THE ENTITY-RELATIONSHIP DATA MODEL

CONCEPTUAL MODELLING

OVERVIEW OF DATABASE DESIGN PROCESS

Two main activities:

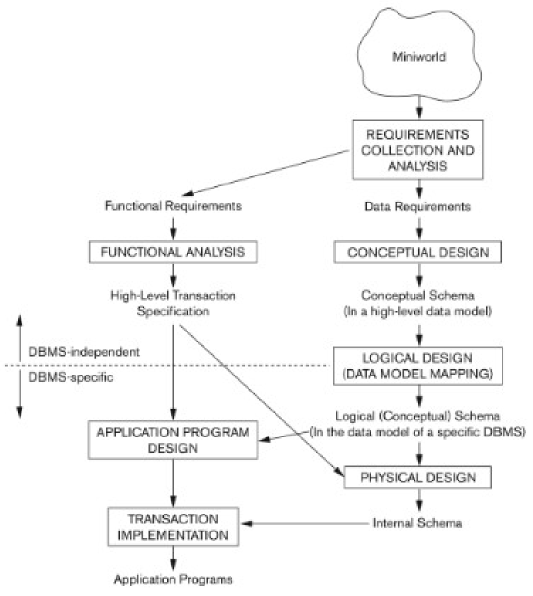
- Database design
- Applications design

Focus on database design here

- To design the conceptual schema for a database application

Applications design focuses on the programs and interfaces that access the database

- Generally considered part of software engineering.

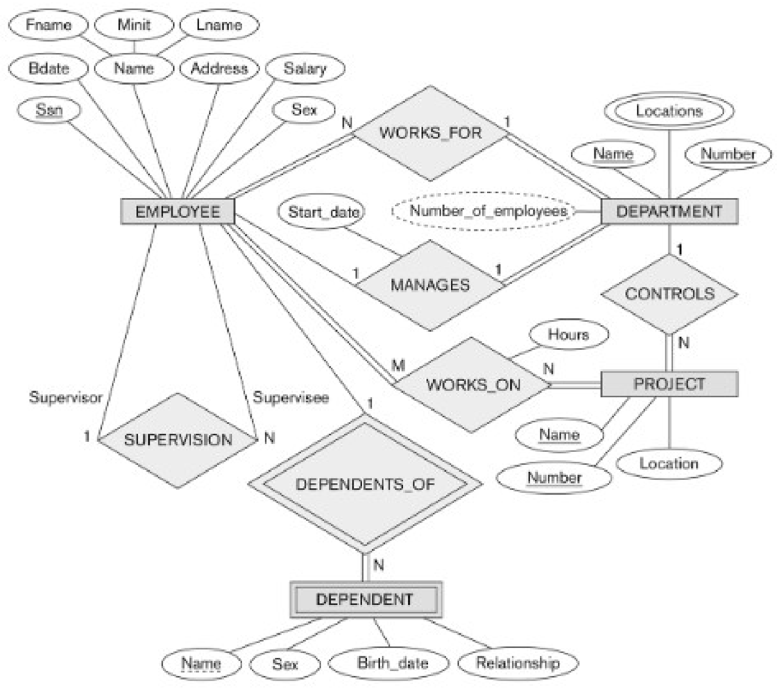

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

EXAMPLE COMPANY DATABASE

We need to create a database schema design based on the following (simplified) requirements of the `COMPANY` database:

- The company is organised into `DEPARTMENT`s. Each department has a name, 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 `PROJECT`s. Each project has a unique name, unique number and is located at a single location.
- We store each `EMPLOYEE` social security number, address, salary, gender and birthdate.
  - 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 `DEPENDENT`s.
  - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.



(how do we derive this?...)

ENTITIES AND ATTRIBUTES IN THE ER DATA MODEL

Based on the previous specification of a mini-world, we define the conceptual model based on Entity-Relationship principles

- An Entity-Relationship model has three main concepts:
- Entities (and their entity types and entity sets)
- Attributes (simple, composite, single-valued, multi-valued)
- Relationships (and their relationship types and relationship sets).

Entities are specific objects or things in the mini-world that are represented in the database.

