Lecture 2: Conceptual Data Modeling Using the Entity-Relationship (ER) Model
"Mapping a Conceptual Design into a Logical Design"
Readings: Chapter 7.

1 Overview:

Conceptual modelling is a very important phase in designing a successful database application.
In this chapter we present the modelling concepts of the Entity-Relationship (ER) model, which is a popular high-level conceptual data model.

2 A simplified Description of the Database Design Process

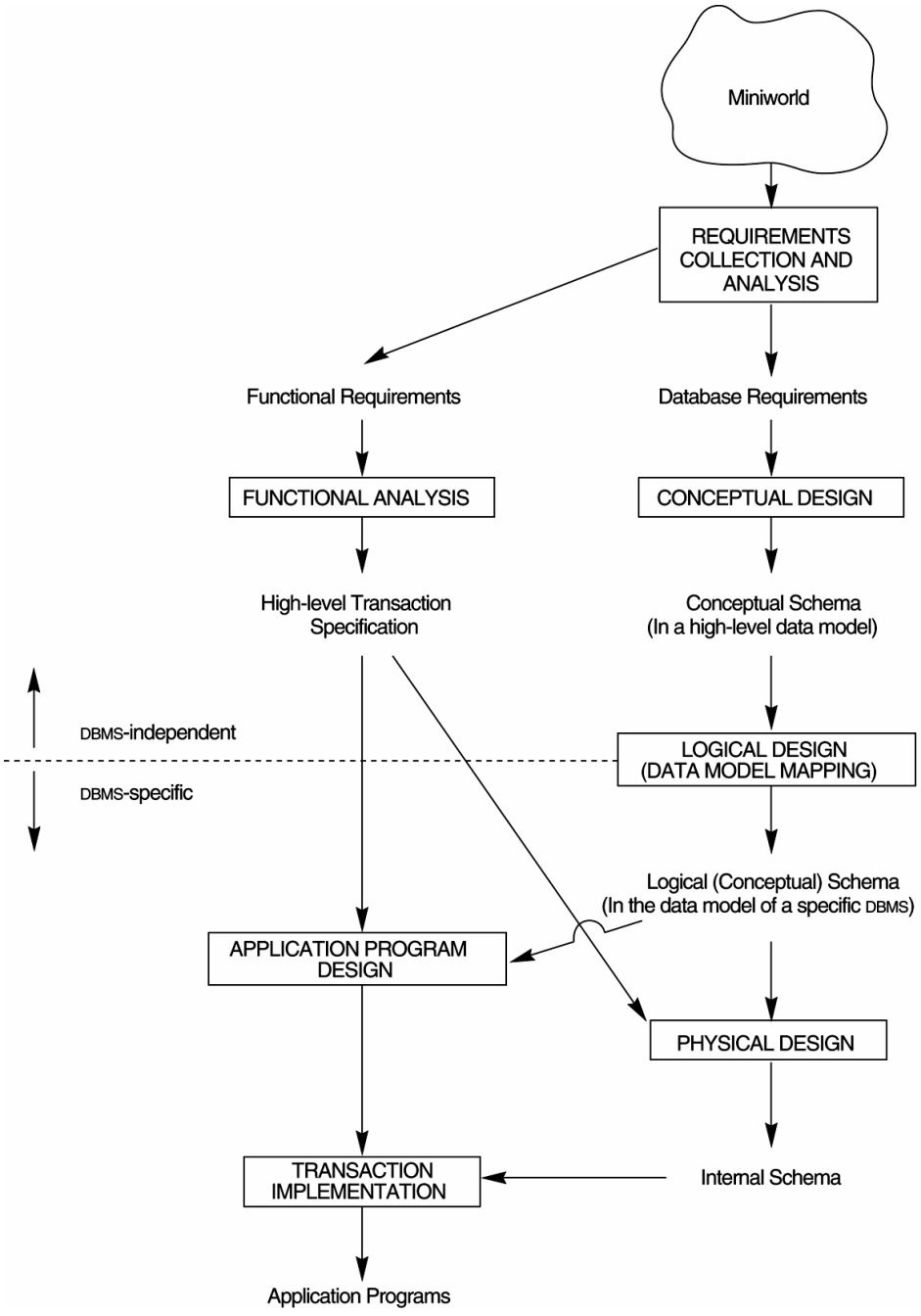


FIGURE 1 a simplified diagram to illustrate the main phases of database design

Step 1: Requirements collection and analysis

During this step, the system analyst interview prospective database users to understand and document their data requirements.

