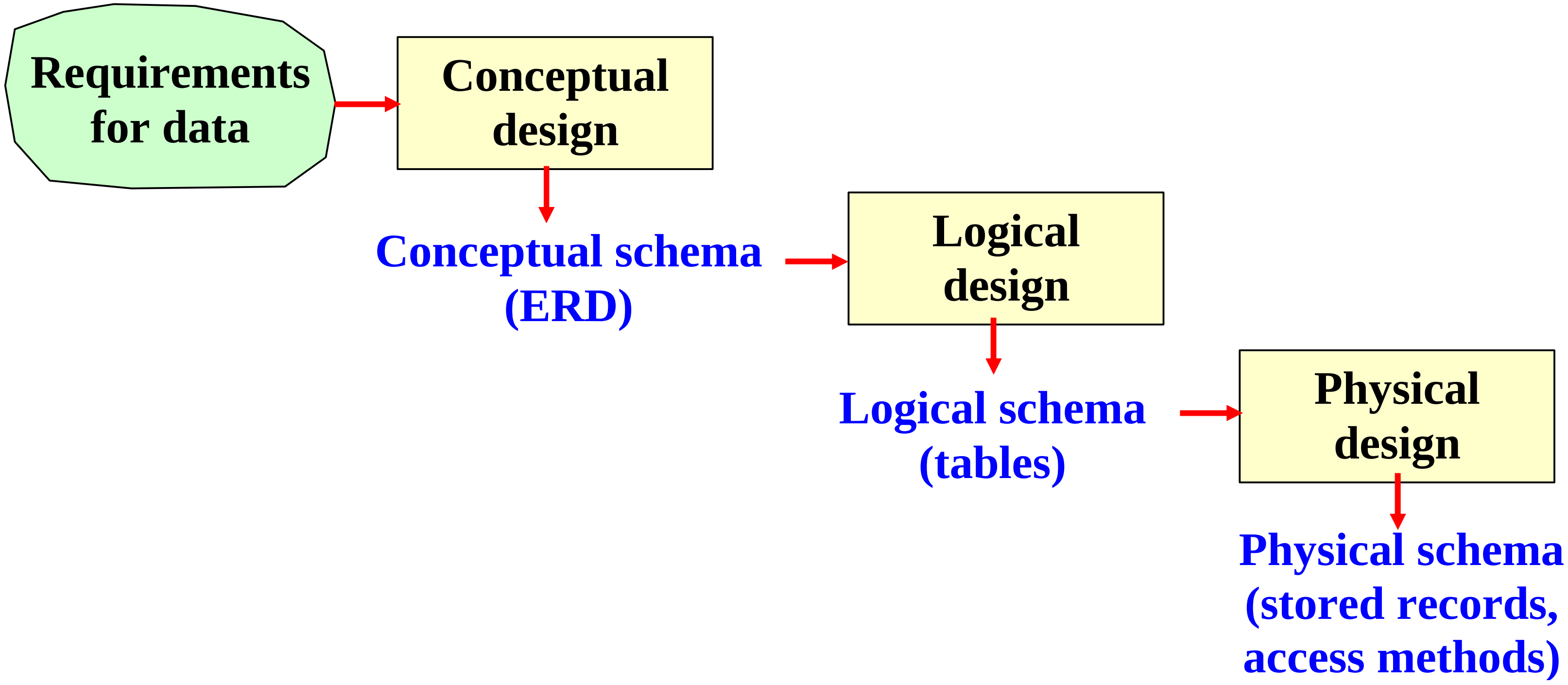
2 Conceptual modelling

2.1 The role of conceptual modelling in database design process

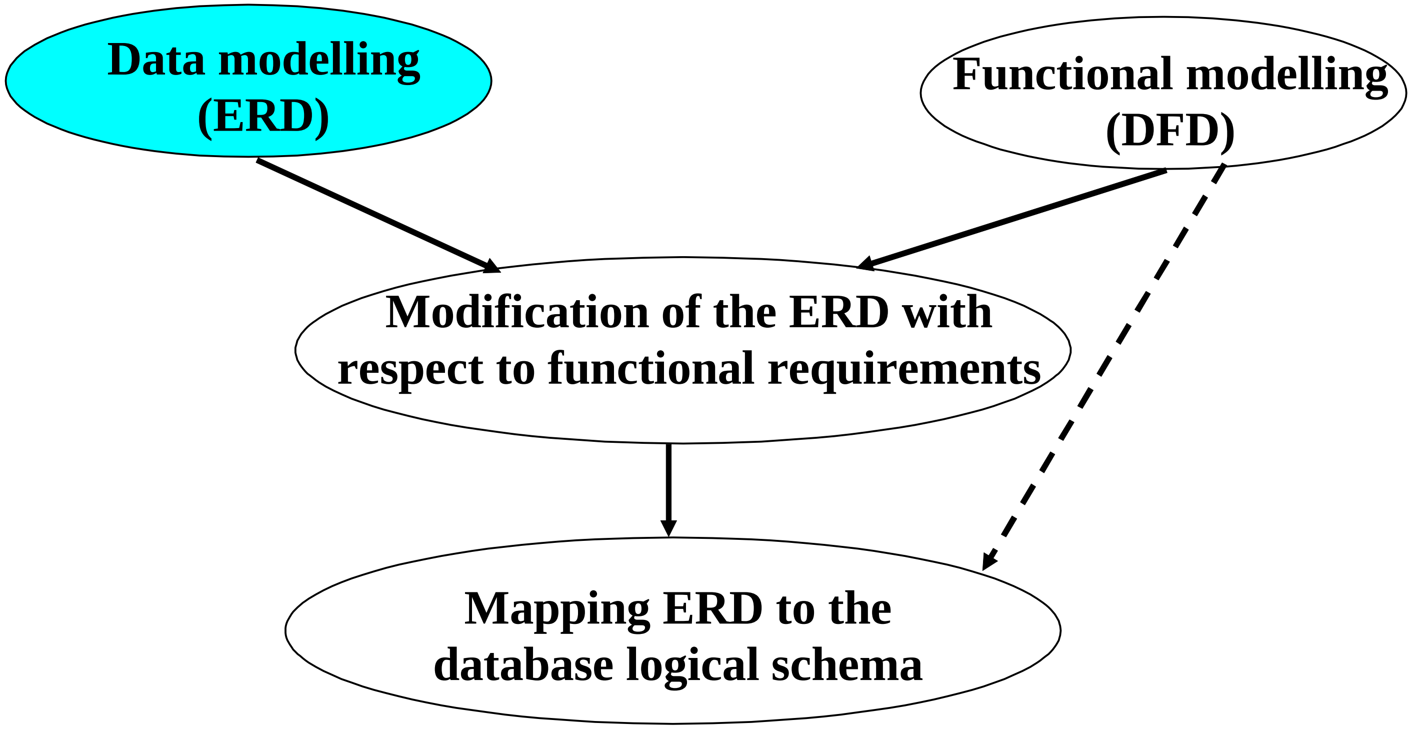
Conceptual modeling **– the step of data or object analysis that describes concepts of the application domain for which we develop the system.**

- **Fundamental steps in database design**

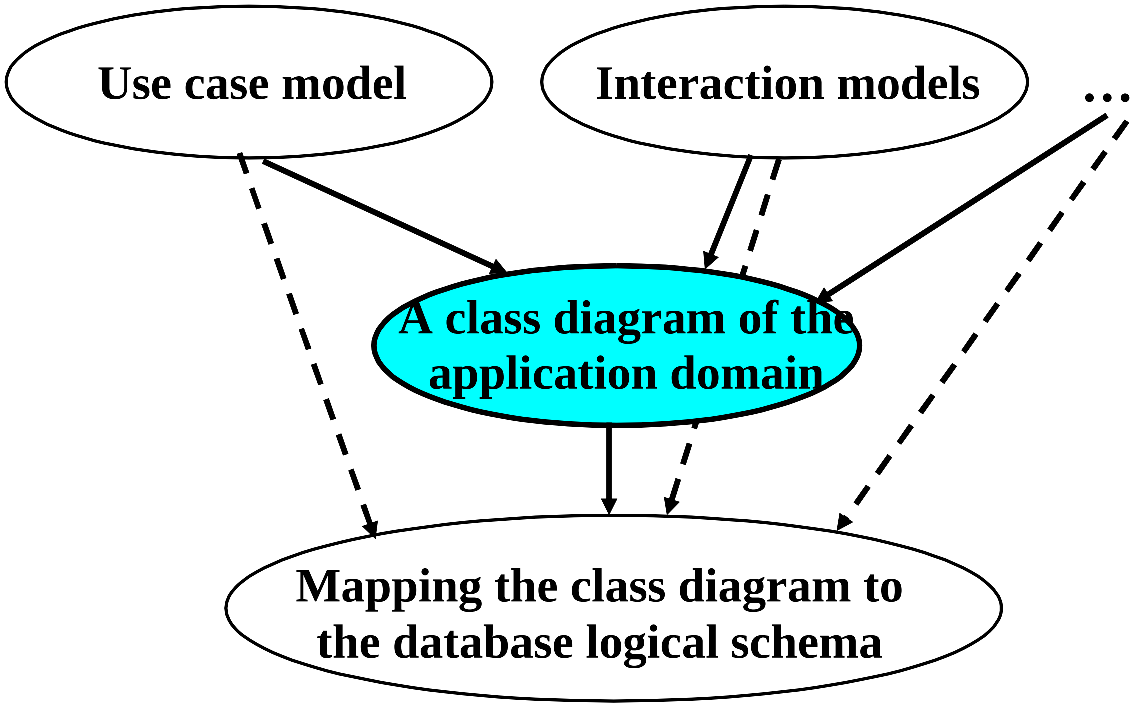


Basic approaches:

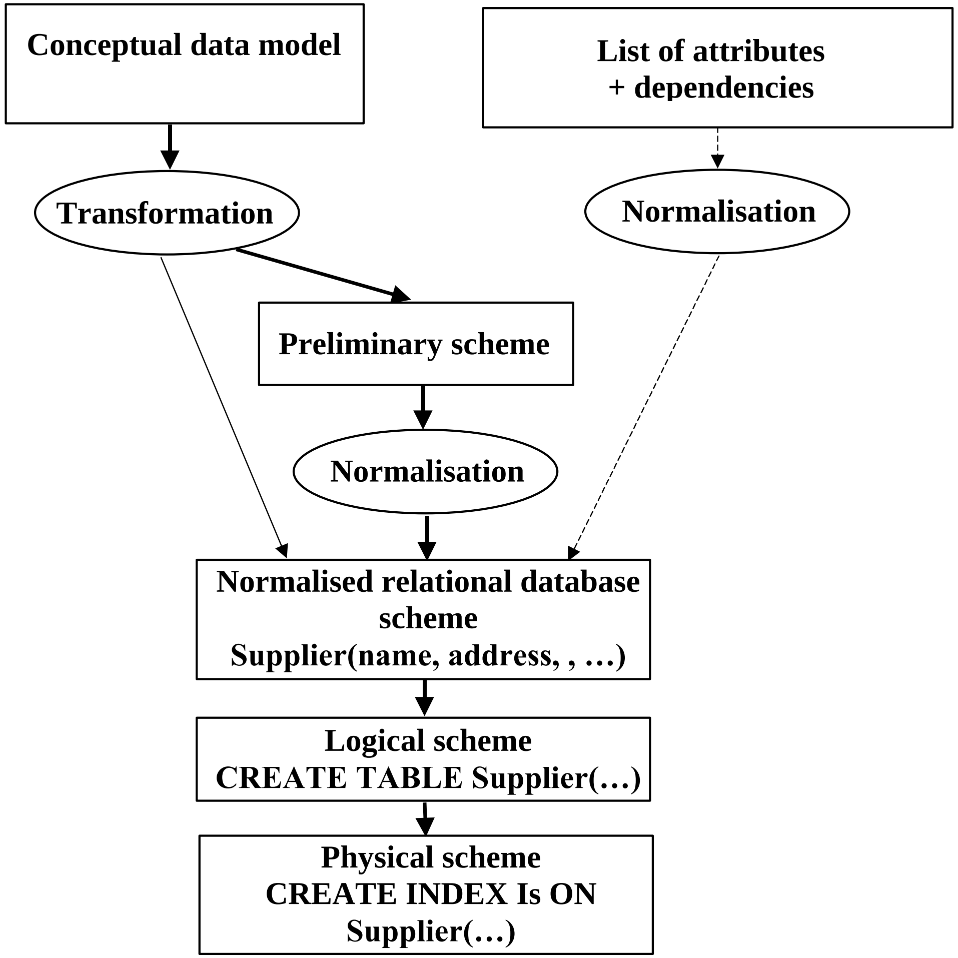
- **Structured** (classic): the base for the database design is the **ER model**


Data modelling (ERD)
Functional modelling (DFD)
Modification of the ERD with respect to functional requirements
Mapping ERD to the database logical schema

- **Object-oriented**: the base for the database design is the **class diagram**

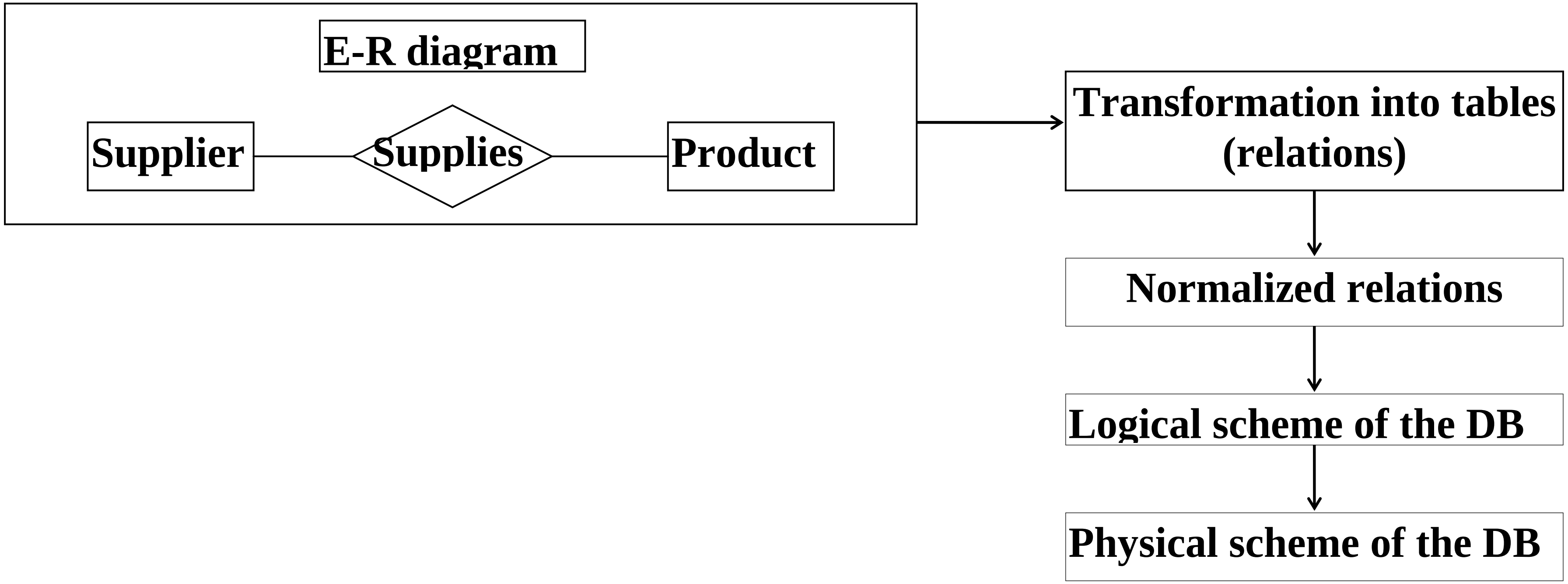

Use case model
Interaction models
...
A class diagram of the application domain
Mapping the class diagram to the database logical schema

- **There are two fundamental approaches of relational database logical structure design:**
  a) conceptual (data) modelling + mapping into tables,
  b) relational analysis (based on normalization)


Conceptual data model
List of attributes + dependencies
Transformation
Normalisation
Preliminary scheme
Normalisation
Normalised relational database scheme Supplier(name, address, , …)
Logical scheme CREATE TABLE Supplier(…)
Physical scheme CREATE INDEX Is ON Supplier(…)

The most practical methodologies support both ways, but start by

examining system at a top level dealing with entities (or objects) and relationships.

