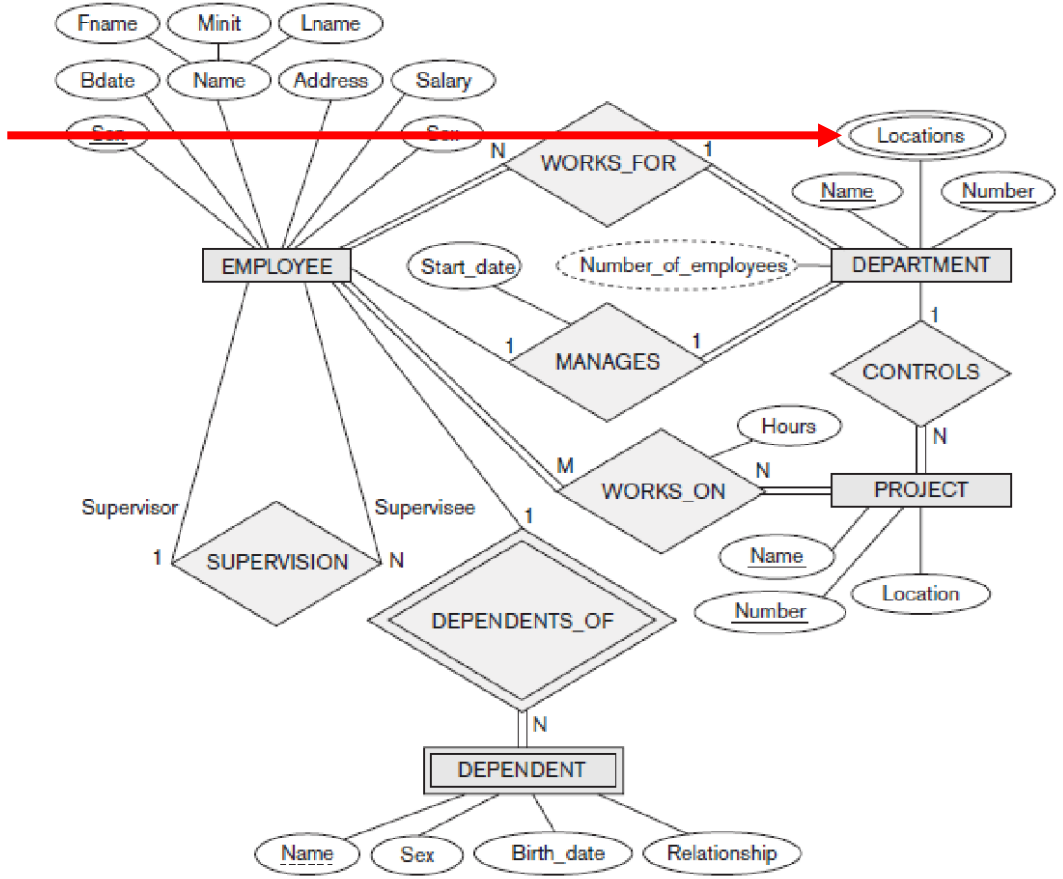
- **Entity**: a thing or object in the real world with an independent existence.
- An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course).
- Each entity has **attributes** - the particular properties that describe it. For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.
- A particular entity will have a value for each of its attributes. The attribute values that describe each entity become a major part of the data stored in the database.



Entity Types, Entity Sets, Attributes, and Keys

- Single-Valued versus Multivalued Attributes
  - Single-valued: a single value for a particular entity. For example, Age is a single-valued attribute of a person.
  - On the other hand, different people can have different numbers of values for the College_degrees attribute (more than one). Such attributes are called multivalued.
  - A multivalued attribute may have lower and upper bounds to constrain the number of values allowed for each individual entity.
- Stored versus Derived Attributes
  - When two (or more) attribute values are related.
  - For example, the Age (derived attribute) and Birth_date (stored attribute) attributes of a person.
  - Representation in the ERM:





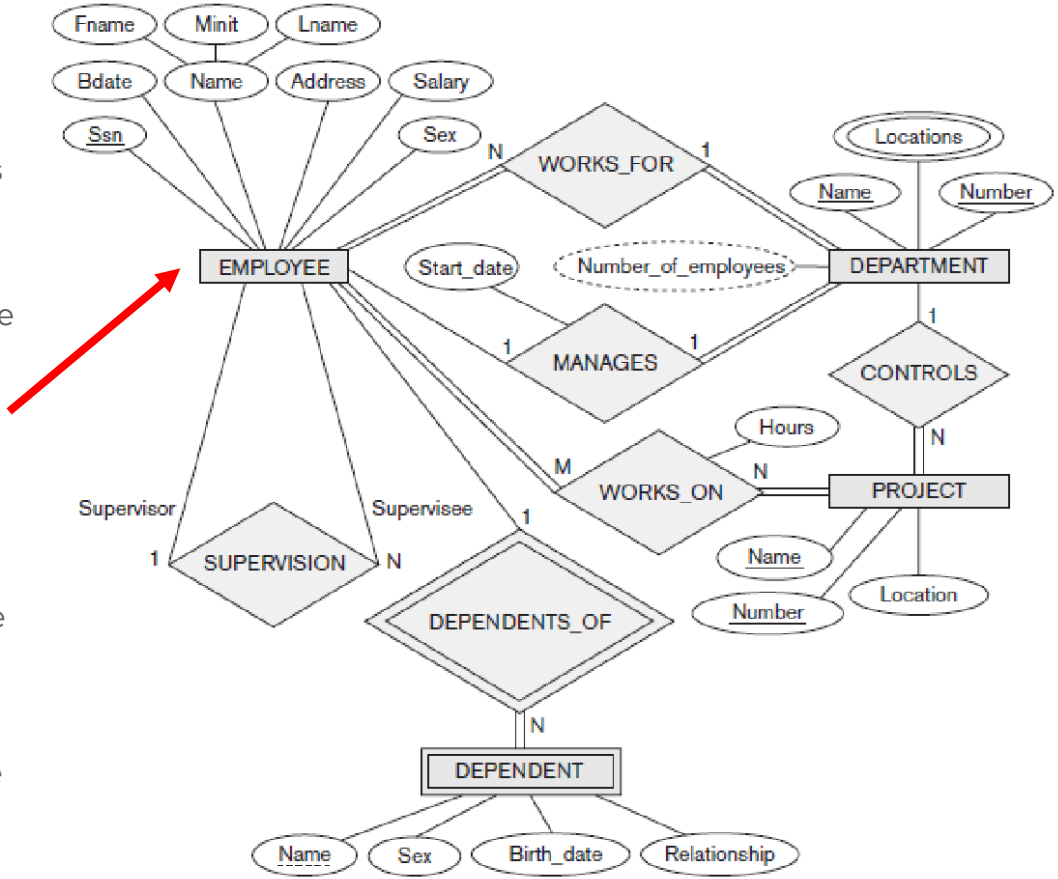
Entity Types, Entity Sets, Attributes, and Keys

Entity Types

- A database usually contains groups of entities that are similar.
- For example, a company employing hundreds of employees may want to store similar information concerning each of the employees. These employee entities share the same attributes, but each entity has its own value(s) for each attribute.
- An entity type defines a collection (or set) of entities that have the same attributes.
- Each entity type in the database is described by its name and attributes.

Entity Sets

- The collection of all entities of a particular entity type in the database at any point in time is called an entity set or entity collection;
- the entity set is usually referred to using the same name as the entity type, even though they are two separate concepts.
- For example, EMPLOYEE refers to both a type of entity as well as the current collection of all employee entities in the database.