In parallel it is useful to specify the functional requirements of the application. These consist of the user-defined operations (or transactions) that will be applied to the database.

Step 2: Conceptual design

Once all requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using high level 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.

Step 3: Logical design

The next step in database design is the actual implementation of the database, using a commercial DBMS. Most current commercial DBMSs use an implementation data model such as the relational or the object-relational database model- so the conceptual schema is transformed from the high-level data model into the implementation data model.

Step 4: Physical design

The last step is the physical design phase, during which the internal storage structures, indexes, access paths, and file organizations for the database files are specified.

3 Summary of Notations for ER Diagrams

The ER model describes data as entities, relationships, and attributes

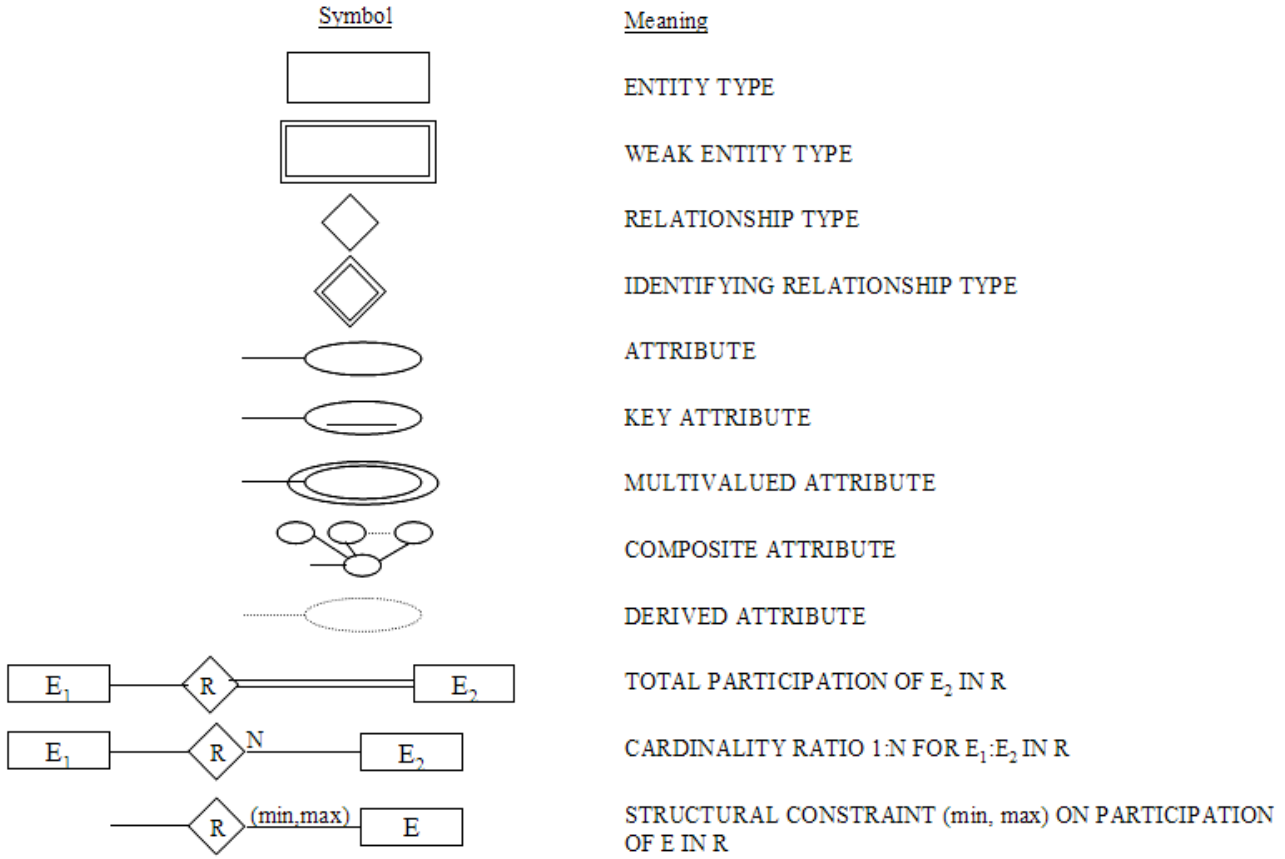


FIGURE 2 Summary of the notation for ER diagrams

4 An Example Database Application

Example COMPANY Database

Requirements of the Company (oversimplified for illustrative purposes)

1) 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 when that employee began managing the department.
A department may have several locations.