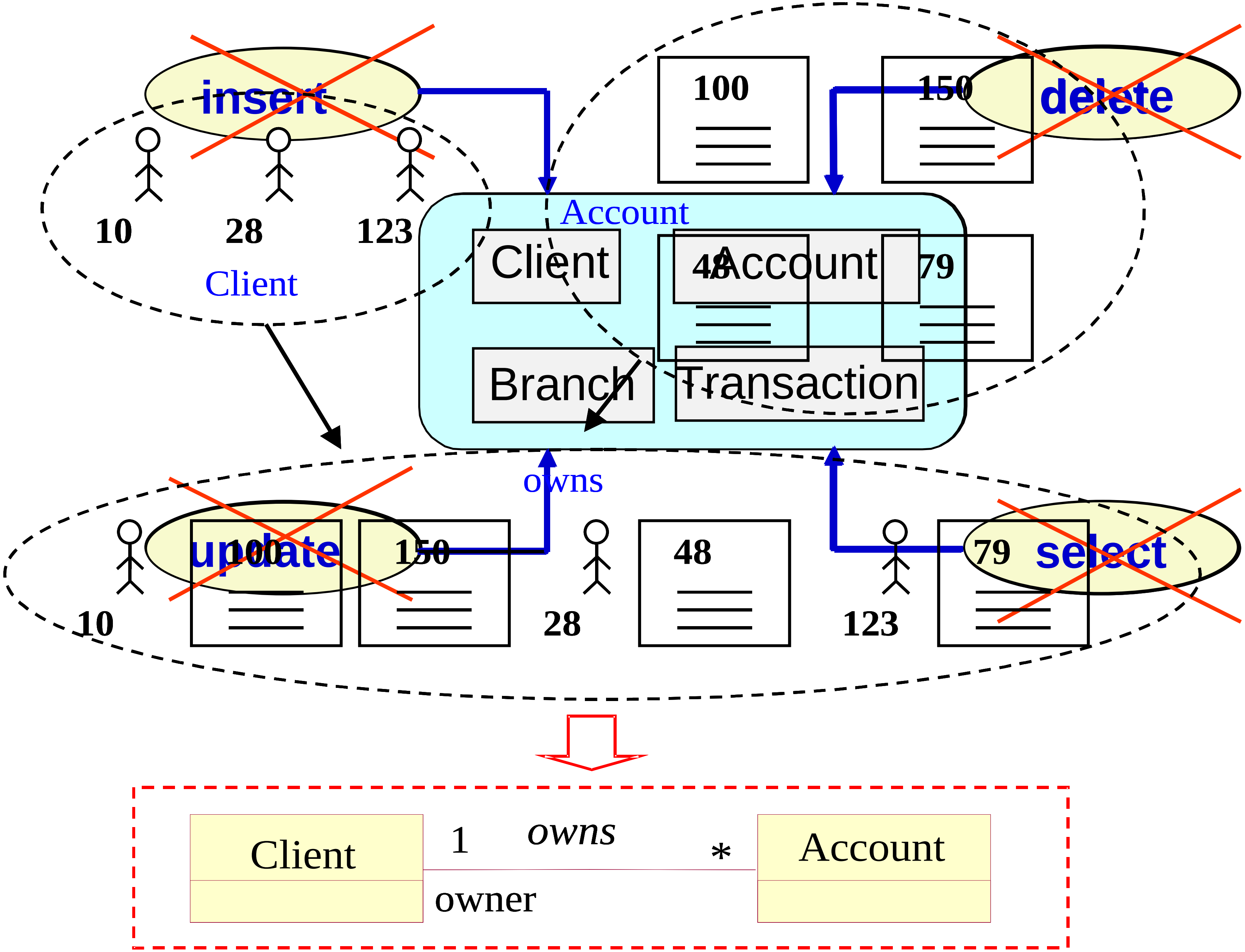


2.2 ER models (Entity Relationship)

ER model is based on perception of the application domain as a set of basic objects, referred to as **entities** and **relationships** among them. It describes data at "rest", it does not show required operations with data. It is sometimes called ERA model because **atributes** of entities and relationships are the third basic element of the model. ER diagram is the graphical form of the model.

- **We model application domain data “at rest”, i.e. we model only static information here**

Ex) An application managing information about clients of a bank, their accounts and operations with them


insert
delete
update
select
10
28
123
Client
100
150
48
79
Account
Client
Account
Branch
Transaction
owns
100
150
48
79
10
28
123
Client
1
owns
*
Account
owner

Entity **- distinguishable object information about which is to be stored in the system.**

Things modelled as entities:

- things that can be physically seen and identified (e.g. clients, branches)
- things that do not exist as physical objects, but which serve to identify some organizational entity (e.g. accounts,)
- things that happen (e.g. shipments, orders)

Entity set **- a set of entities of the same type.**

Attribute **- an entity property the value of which is important for us in the context of a solved problem, and,will be stored in database.**

Ex) Client: clientNo, first name, last name, address, …

Relationship **– an association among entities.**

Ex) A client with clientNo C999 *owns* an account with accountNo A100.

Relationship set **- a set of relationships of the same type.**

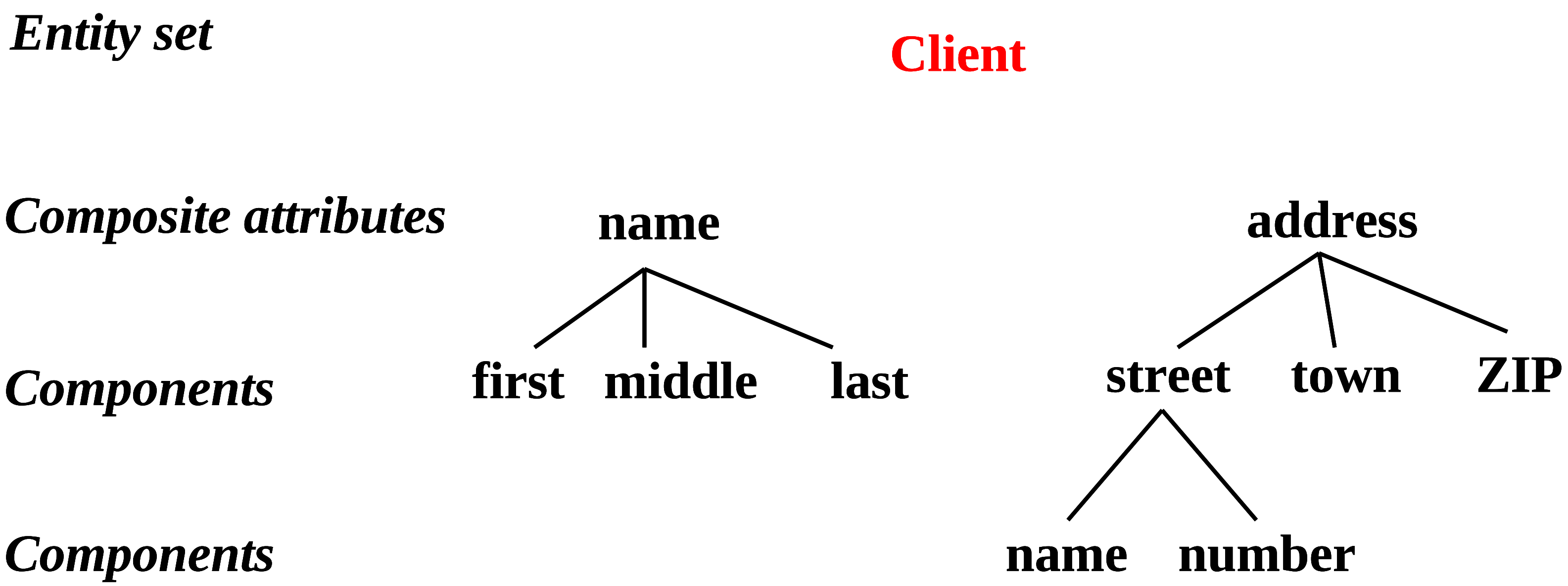
Ex) Client *owns* Account – for the relationships between entities of the type Client and Account

Identifier (primary key) of an entity or relationship set **– an attribute the value of which is unique within a given entity or relationship set, and is minimal (irreducible) (if the attribute is composite – see later).**

Note: We often use names “entity” and “relationship” in the sense of “entity set” and “relationship set”, respectively.

- **Types of attributes**
  - **Simple, composite (depends on semantics)**

    Ex) Name and address as composite attributes.



Entity set Client

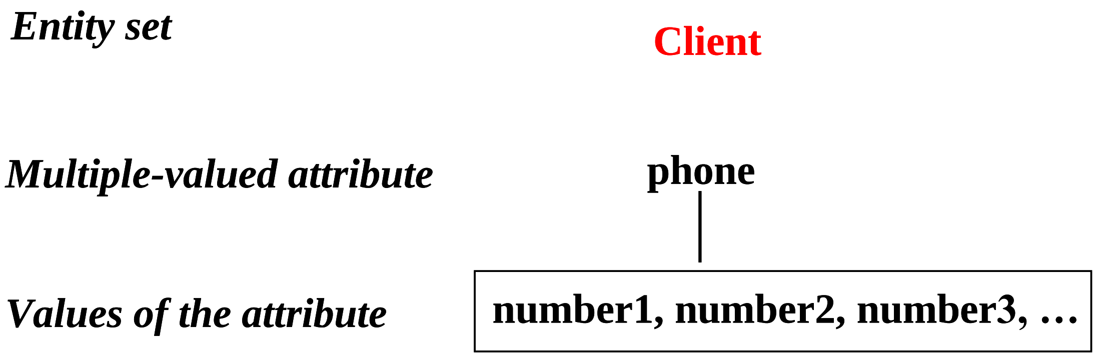
Composite attributes name address

Components first middle last street town ZIP

Components name number

- **Single-valued, multiple-valued**

Ex) Another attribute of Client – phone.