ENTITIES

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

Entities may be either physical (employees) or non-physical objects (projects)

ATTRIBUTES

Attributes are properties used to describe an entity

- An EMPLOYEE entity may have the attributes Name, SSN, Address, BirthDate, etc.

A specific entry will have a value for each of its attributes

- A specific employee may have the Name='John Smith', etc.

Each attribute has a value set (or data type) associated with it, e.g. integer, string, date.

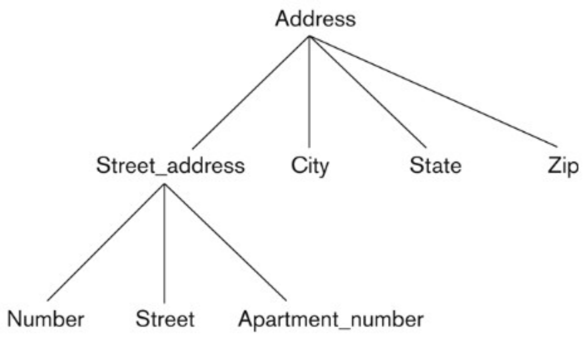
TYPES OF ATTRIBUTES

Simple Attribute

- Each entity has an atomic (non-divisible) value for this attribute
- For example: Employee Social Security Number (SSN)

Composite

- The attribute may be composed of several components
- For example:
  - Address(Apt#, House#, ...)
  - Name (FirstName, MiddleName, LastName).



Single-valued

- Each entity can have a single value for this attribute
- For example: the "Age" attribute for each `EMPLOYEE` entity

Multi-valued

- An entity may have multiple values for that attribute
- For example: Colour of a CAR Entity or `PreviousDegrees` of a `STUDENT`.
- Denoted as `{Colour}` or `{PreviousDegrees}`

An attribute can be both composite and multi-valued.

For example, `PreviousDegrees` of a `STUDENT` is a composite and multi-valued attribute

- Denoted by `{PreviousDegrees (College, Year, Degree, Field)}`
  Multiple `PreviousDegrees` values can exist
- Each degree has four subcomponent attributes
  - 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 entity type `EMPLOYEE` and `PROJECT`

An attribute of an entity type for which each entity must always have a unique value that is called a key attribute of the entity type.

- For example, SSN of `EMPLOYEE`

A key attribute must be composite.

- Example (from US car IDs):
    - `VehicleTagNumber` is a key of the CAR entity type with components (Number, State)

An entity type may have more than one key attribute

- The `CAR` entity type may have two keys:
- `VehicleIdentificationNumber` (popularly called VIN)
- `VehicleTagNumber` (Number, State) or licensed plate number.

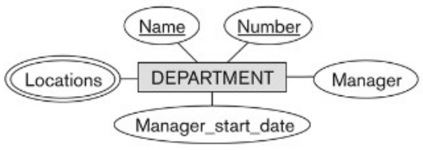
Each key attribute is underlined in the ER diagram.

DISPLAYING AN ENTITY TYPE

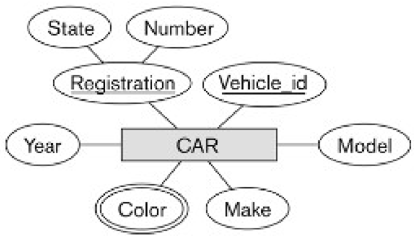
In ER diagrams, an entity type is displayed in a rectangular box.

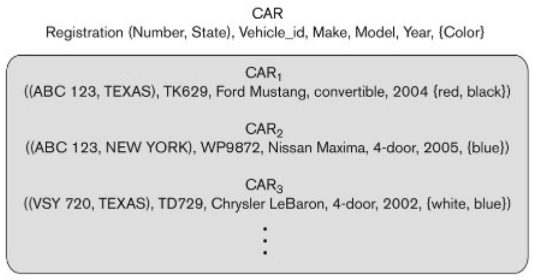
Attributes are displayed in ovals.

- Each attribute is connected to its entity type
- Components of a composite attribute are connected to the oval representing the composite attribute
- Each key attribute is underlined
- Multivalued attributes displayed in double ovals.