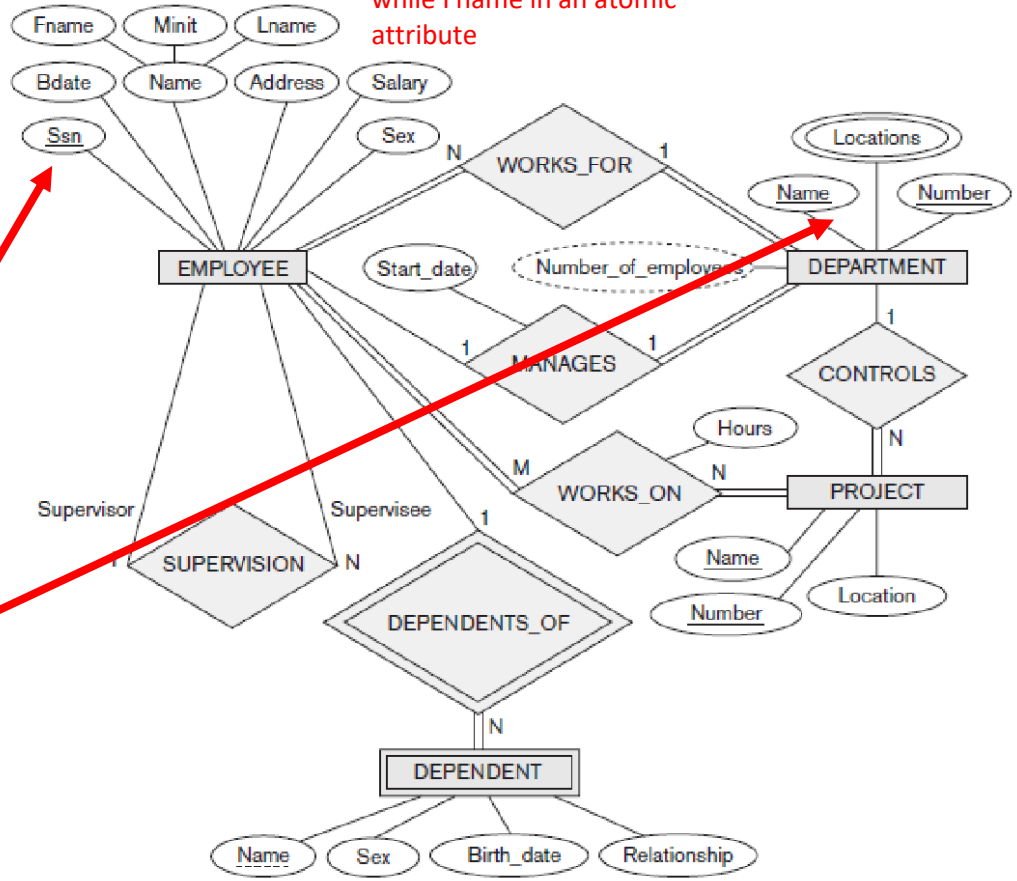


Entity Types, Entity Sets, Attributes, and Keys

Key Attributes of an Entity Type

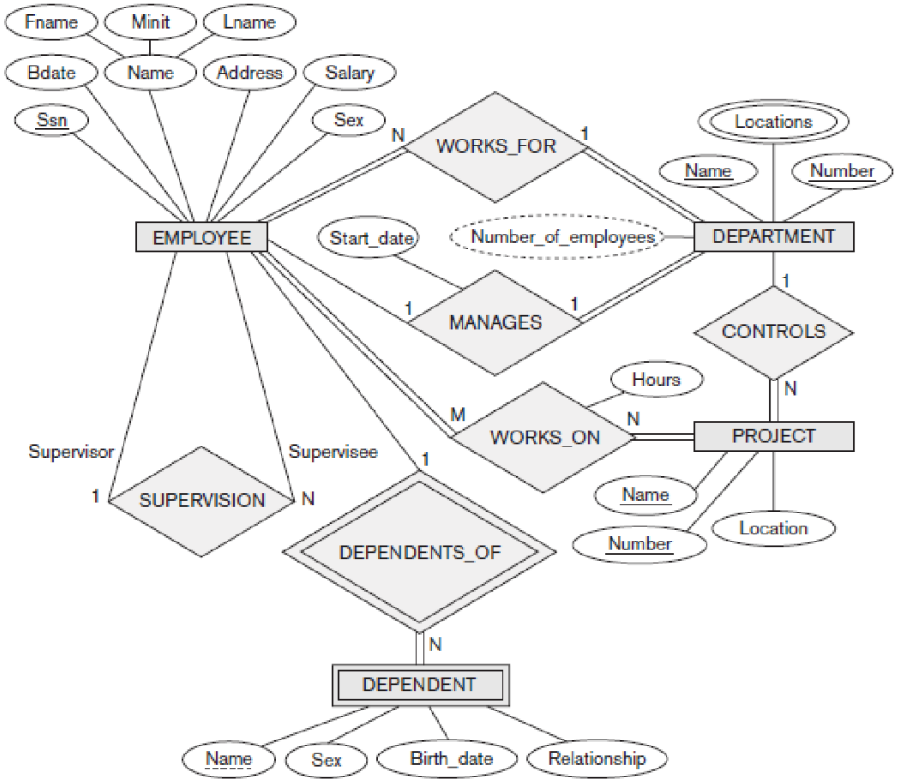
- An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes.
- An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set.
- The values of the attribute key can be used to identify each entity uniquely. For example, the Ssn attribute is a key of the EMPLOYEE entity type because no two employees are allowed to have the same Ssn.
- Composite key: Sometimes several attributes together form a key, meaning that the combination of the attribute values must be distinct for each entity. Superfluous attributes must not be included in a key.
- In ER diagrammatic notation, each key attribute has its name underlined inside the oval.