2) Each department *controls* a number of PROJECTs. Each project has a unique name, a unique number and is located at a single location.

3) 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 works on each project. We also keep track of the *direct supervisor* of each employee.

4) Each employee may *have* a number of DEPENDENTs. We keep each dependent's name, sex, birth date, and relationship to the employee.

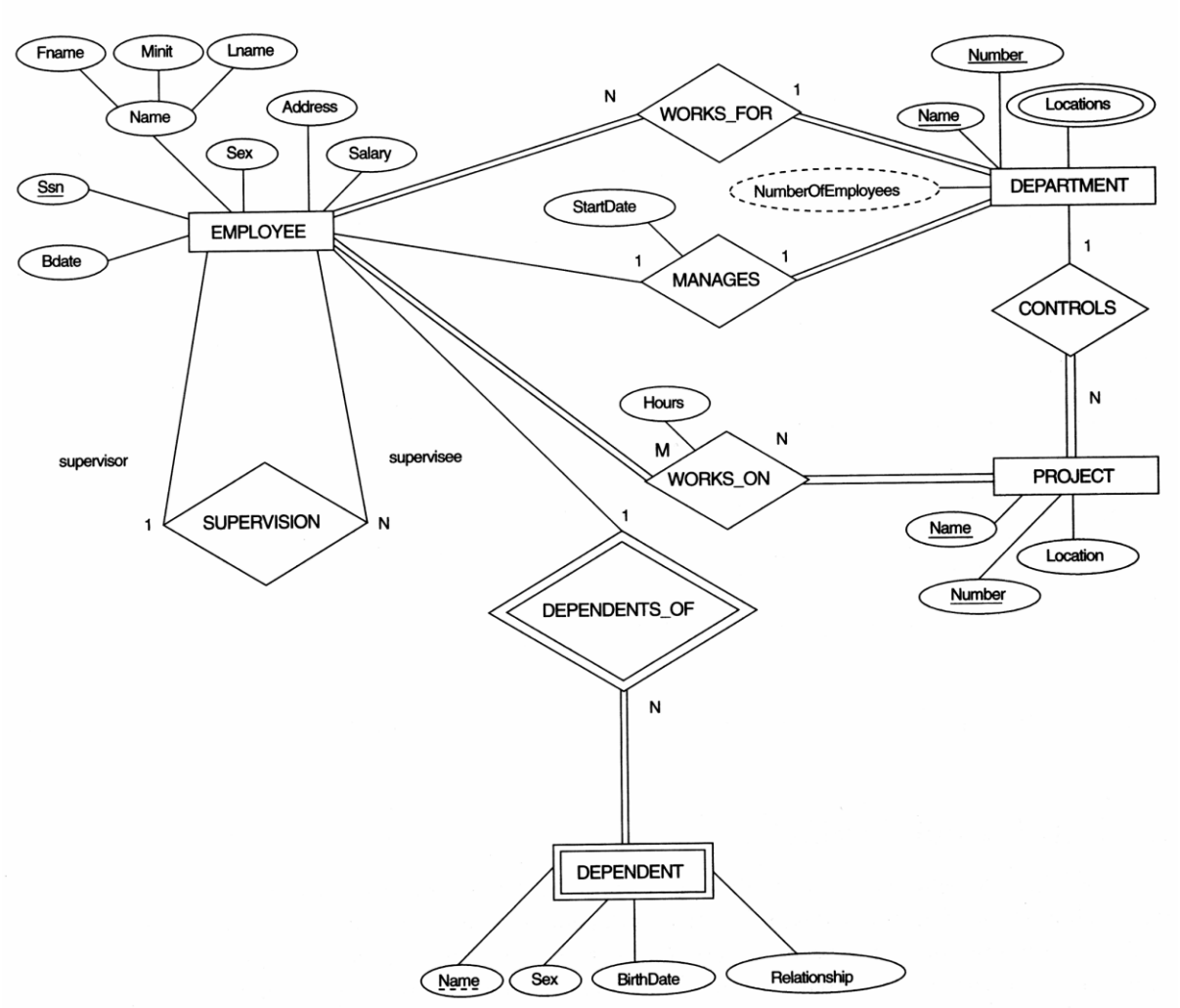


FIGURE 3 An ER schema diagram for the COMPANY database.