Entity type car with two keys and a corresponding entity set:




CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

ENTITY SET

Each entity type will have a collection of entities stored in the database

- Called the entity set

Three car entity instances in the entity set for CAR are shown

Same name (CAR) used to refer to both the entity type and the entity set

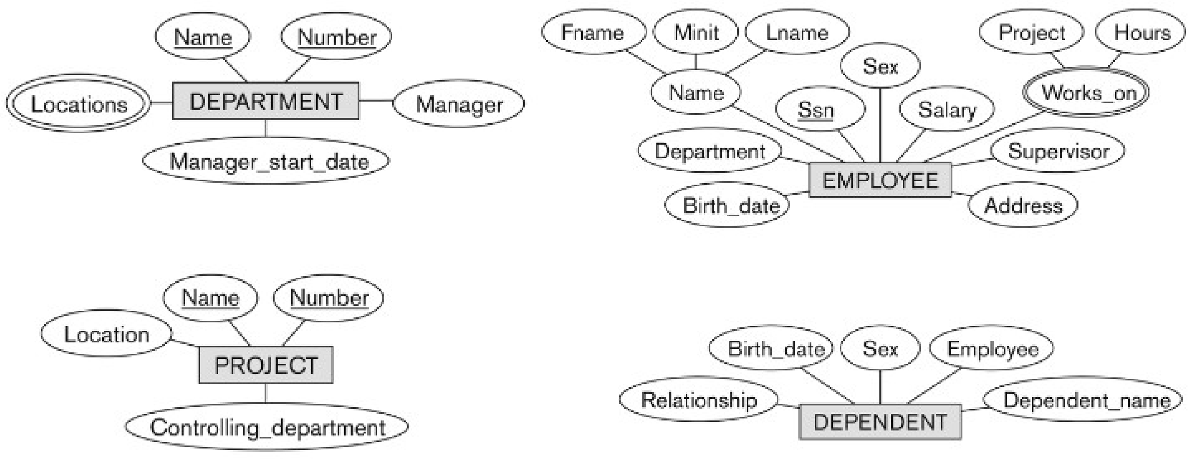
Entity set is the current state of the entities of that type that are stored in the database.

INITIAL DESIGN OF ENTITY TYPES

Based on the requirements, we can identify four initial entity types in the COMPANY database:

- DEPARTMENT
- PROJECT
- EMPLOYEE
- DEPENDENT

The initial attributes are derived from the requirements description



RELATIONSHIPS

A relationship relates two or more distinct entities with a specific meaning

- For example:
    - EMPLOYEE John Smith works on the ProductX PROJECT

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

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

- WORKS_ON is a binary relationship.

RELATIONSHIP TYPE VS RELATIONSHIP SET

Relationship Type:

- A set of associations among entities
- The schema description of a relationship
- Identifies the relationship name and the participating entity types
- Also identifies certain relationship constraints

Relationship Set:

- The current set of relationship instances represented in the database
- The current state of a relationship type

In ER diagrams, we represent the relationship type as follows:

- Diamond-shaped box is used to display a relationship type
- Connected to the participating entity types via straight lines

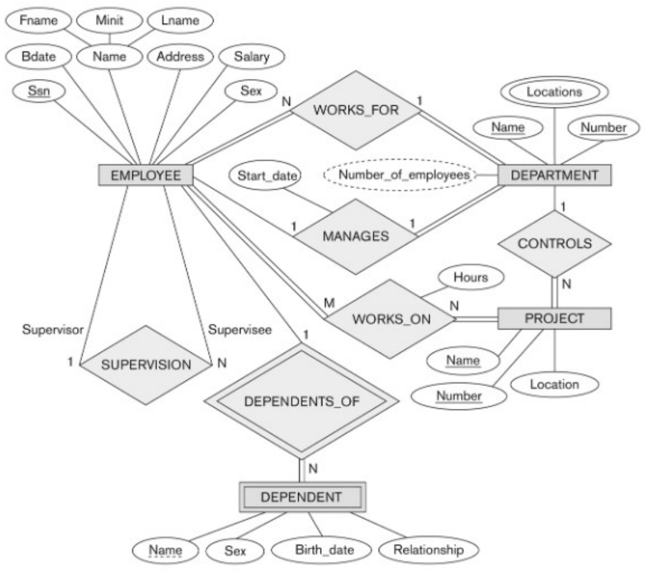
COMPANY DATABASE REQUIREMENTS

- An employee **manages** the department
- A department **controls** a number of projects
- An employee **works for** one department, but may **work on** several projects. Each employee has a direct **supervisor**.
- Each employee may **have** a number of dependents.