Name is a composite attribute, while Fname in an atomic attribute



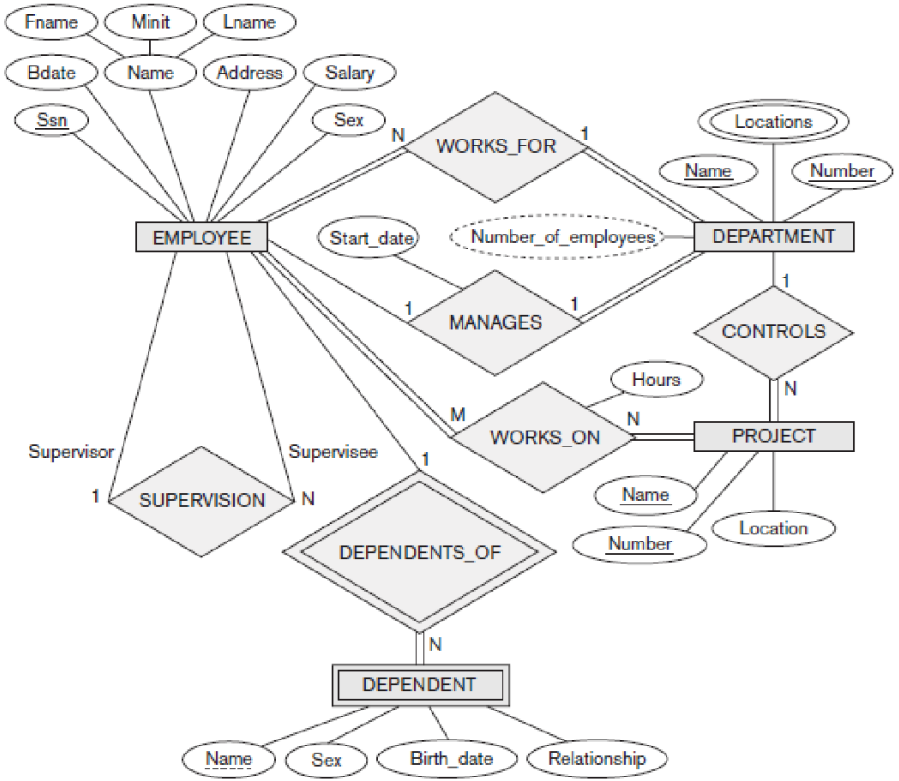
Relationship Types, Relationship Sets, Roles, and Structural Constraints

- Whenever an attribute of one entity type refers to another entity type, some relationship exists.
- Examples:
  - The attribute Manager of DEPARTMENT refers to an employee who manages the department
  - The attribute Controlling_department of PROJECT refers to the department that controls the project
  - The attribute Supervisor of EMPLOYEE refers to another employee (the one who supervises this employee)
  - the attribute Department of EMPLOYEE refers to the department for which the employee works
- In the ER model, these references should not be represented as attributes but as relationships