Entity set Client

Multiple-valued attribute phone

Values of the attribute number1, number2, number3, …

- **Allowing an empty value (NULL)**

  Missing **- it exists but we do not know it (e.g. date of birth).**
  Unknown **- it may exist but we do not know it (e.g. phoneNo).**

- **Derived – its value is derived from other attributes or relationships.**

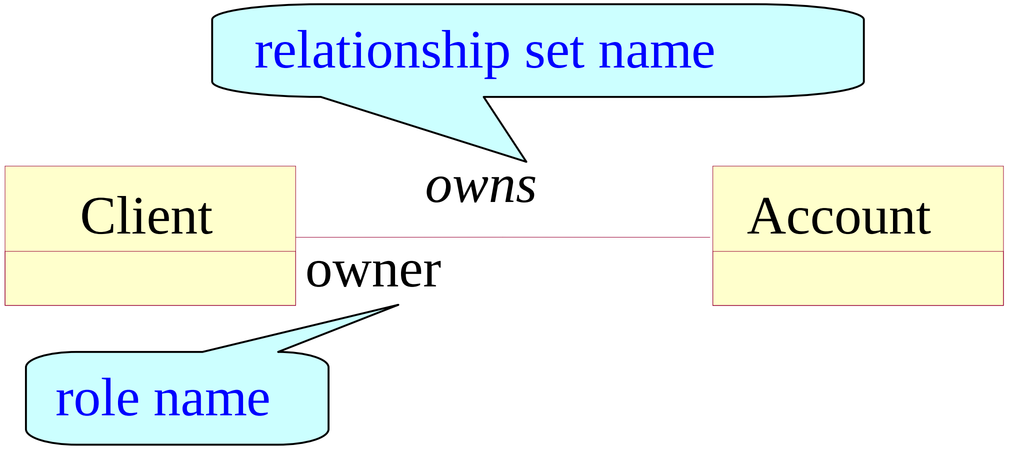
  Ex) The age of a person can be derived from the day of birth

- **Properties of relationships**

  - ***Relationship set name, the role name***

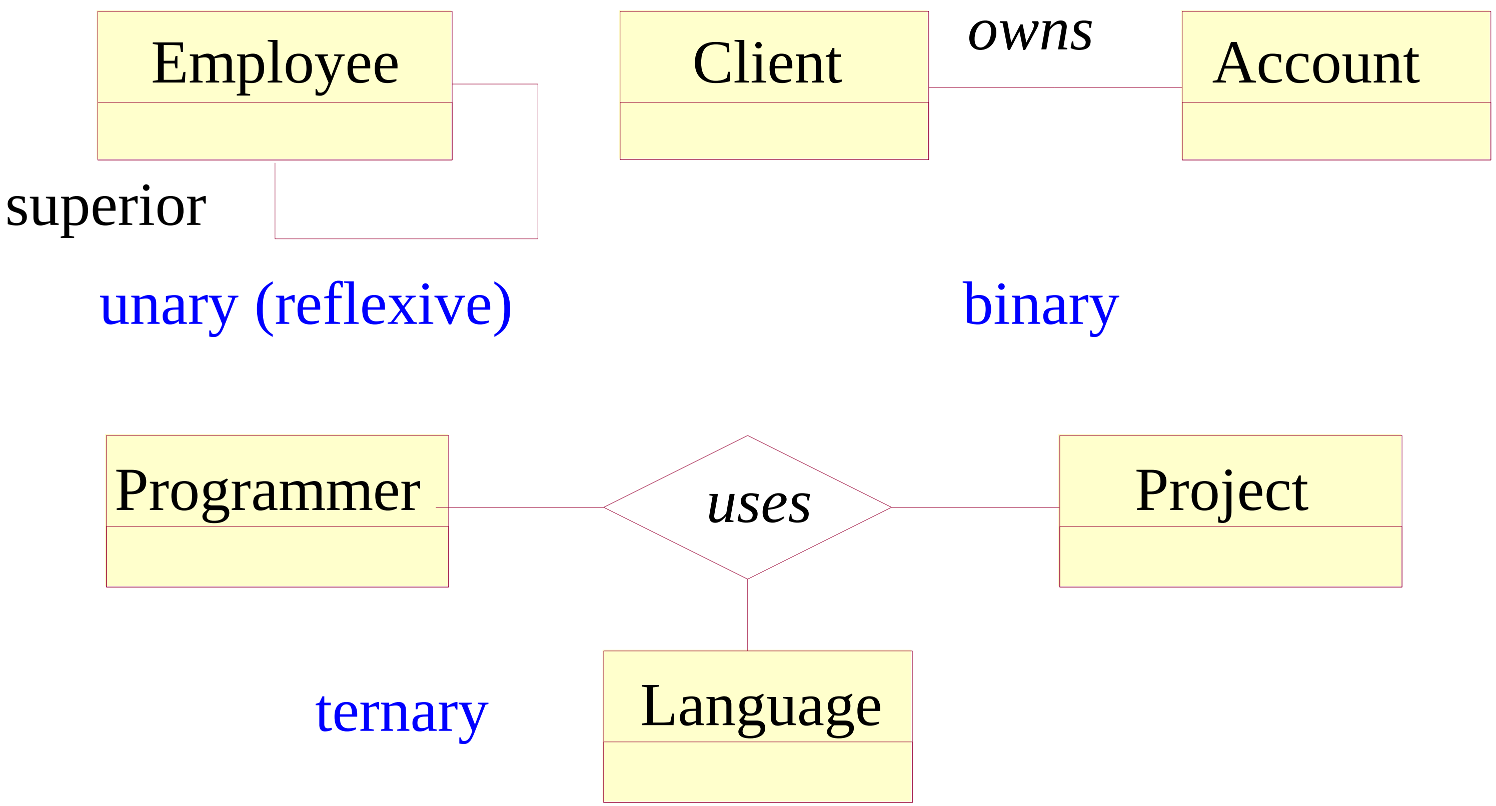
    It must reflect the meaning of the relationship.

    Ex)



➢ *Degree*

Ex)



unary (reflexive)

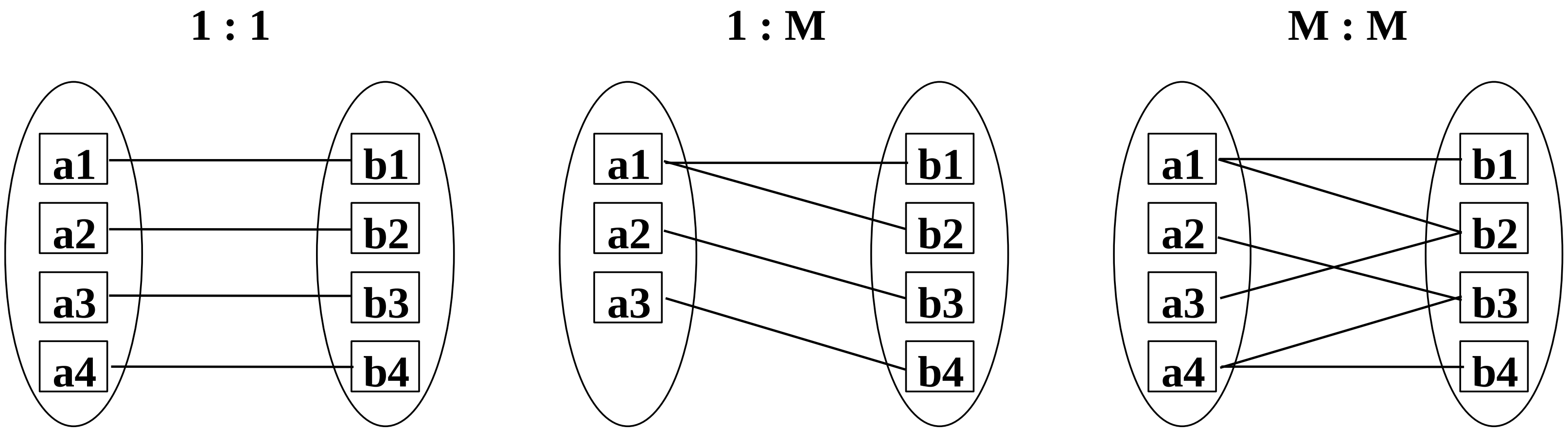
binary

ternary

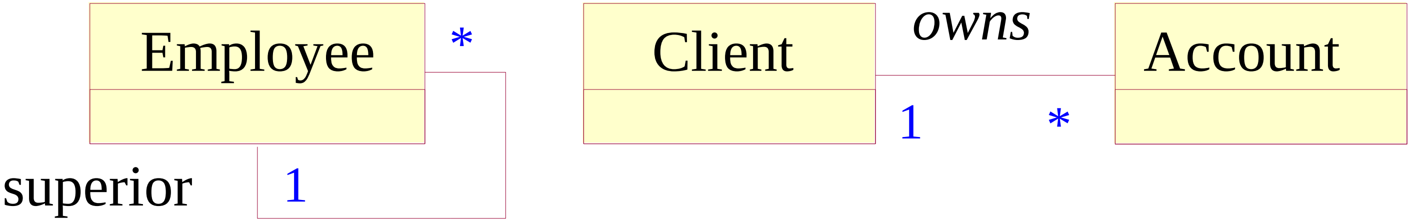
- ***Cardinality*** **(maximum cardinality),**

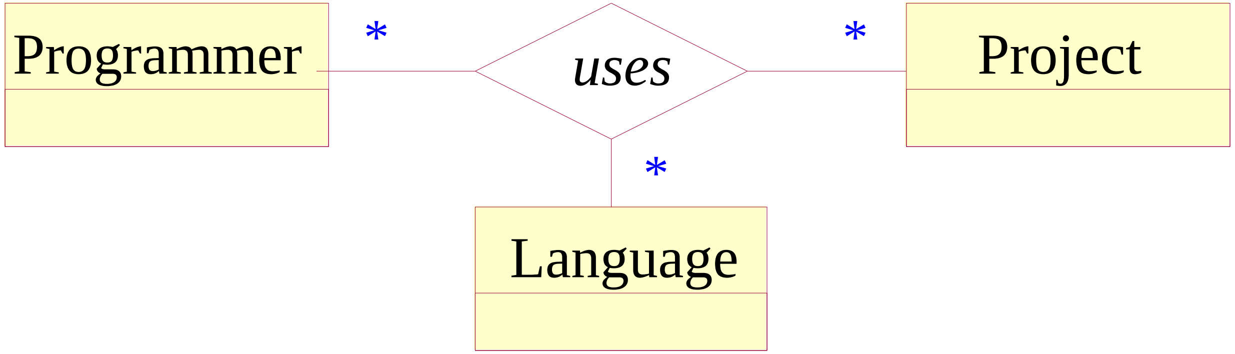
  Maximum number of relationships of a given type (relationship set) in which one entity can participate (1,M(any), or more precisely, e.g. 5).