By examining the requirements, six relationship types are identified:

- `WORKS_FOR` (between EMPLOYEE, DEPARTMENT)
- `MANAGES` (also between EMPLOYEE, DEPARTMENT)
- `CONTROLS` (between DEPARTMENT, PROJECT)
- `WORKS_ON` (between EMPLOYEE, PROJECT)
- `SUPERVISION` (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
- `DEPENDENTS_OF` (between EMPLOYEE, DEPENDENT)

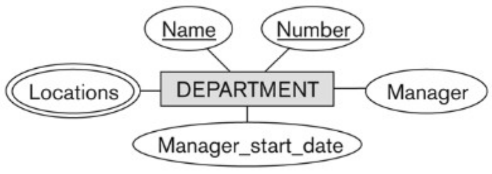
Most are binary relationships, apart from `SUPERVISION`, that is unary.

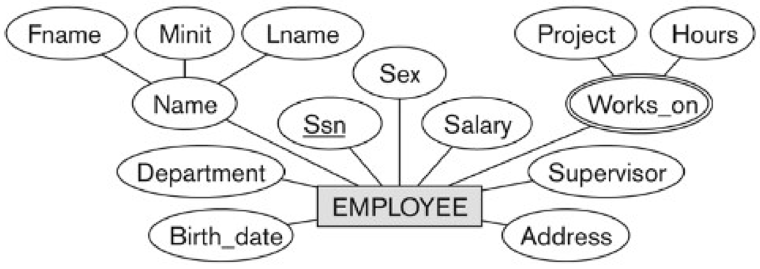


DISCUSSION ON RELATIONSHIP TYPES

In the refined design, some attributes from the initial entity types are refined into relationships:

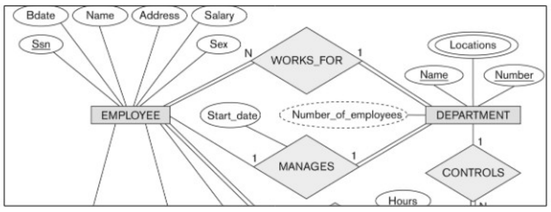
- `Manager` of `DEPARTMENT` → `MANAGES`
- `Works_on` of `EMPLOYEE` → `WORKS_ON`
- `Department` of `EMPLOYEE` → `WORKS_FOR`





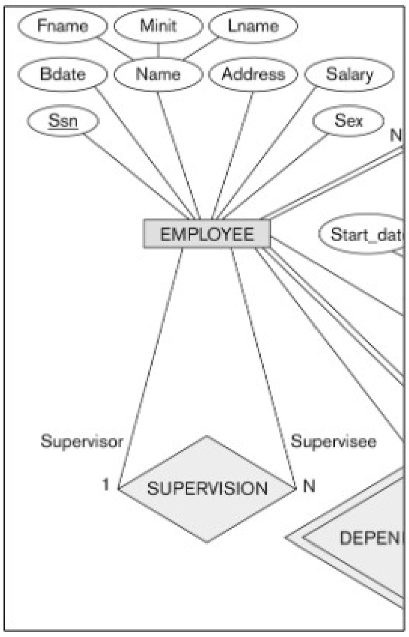
In general, more than one relationship type can exist between the same participating entity types.