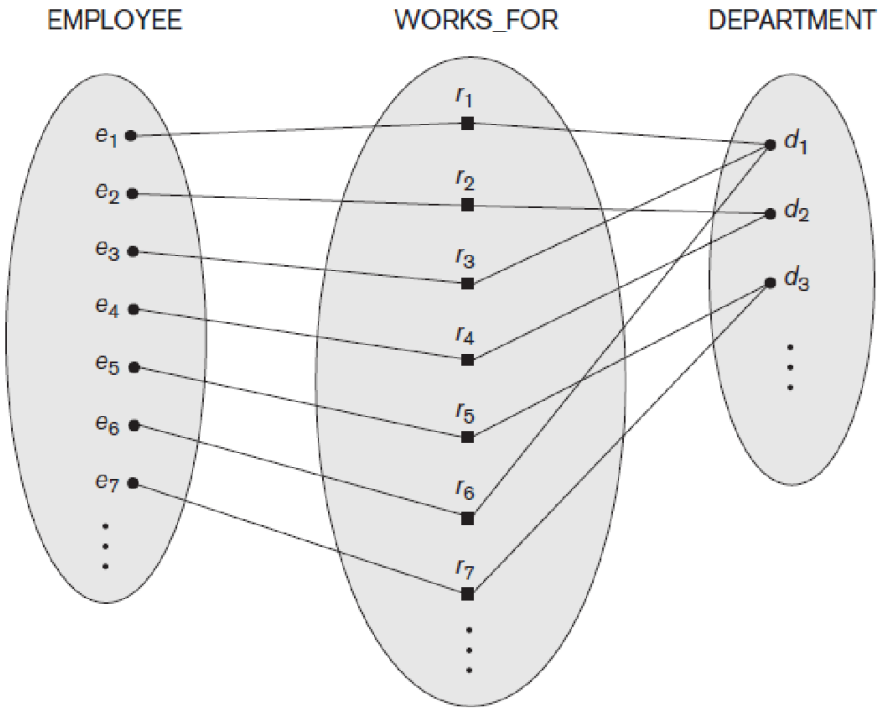
Relationship Types, Relationship Sets, Roles, and Structural Constraints

- Relationship Types/Sets/Instances
  - A relationship type R among n entity types $E_1, E_2, \ldots, E_n$ defines a set of associations - or a relationship set - among entities from these entity types.
  - A relationship type and its corresponding relationship set are customarily referred to by the same name, R.
  - Mathematically, the relationship set R is a set of relationship instances r_i, where each r_i associates n individual entities $(e_1, e_2, \ldots, e_n)$, and each entity e_j in r_i is a member of entity set E_j, $1 \le j \le n$.
  - Hence, a relationship set is a mathematical relation on $E_1, E_2, \ldots, E_n$; alternatively, it can be defined as a subset of the Cartesian product of the entity sets $E_1 \times E_2 \times \ldots \times E_n$.
  - Each of the entity types $E_1, E_2, \ldots, E_n$ is said to participate in the relationship type R
  - Similarly, each of the individual entities $e_1, e_2, \ldots, e_n$ is said to participate in the relationship instance $r_i = (e_1, e_2, \ldots, e_n)$.