  - it is a property of the relationship end, so we have to determine it for every end of the relationship.



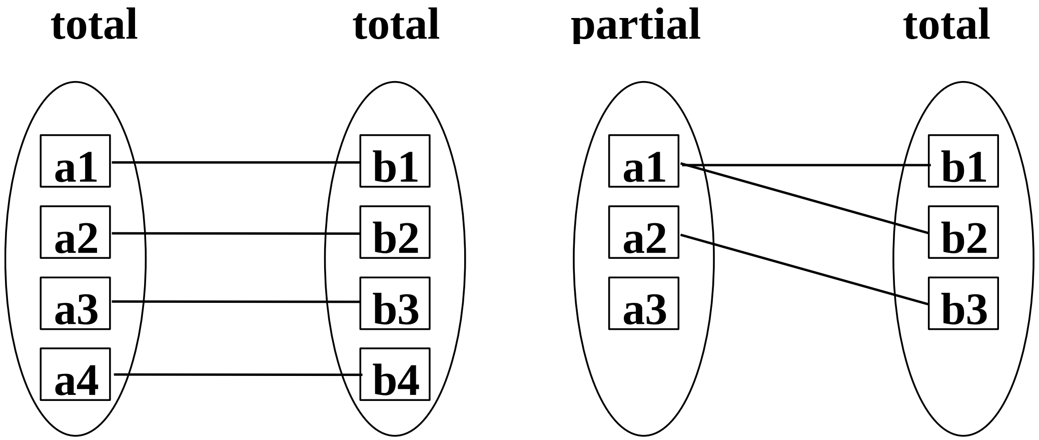
Ex)


Employee
*
superior
1
Client
owns
Account
1
*

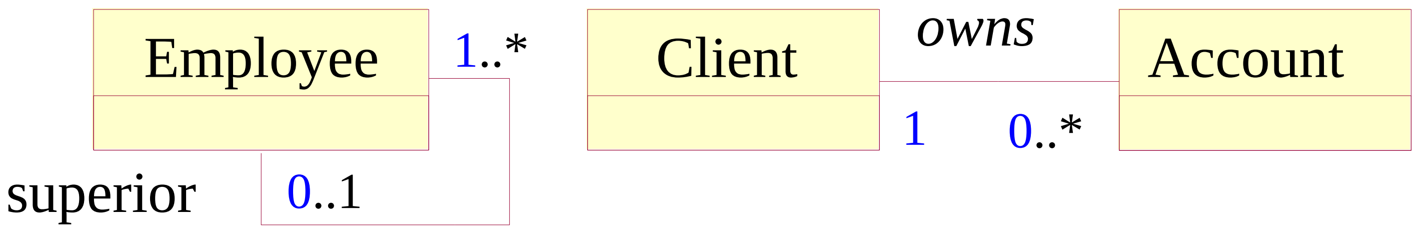

Programmer
*
uses
*
Project
*
Language

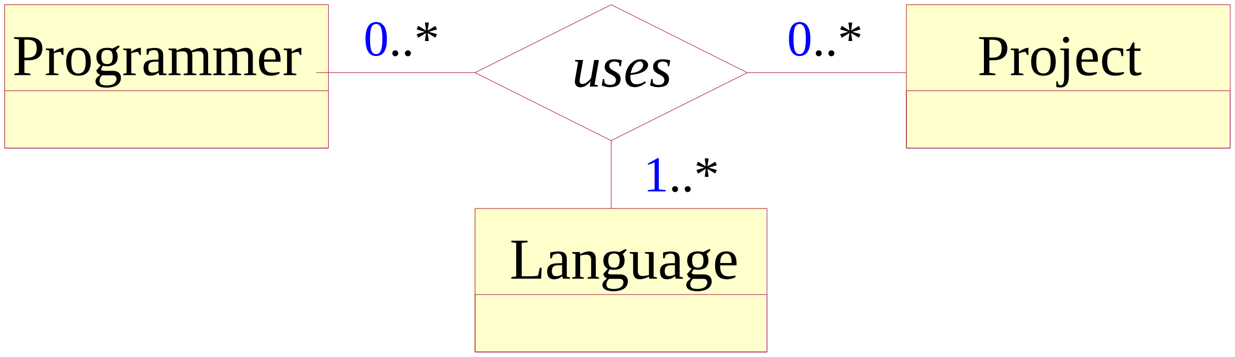
- ***Membership*** **(also called *minimum cardinality*)**

  Minimum number of relationships of a given type (relationship set) in which one entity must participate (0 - optional, 1 - mandatory, or more precisely, e.g. 2).



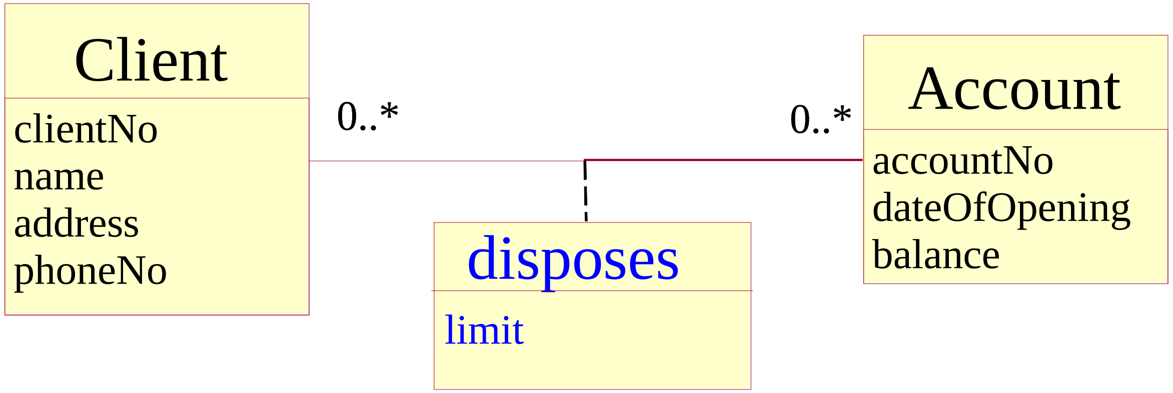
Ex)



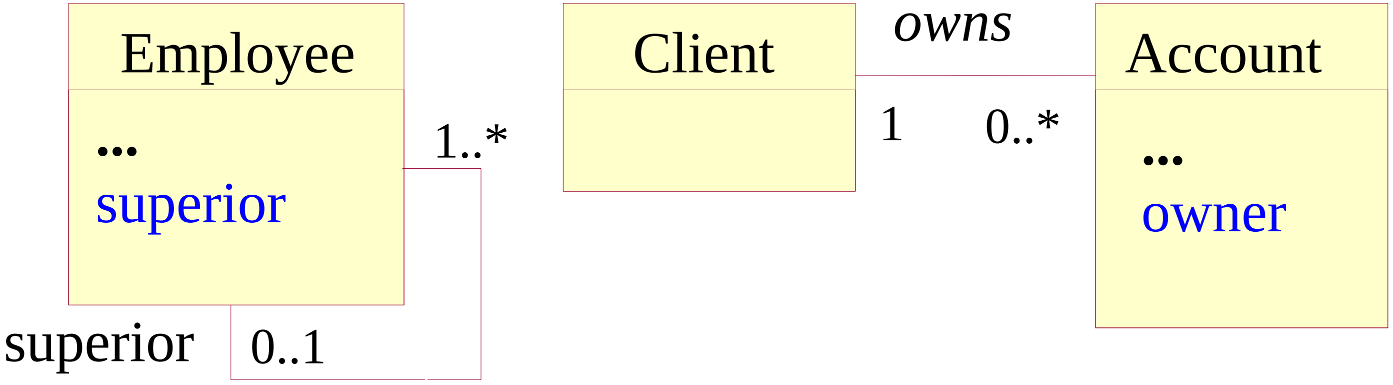


The ability to determine maximum cardinality correctly is a key point of database schema design.