- MANAGES and WORKS _FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
- Different meanings and different relationship instances



RECURSIVE RELATIONSHIP TYPE

- A relationship type with the same participating entity type in distinct roles, e.g. the SUPERVISION relationship.
- EMPLOYEE participates twice in two distinct roles:
  - Supervisor role
  - Supervisee role
- Each relationship instance relates two distinct EMPLOYEE entities:
  - One employee in supervisor role
  - One employee in supervisee role



DISPLAYING A RECURSIVE RELATIONSHIP

In a recursive relationship type:

- Both participations are the same entity type in different roles
  e.g.: SUPERVISION relationships between EMPLOYEE in role of supervisor and another EMPLOYEE in role of subordinate

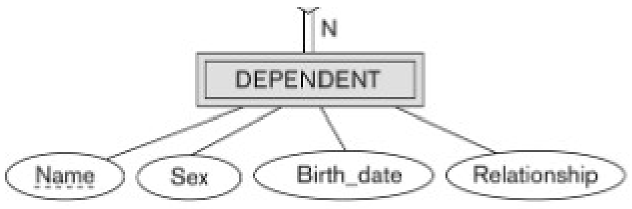
In the figure, first role participation labelled with ‘Supervisor’ and second role participation labelled with ‘Supervisee’.

In ER diagrams, you need to display role names to distinguish participations.

WEAK ENTITY TYPES

An entity that does not have a key attribute is a weak entity type.

- A weak entity must participate in an identifying relationship type with an owner or identifying entity type.
- Entities are identified by the combination of:
    - The particular entity they are related to in the identifying entity type
    - A partial key: an attribute that can uniquely identify weak entities that are related to the same owner entity.
- For example:
    - `DEPENDENT` is a weak entity type.
    - A `DEPENDENT` entity is identified by the dependent's first name and the specific `EMPLOYEE` with whom the dependent is related
    - `EMPLOYEE` is its identifying entity type via the identifying relationship type `DEPENDENT_OF`
    - Name of `DEPENDENT` is the partial key.



STRUCTURAL CONSTRAINTS OF RELATIONSHIPS

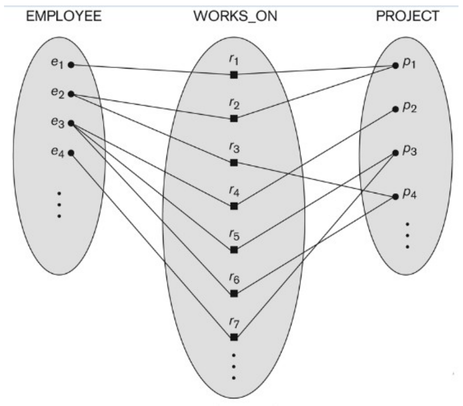
Structural constraints on relationship types are also known as ratio constraints.

- Cardinality Ratio: specifies maximum number of relationship instances that an entity can participate in.
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N)
- Existence Dependency Constraint: specifies minimum participation of an entity in a relationship, i.e. if existence of an entity depends on it being related to another entity via relationship type (also called participation constraint)
  - zero (optional participation, not existence-dependent)
  - one or more (mandatory participation, existence-dependent)

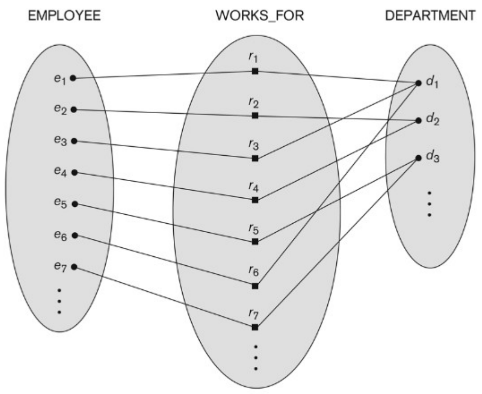
NOTATION

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - Shown by placing appropriate numbers on the relationship edges
- Participation constraint (on each participating entity type)
  - total (called existence dependency) showed by double line
  - partial shown by single line

Each instance of a relationship set relates individual participating entities, one from each participating entity type.