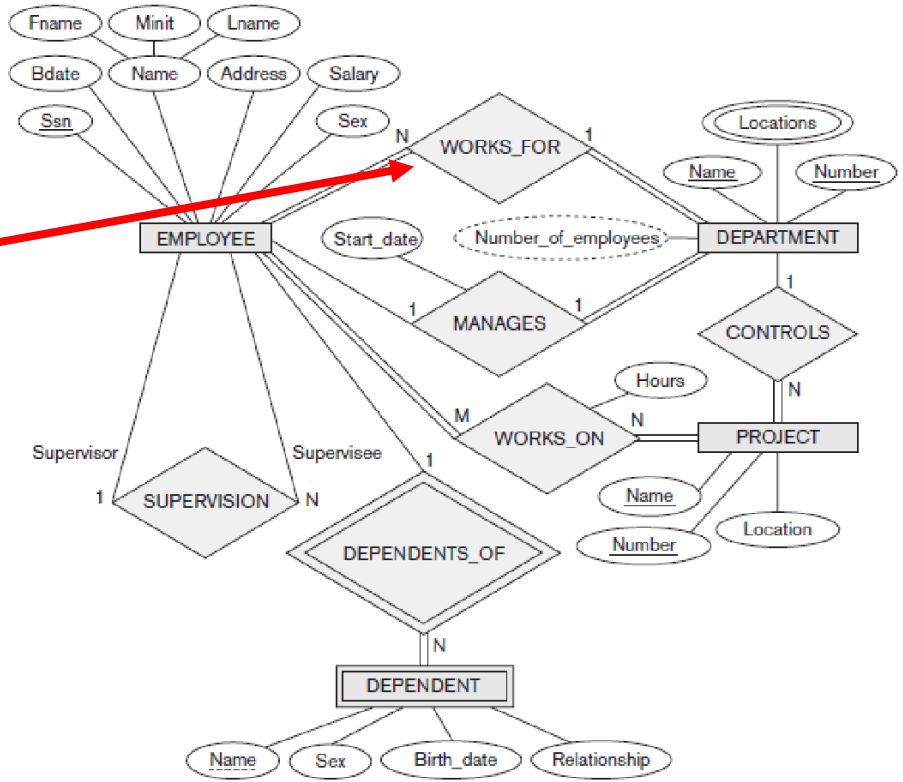
Relationship Types, Relationship Sets, Roles, and Structural Constraints

- For example, consider a relationship type WORKS_FOR between the two entity types EMPLOYEE and DEPARTMENT, which associates each employee with the department for which the employee works.
- Each relationship instance in the relationship set WORKS_FOR associates one EMPLOYEE entity and one DEPARTMENT entity.
- The figure illustrates this example, where each relationship instance r_i is shown connected to the EMPLOYEE and DEPARTMENT entities that participate in r_i.
- In the miniworld represented in the figure, the employees e_1, e_3, and e_6 work for department d_1; the employees e_2 and e_4 work for department d_2; and the employees e_5 and e_7 work for department d_3.



Relationship Types, Relationship Sets, Roles, and Structural Constraints

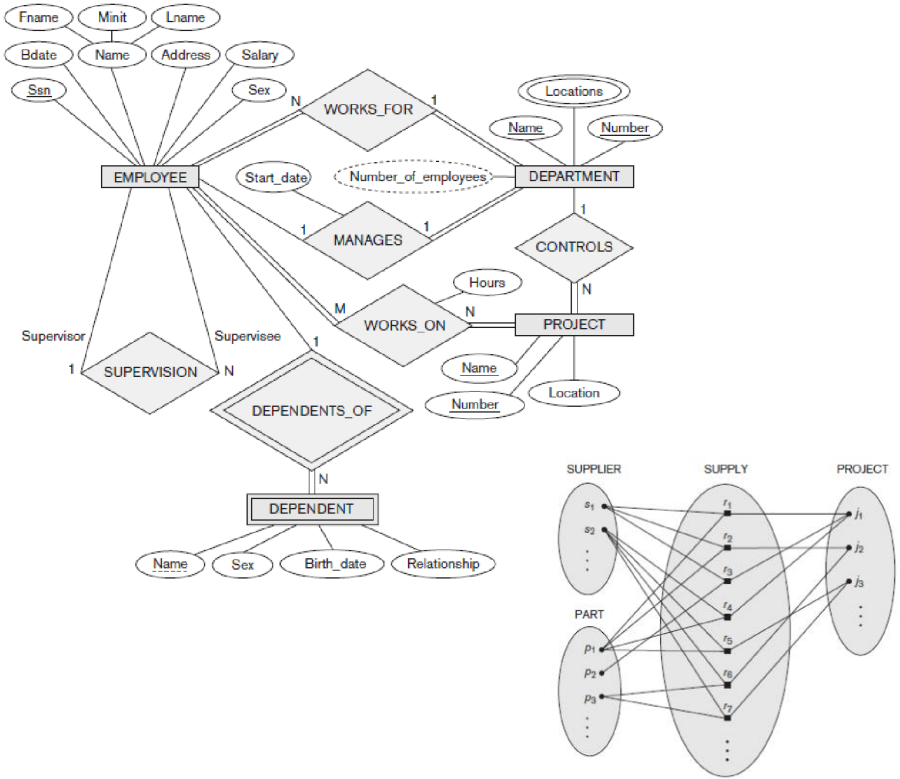
- In ER diagrams, relationship types are displayed as diamond-shaped boxes, which are connected by straight lines to the rectangular boxes representing the participating entity types.
- The relationship name is displayed in the diamond-shaped box (see figure).