5 ER Model Concepts (Entity Types, Entity sets, Attributes, and Keys)

5.1 Entities and Attributes

- **Entities are specific objects or things in the mini-world that are represented in the database. An entity may be an object with physical existence (car, person,..) or with conceptual existence (course, job,…).**
- **An entity type is a collection of entities that have the same attributes. An entity type is represented in ER diagrams as a rectangular box enclosing the entity type.**
- **Attributes are properties used to describe an entity. For example an EMPLOYEE entity may have a Name, SSN, Address, Sex, Birth Date. Attribute names are enclosed in ovals.**
- **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', BirthDate='09-JAN-55'**
- **Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, sub range, enumerated type, …**

Types of Attributes (1)

- **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.

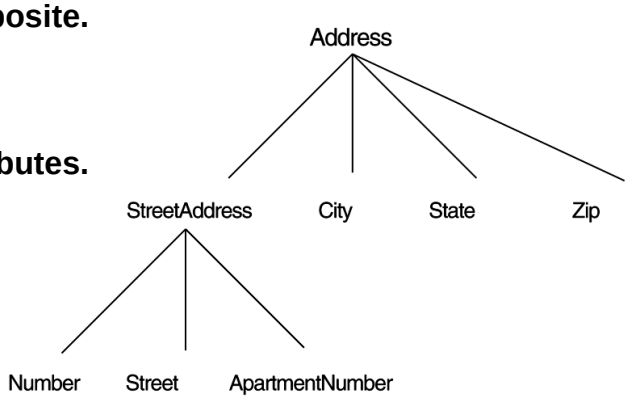


FIGURE 4 A hierarchy of composite attributes.

- **Multi-valued**
  - An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.
  - Multi-valued attributes are displayed in double ovals.

Types of Attributes (2)

- **In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.**

We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multivalued attributes between braces {}.

{AddressPhone({Phone(AreaCode,PhoneNumber)},
Address(StreetAddress(Number,Street,ApartmentNumber),
City,State,Zip)) }

FIGURE 5 A complex attribute: AddressPhone.

Types of Attributes (3)

- **Stored versus Derived attributes:**
  In some cases, two or more attribute value are related- for example, the Age and BirthDate attributes of a person. For a particular person entity, the value of Age can be determined from the current date and the value of that person's BirthDate. The Age attribute is hence called a derived attribute and is said to be derivable from the BirthDate attribute, which is called a stored attribute. Some attribute values can be derived from related entities; for example, an attribute NumberOfEmployees of a department entity can be derived by counting the number of employees related to (working for) that department.
  ******A derived attribute is represented by dashed oval.**
- **Null values:**
  In some cases a particular entity may not have an applicable value for an attribute. For example, the AppartmentNumber attribute of an address applies only to address that are in apartment buildings and not to other types of residences, such as single-family homes.

5.2 Entity Types, Entity Sets, Keys, and Value Sets

Entity Types

- **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 entity type defines a collection of entities that have the same attributes. The collection of all entities of a particular entity type in the database at any point of time is called an entity set.

Key Attributes

- **An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes.**
- **An entity type usually has an attribute whose values are distinct for each individual entity in the entity set. Such an attribute is called a key attribute, and its values can be used to identify each entity uniquely. A key attribute has its name underlined inside the oval.**
- **An attribute of an entity type for which each entity must have a unique value is called a key attribute 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).**

Value sets(Domains) of attributes

Each single attribute of an entity type is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity. Value sets are not displayed in ER diagrams.

5.3 Relationship Types, Sets, and Instances

Relationships and Relationship Types (1)

- **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.**
- **Relationship types are displayed as diamond-shaped.**

Relationships and Relationship Types (2)

- **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.**

ER DIAGRAM – Relationship Types are:
WORKS_FOR, MANAGES, WORKS_ON, CONTROLS,
SUPERVISION, DEPENDENTS_OF

The degree of relationship type

- **The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships. 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.**

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 helps to explain what the relationship means.**
  For example, in 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 participating entity types are distinct, since each participating entity type name can be used as a role name.**

Recursive Relationship

- **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 each participation. Such relationship types are called recursive relationships.**
- **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 ER diagram, need to display role names to distinguish participations.**

A RECURSIVE RELATIONSHIP SUPERVISION
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

5.4 Constraints on Relationship Types

Relationship types have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set.
We can distinguish two main types of relationship constraints: cardinality ratio and participation.