Many-to-many (M:N) Relationship

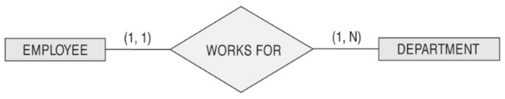


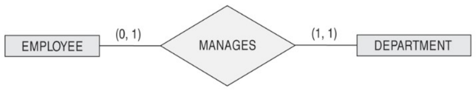
Many-to-one (N:1) Relationship

ALTERNATIVE NOTATION FOR RELATIONSHIPS

Structural Constraints:

- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least min and at most max relationship instances in R
- Default(no constraint): min = 0, max = n (signifying no limit)
- Must have min<= max, min >= 0, max >= 1
- Derived from the knowledge of mini-world constraints.

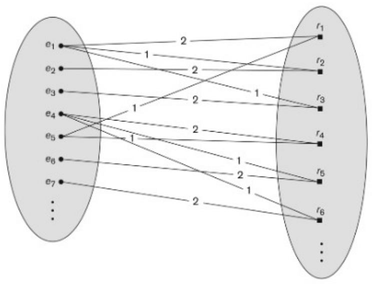




DISPLAYING A RECURSIVE RELATIONSHIP

In figure, first role participation labelled with 1 and second role participation labelled with 2

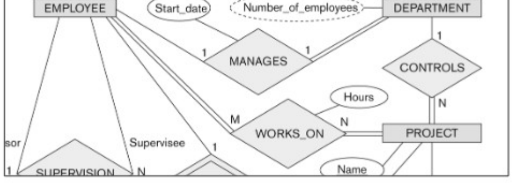
- In ER diagram, need to display role names to distinguish participations.