Relationship Types, Relationship Sets, Roles, and Structural Constraints

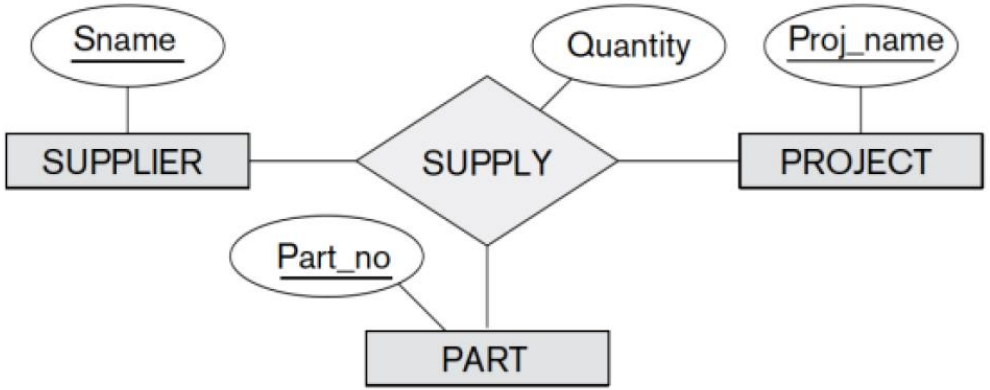
Degree of a Relationship Type

- The number of participating entity types. Hence, the WORKS_FOR relationship is of degree two.
- A relationship type of degree two is called binary, and one of degree three is called ternary.
- Relationships can generally be of any degree, but the ones most common are binary relationships.
- Higher-degree relationships are generally more complex than binary relationships.



Ternary Relationship

- An example of a ternary relationship is SUPPLY, where each relationship instance r_i associates three entities - a supplier s, a part p, and a projec – “whenever s supplies part p tc project j”. Can you think of another example?

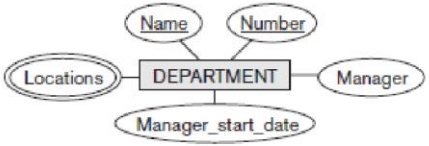


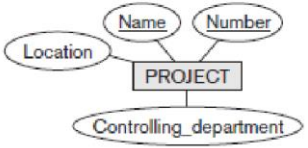
Relationship Types, Relationship Sets, Roles, and Structural Constraints

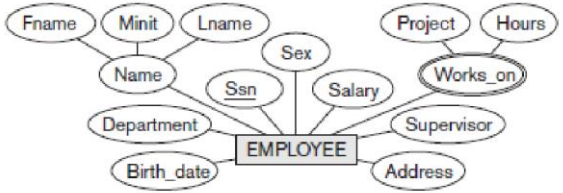
Relationships as Attributes

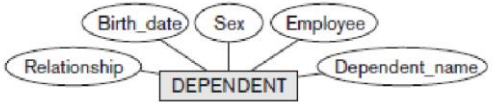
- It is sometimes convenient to think of a binary relationship type in terms of attributes.
- Consider the WORKS_FOR relationship type. One can think of an attribute called Department of the EMPLOYEE entity type, where the value of Department for each EMPLOYEE entity is (a reference to) the DEPARTMENT entity for which that employee works.
- Hence, the value set for this Department attribute is the set of all DEPARTMENT entities, which is the DEPARTMENT entity set.
- This is what we did in this figure when we specified the initial design of the entity type EMPLOYEE for the COMPANY database.
- However, when we think of a binary relationship as an attribute, we always have two options or two points of view. In this example, the alternative point of view is to think of a multivalued attribute Employees of the entity type DEPARTMENT whose value for each DEPARTMENT entity is the set of EMPLOYEE entities who work for that department. The value set of this Employees attribute is the power set of the EMPLOYEE entity set. Either of these two attributes—Department of EMPLOYEE or Employees of DEPARTMENT—can represent the WORKS_FOR relationship type.