Cardinality ratios for binary relationships

Cardinality ratio specifies the maximum number of relationship instances that an entity can participate in.
(Also known as ratio constraints)
The possible cardinality ratios for binary relationship type are:

- **One-to-one (1:1)**

An example of a 1:1 binary relationship is MANAGES, which relates a department entity to the employee who manages that department. This represents the miniworld constraints that- at any point in time-an employee can manage only one department and a department has only one manager.

- **One-to-many (1:N) or Many-to-one (N:1)**

An example of a 1: N binary relationship is WORKS_FOR, which relates a department entity to the employees who works for that department.
The ratio 1:N means that each department can be related to (that is, employs) any number of employees, but an employee can be related to (work for) only one department.

- **Many-to-many(M:N**

The relationship type WORKS_ON is of cardinality ratio M:N because the miniworld rule is that an employee can work on several projects and a project can have several employees

*****Cardinality ratios are represented by displaying 1, M, N on diamonds**

Participation constraints or existence dependency constraints

The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
It specifies the minimum number of relationship instances that each entity can participate in.
There are two types of participation constraints:

- **Total participation (mandatory, existence-dependent):**
  (one or more)

An example of total participation, if a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS_FOR relationship instance.

- **Partial participation (optional participation, not existence-dependent) zero**

In figure 3 we do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is partial,, meaning that some or part of the set of employee entities are related to some department entity via MANAGES.

******In ER diagrams, total participation is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line.**

5.5 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.**
  Another example is to include the date on which a manager started managing a department via an attribute StartDate for the MANAGES relationship type

Attribute of a Relationship Type is: Hours of WORKS_ON

5.6 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 Type is: DEPENDENT
Identifying Relationship is: DEPENDENTS_OF

A weak entity type normally has a partial key, which is the set of attributes that can uniquely identify weak entities that are related to the same owner (sometimes called the discriminator).

******In ER diagram, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines. The partial key attribute is underlined with a dashed or dotted line**

5.7 Alternative (min, max) notation for relationship structural constraints:

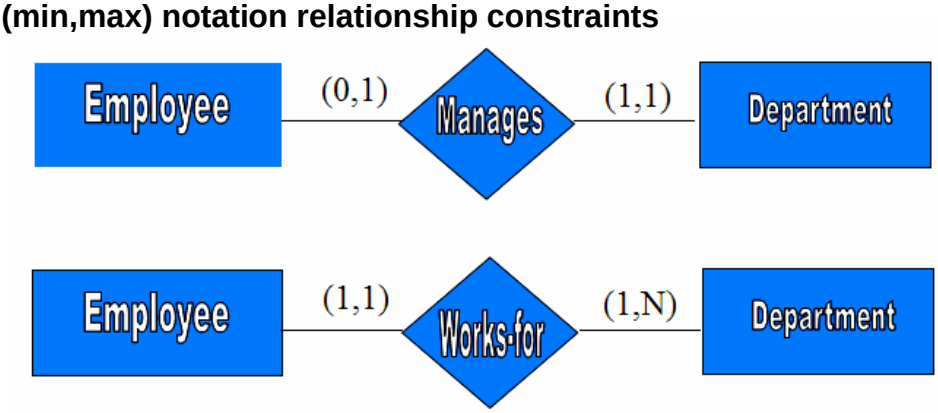
This notation involves associating a pair of integer numbers (min, max) with each participation of an entity type E in relationship type R where
$0 \leq min \leq max$ and $max \geq 1$. The numbers mean that for each entity e in E, e must participate in at least min and at most max relationship instances in R at any point in time.
In this method , min=0 implies partial participation, whereas min >0 implies total participation.
Examples:

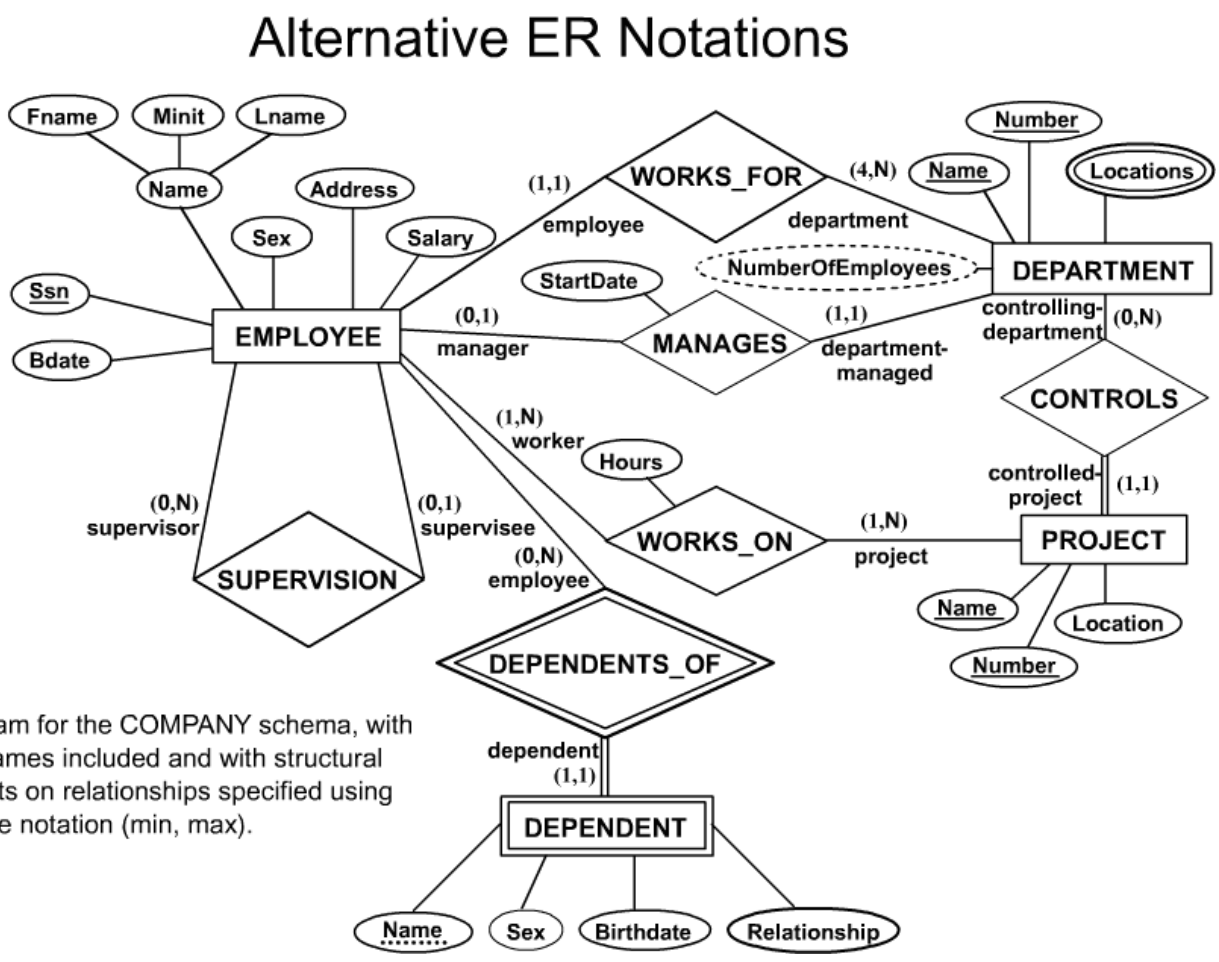
- **A department has *exactly one* manager and an employee can manage *at most one* department.**
  - **Specify (0,1) for participation of EMPLOYEE in MANAGES**
  - **Specify (1,1) for participation of DEPARTMENT in MANAGES**

An employee can work for *exactly one* department but a department can have *any number of employees*.

- **Specify (1,1) for participation of EMPLOYEE in WORKS_FOR**
- **Specify (0,n) for participation of DEPARTMENT in WORKS_FOR**

The (min,max) notation relationship constraints





ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).

Figure 6

6. Data Modeling Tools

Some of the Currently Available Automated Database Design Tools

Oracle Designer, ER-WIN, Rational Rose, and Visio