➢ *Atributes* of relationships

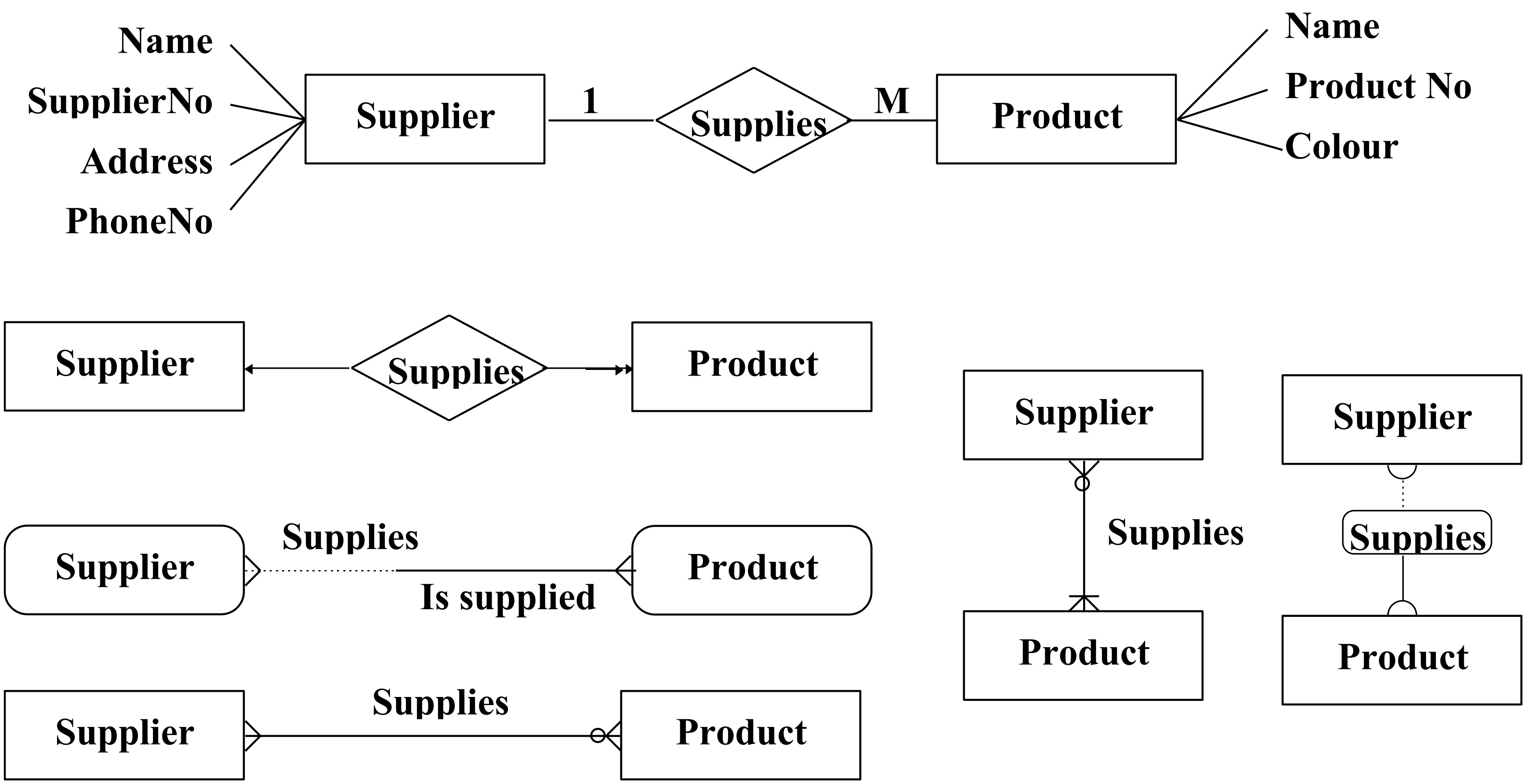


➢ Attributes representing relationships

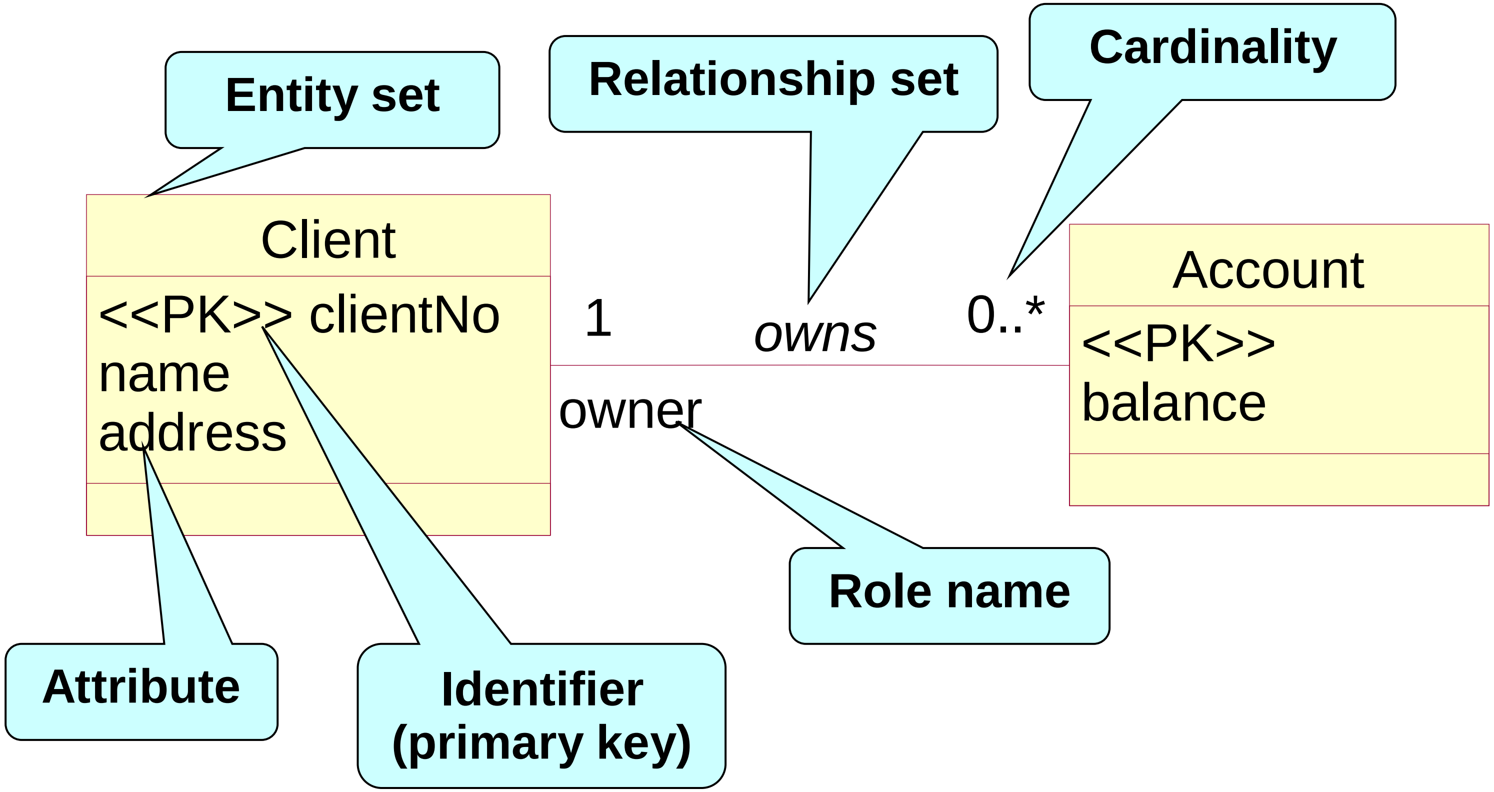


- it is not necessary to show them but it is necessary to draw the lines of relationships