Preliminary design





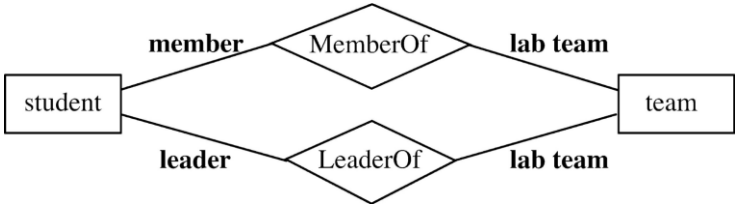




Relationship Types, Relationship Sets, Roles, and Structural Constraints

Role Names

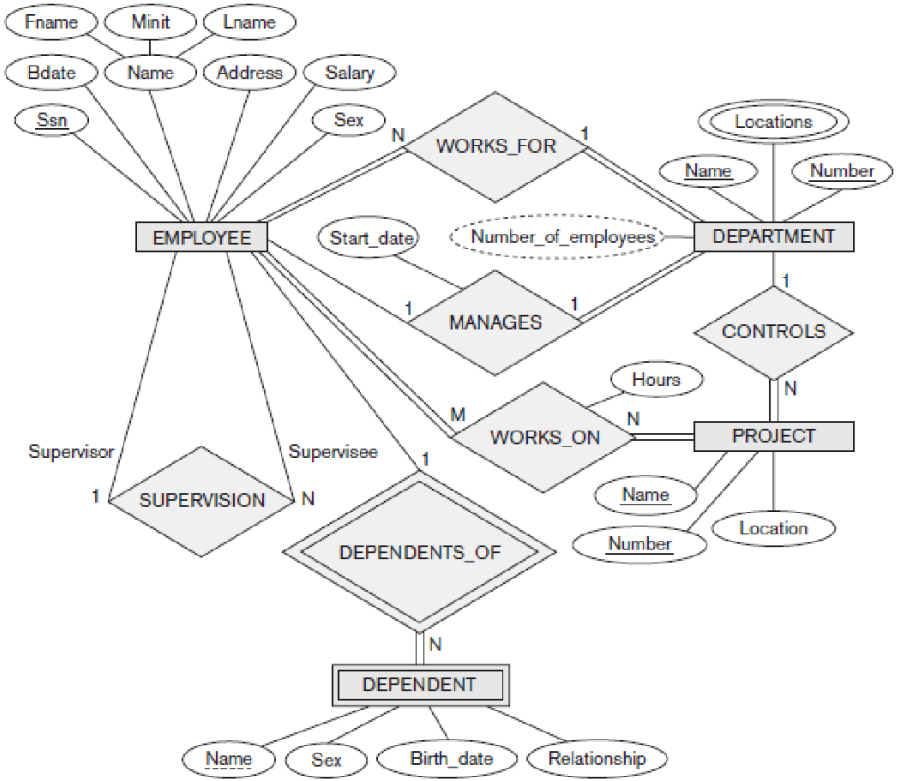
- Each entity type that participates in a relationship type plays a particular role in the relationship.
- The role name signifies the role that a participating entity from the entity type plays in each relationship instance, and it helps to explain what the relationship means.
- For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer.
- Role names are **not technically necessary in relationship types where all the participating entity types are distinct, since each participating entity type name can be used as the role name**. However, in some cases the same entity type participates more than once in a relationship type in different roles.
- **In such cases the role name becomes essential for distinguishing the meaning of the role that each participating entity plays.**



Relationship Types, Relationship Sets, Roles, and Structural Constraints

Recursive Relationships or self-referencing relationships

- The SUPERVISION relationship type relates an employee to a supervisor, where both employee and supervisor entities are members of the same EMPLOYEE entity set.
- Hence, the EMPLOYEE entity type participates twice in SUPERVISION: once in the role of supervisor (or boss), and once in the role of supervisee (or subordinate).
- Each relationship instance r_i in SUPERVISION associates two different employee entities e_j and e_k, one of which plays the role of supervisor and the other the role of supervisee.



Reference

- https://wofford-ecs.org/DataAndVisualization/ermodel/material.htm