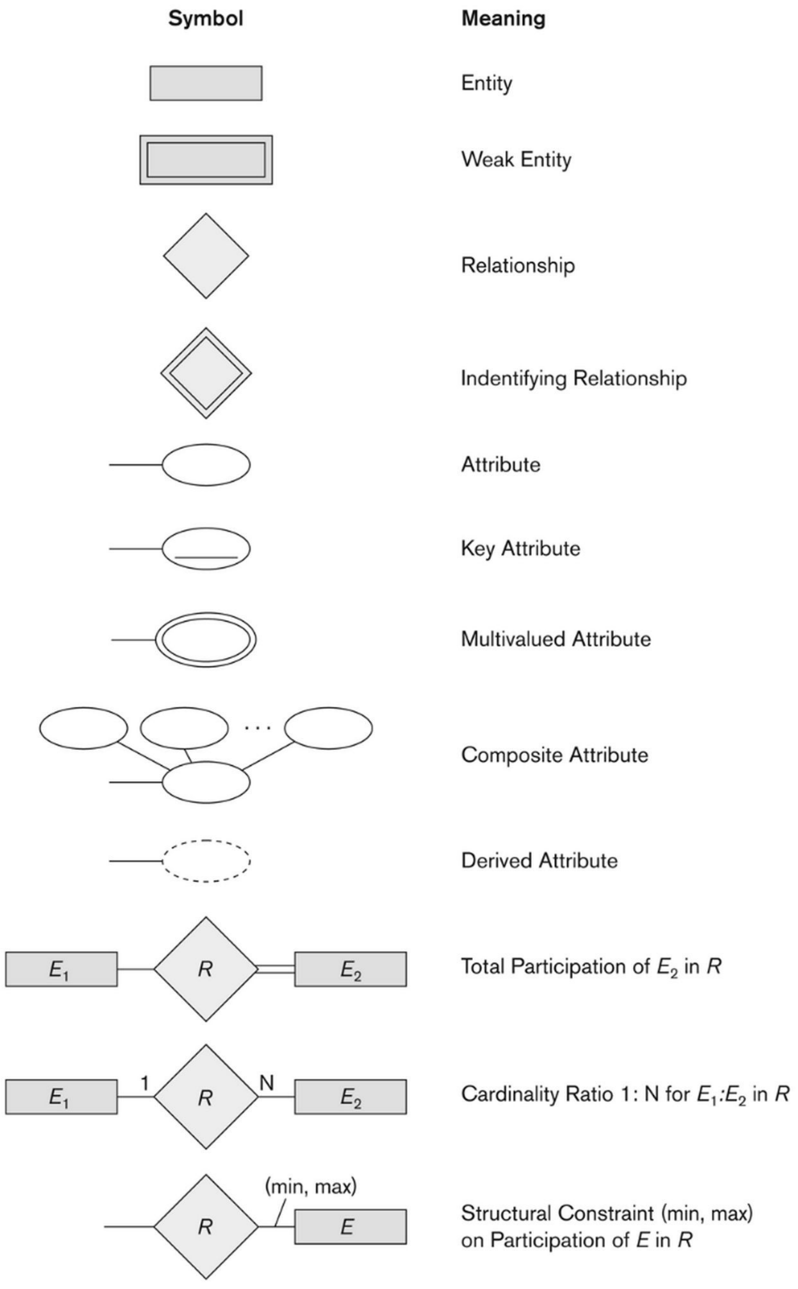
1: supervisor role
2: supervisee role

A relationship type can have attributes:

- e.g. HoursPerWeek of WORKS_ON
- Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT
- Most relationship attributes are used with M:N relationships



SUMMARY


Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying 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
R
(min, max)
E
Structural Constraint (min, max) on Participation of E in R

RELATIONSHIPS OF HIGHER DEGREE

- Relationship types of degree n are called n-ary.
- In general, an n-ary relationship is not equivalent to binary relationships.
- Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships.

ALTERNATIVE DIAGRAMMATIC NOTATION

- ER diagrams is a popular means for displaying database schemas
- Many other notations exist in the literature and in various database design and modelling tools.
- UML class diagrams is representative of another way of displaying ER concepts that is used in several commercial design tools

UML

Represent classes (similar to entity types) as large boxes with three sections:

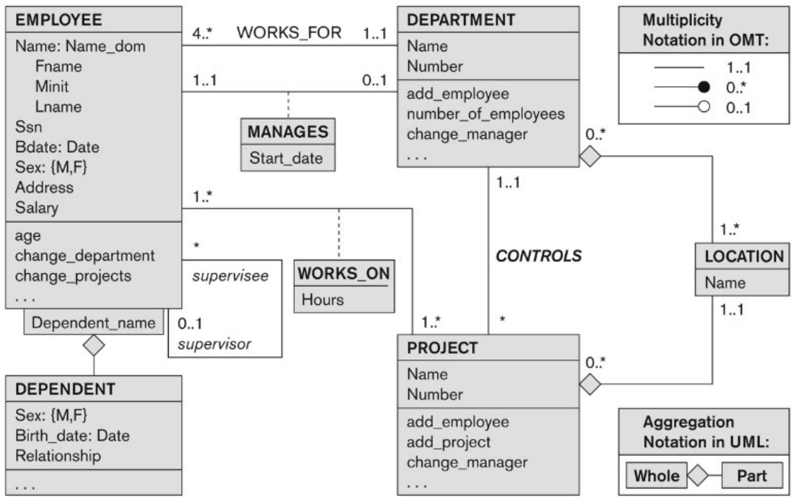
- Top section includes entity type (class) name
- Second section includes attributes
- Third section includes class operations (operations are not in basic ER model)

Relationships (called associations) represented as lines connecting the classes

- Other UML terminology also differs from ER terminology

Used in database design and object-oriented software design

UML has many other types of diagrams for software design.


EMPLOYEE
Name: Name_dom
Fname
Minit
Lname
Ssn
Bdate: Date
Sex: {M,F}
Address
Salary
age
change_department
change_projects
. . .
Dependent_name
4..*
WORKS_FOR
1..1
1..1
0..1
MANAGES
Start_date
1..*
*
supervisee
0..1
supervisor
WORKS_ON
Hours
DEPARTMENT
Name
Number
add_employee
number_of_employees
change_manager
. . .
0..*
1..1
CONTROLS
1..*
*
PROJECT
Name
Number
add_employee
add_project
change_manager
. . .
0..*
Multiplicity Notation in OMT:
1..1
0..*
0..1
1..*
LOCATION
Name
1..1
DEPENDENT
Sex: {M,F}
Birth_date: Date
Relationship
. . .
Aggregation Notation in UML:
Whole
Part