- **Notations for drawing ERDs**

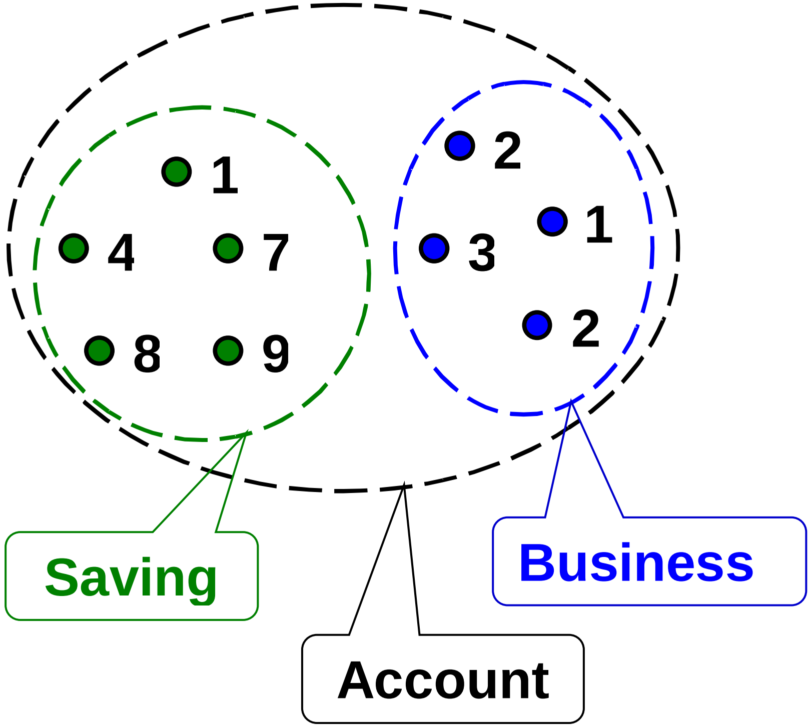


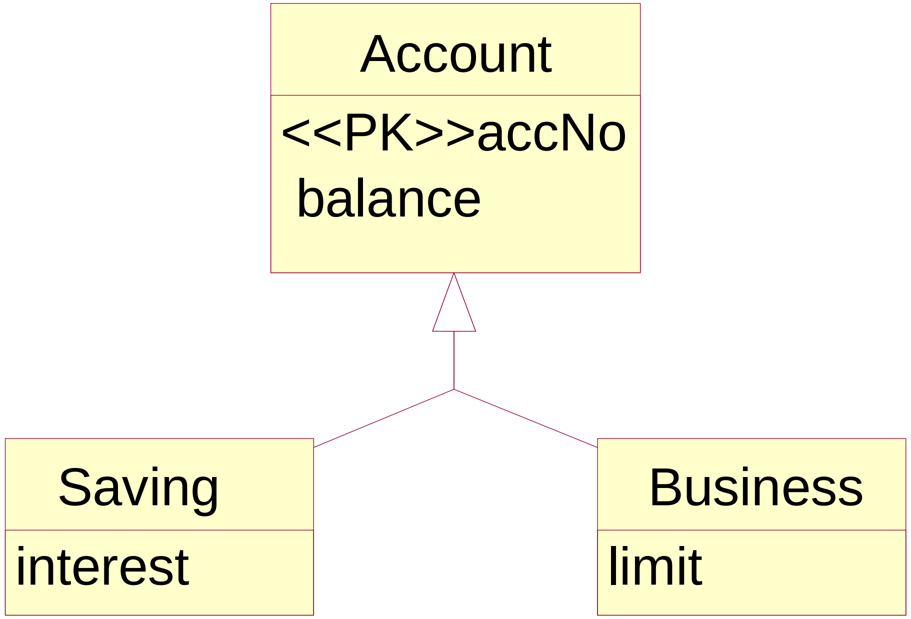
- **We will use a UML-based notation**



2.3 Extended ER model

- **Generalization/specialization (subsets)**





➢ **Possible constraints:**

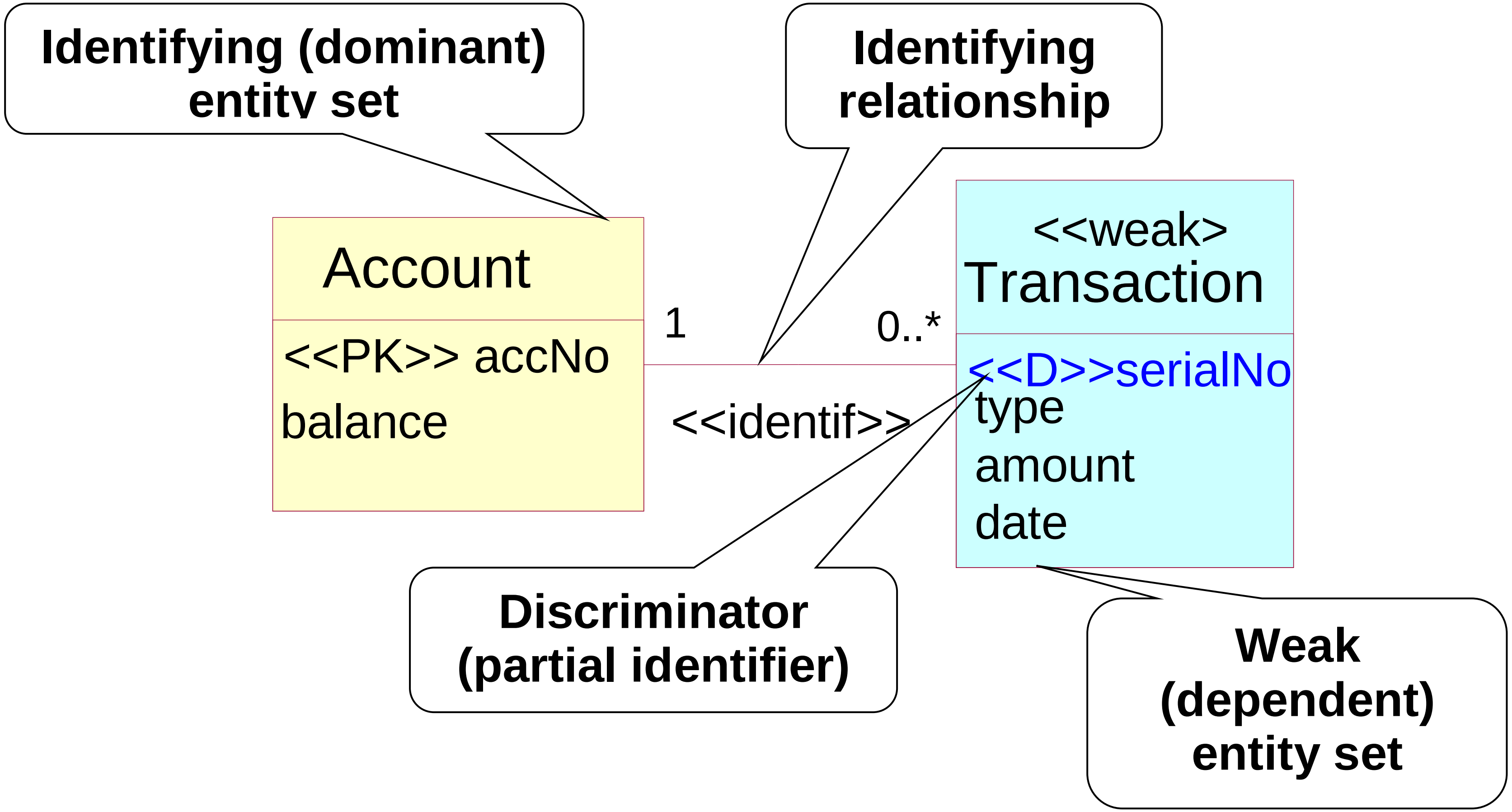
***Membership* – disjoint/overlapping**

***Completeness* – partial/total**

- **Strong and weak entity sets**

A strong entity set – **each its entity can exist independently on other entities, its identifier contains only its own attributes.**

A weak entity set **– each its entity depends on an entity of another entity set. The weak entity set has no attribute that plays the role of identifier. Identification is only partial. It is necessary to add an identifier from the entity set which a weak entity set depends on.**



Rule: If you are not sure about a weak or strong entity set, use a strong one.

2.4 Some guidelines for drawing E-R diagrams

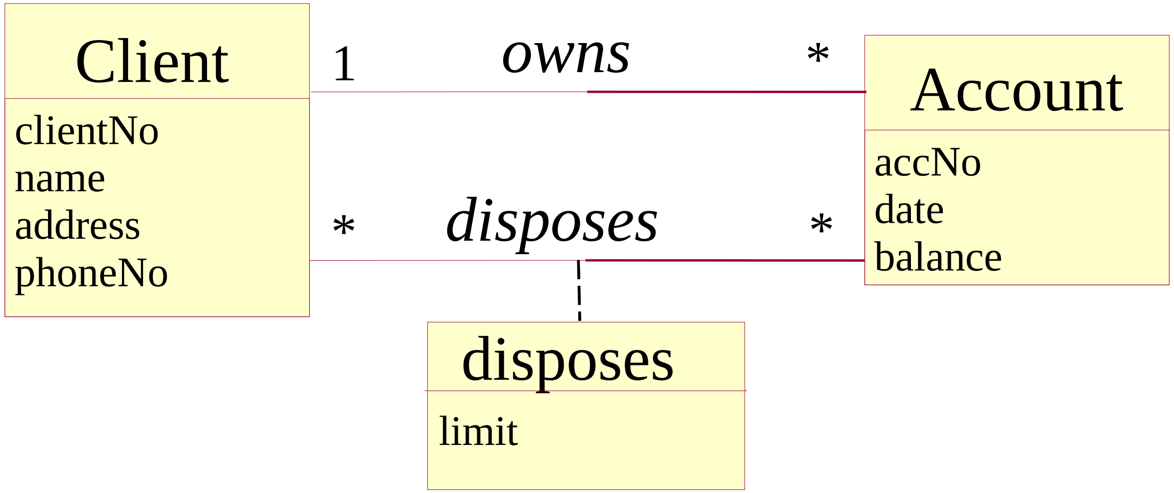
- **Names**

- choose names that make a diagram readable

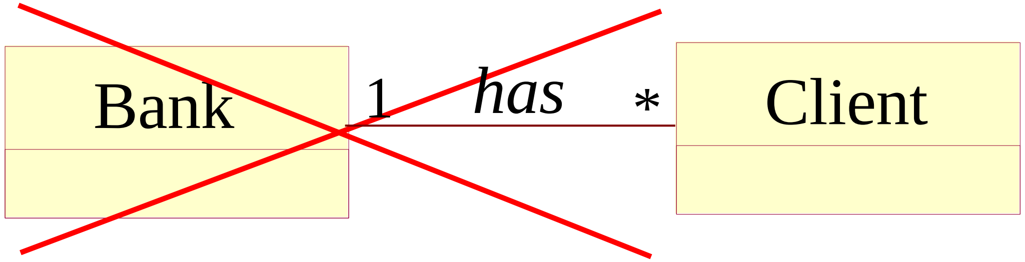
- it is not necessary to name relationships if their meaning is clear

- sometimes, it is better to use role names

- if there are several relationship sets between the same two entity sets, they must be named (or role names used)

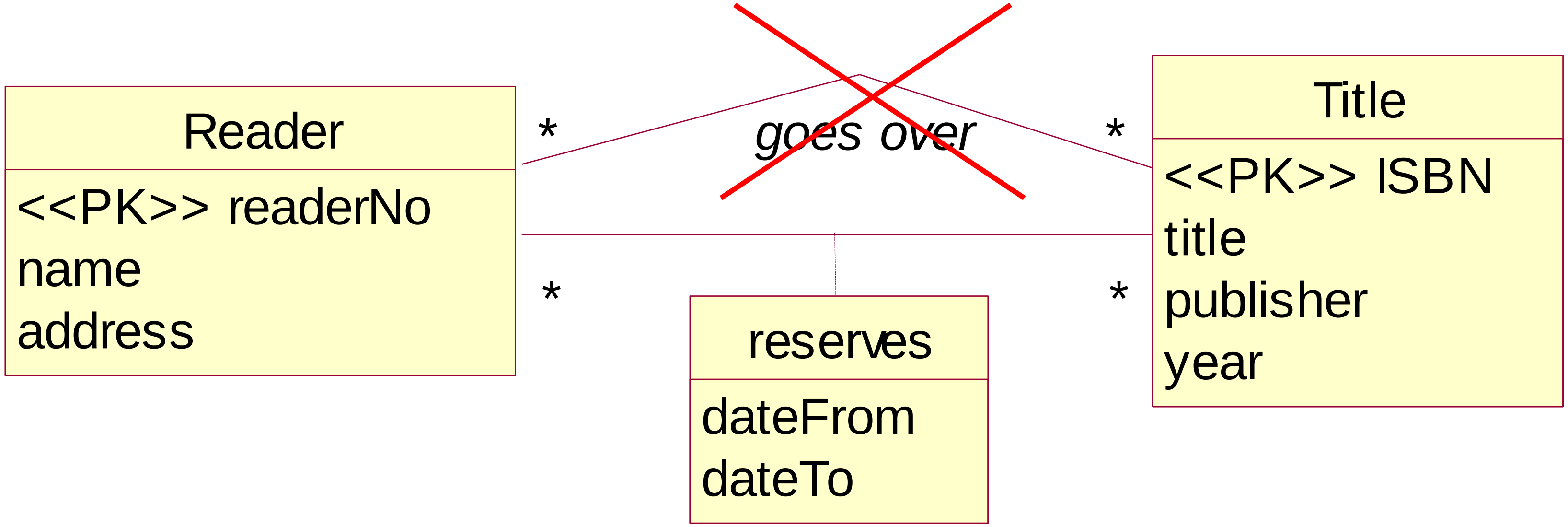


- The total system should not be included in the diagram

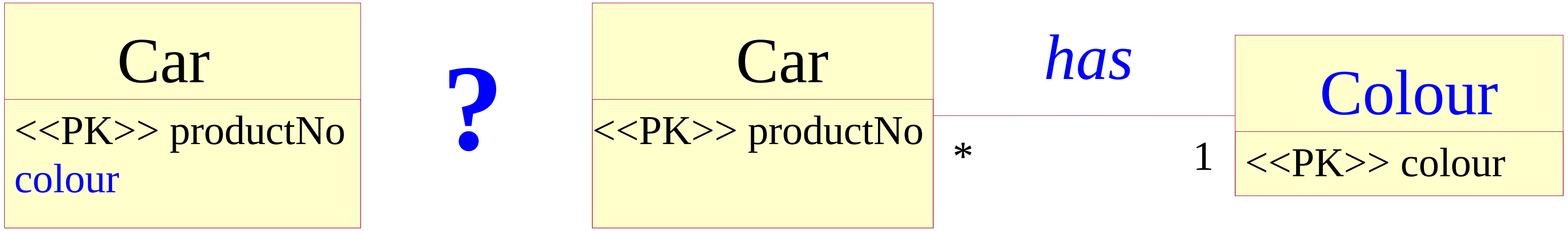


- The choice of the identifier
  - keys uniquely distinguish objects within the system, not within the whole country or whole world (e.g. clientNo)
  - the identifier can be simple or composite
  - if there are several choices, choose the simplest one
  - We use composite identifiers if:
    - ○ It is a natural way of identification for a given entity set (e.g. Course (name, year) if it represents a run of the course)
    - ○ A given entity set is a weak one (e.g. Transaction)
    - ○ A given entity set replaces a relationship set with cardinality M:M
- Avoid relationship sets that represent only operations without the requirement for storing information about the operations.

Ex) …the reader can go over titles and reserve them …

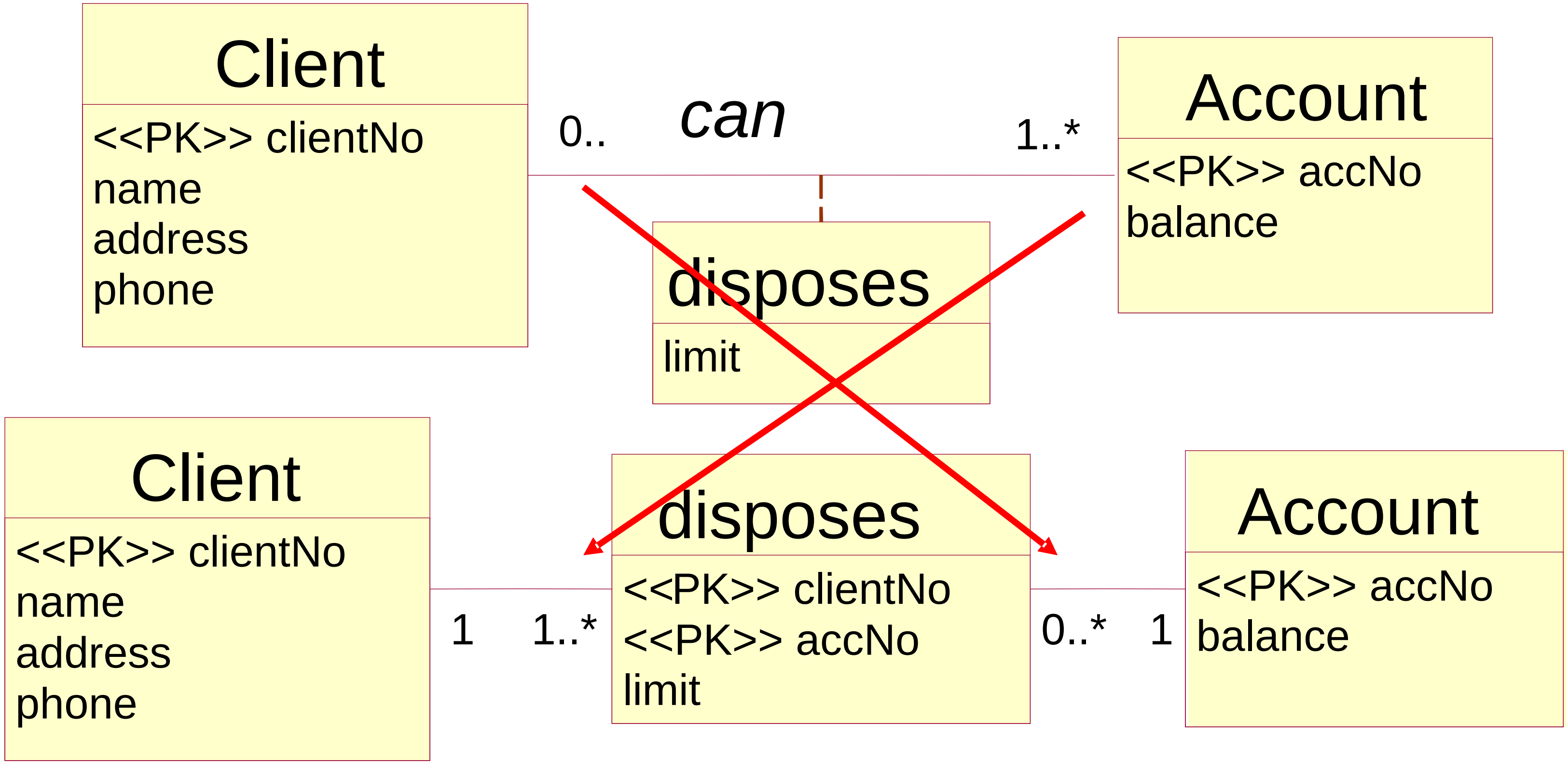


- **An entity set or an attribute?**



Rule: If a value of an attribute is important although there is no entity with the value, we should model the property as an entity set.

- **Resolving M:M relationship sets**



Bibliography

1. Silberschatz, A., Korth H.F, Sudarshan, S.:Database System Concepts. Fourth Edition. McGRAW-HILL. 2001, pp. 27 – 62.
2. T. Hawryszkiewycz, I.T.: Relational Database Design. An Introduction.Prentice Hall Inc. 1990. pp. 85 – 152.
3. Batini, C., Ceri, S., Navathe, S., B.: Conceptual Database Design. Benjamin/ Cummings. 1992. p. 460.