Example: one-to-one Relationship

Consider Bestseller between Manfs and Beers

A beer is the best-seller for 0 or 1 manufacturer

A manufacturer has exactly one best seller
Example: many-to-one/many-to-many

Notice: two relationships connect the same entity sets, but are different
Attributes on Relationships

Sometimes it is useful to attach an attribute to a relationship. Think of this attribute as a property of tuples in the relationship set.

Price is a function of both the bar and the beer, not of one alone.
**Subclasses**

*Subclass are special cases with*
- Fewer entities
- More/additional properties

*is-a hexagons indicate the subclass relationship*
- Subclasses form a tree
- No multiple inheritance

**Example**
- Ales are a kind of beer
- Not every beer is an ale, but some are
- In addition to the properties (attributes and relationships) of beers, Ales also have the attribute color
Weak Entity Sets

**Definition**

- Entity set E is said to be **weak** if in order to identify entities of E uniquely, we need to follow one or more many-to-one relationships from E and include the key of the related entities from the connected entity sets.
- In E/R diagrams a weak entity set is indicated by a double rectangle (the entity) connected by a bold type arrow to a double diamond (the relationship).

**Example**

- **name** is almost a key for football players, but there might be two with the same name.
- **number** is certainly not a key, since players on two teams could have the same number.
- But **number**, together with the **team name** related to the player should be unique.
In E/R Diagrams

- Double diamond for supporting many-to-one relationship
- Double rectangle for the weak entity set
Possible decisions

- Attribute or entity?
- Attribute to which entity?
- Key attributes?
- Entity or relationships?
- Relationships only between two entities?
- Attribute or relationship?

Difference to UML

- UML also focuses the operational part \(\rightarrow\) not relevant in E/R
- E/R is very simple and therefore very popular
Each entity is completely characterized by the values of all its attributes.
Two similar entities

Two entities with identical attributes:
- persNr: 765, 765
- firstName: "Rainer", "Armin B.
- name: "Manthey", „Cremers"
- age: 51, 58
- topic: "Informatik", "Informatik"
- rank: "C3", "C4"

Two entities with partially different values:
- persNr: ??, ??
- firstName: "Manthey", „Cremers"
- name: "Rainer", „Armin B.
- age: 51, 58
- topic: "Informatik", "Informatik"
- rank: "C3", "C4"
- Similar entities can be combined into entity types.

- „Similarity" requires at least identical attribute structure. (Attribute names and corresponding value domains are identical.)

- Entity types are graphically represented by rectangles. Attributes label the line connecting an entity type and a value domain (often symbolized by an oval):
- Domain names are often omitted (in case they are obvious or irrelevant). Instead attributes are placed inside ovals:

```
professor
```

- persNr  firstName  name  age  topic  rank

- In larger diagrams, the attribute structure is often entirely omitted in order to save space (or is written down in abbreviated form only):

```
professor
```

```
persNr, firstName, name, ...`
```
Instances and population of an entity type

- Each entity „belongs to“ at least one entity type: It is called an instance of this type.

- The set of all current instances of an entity type is called its current population.
> Multiple classification

- One and the same entity can be an instance of various entity types.

- In such a case, the attributes of the different types of this entity may well be quite different.
Classification in presence of identical attributes

- Entities with the same attributes do **not at all** have to be instances of the same type!

- Almost always additional classification criteria are required, usually **not** derivable from the attribute structure.
Key attributes

- As in the relational model (Access, SQL), there are usually one or more attributes per entity type the values of which are sufficient for uniquely identifying each instance:

  - professor
    - persNr
    - firstName
    - name
    - age
    - topic
    - rank

- (Primary) key attributes are usually underlined in an ER-diagram.
- Keys ought to be "minimal" (no attribute can be omitted).
- A distinction between primary key and other candidate keys is not made in the ER-model, even though it would be useful to do so.
 Relationships

- second main concept of the ER-model: elementary relationships between two or more entities (possibly with their own attribute values)

Each relationship is uniquely characterized by the key values of the participating entities and by the values of all relationship attributes.

However, there are no separate key attributes for relationships, as the keys of the participating entities always suffice for unique identification.
Multiple relationships

Entities may participate in various relationships (also similar ones).
Relationship types

- Similar relationships may be grouped into relationship types.

- „Similarity" requires at least identical attribute structure and identical types (and number) of participating entities.

- Relationship types are graphically denoted in the ER-model by a diamond. Attributes are written as for entity types.
Relationship types have instances, too: Individual relationships between individual entities are analogously considered instances of the corresponding R-type.
The distinction between types and instances is difficult even for specialists: Try to be precise from the very beginning!
Roles

- One and the same entity type may participate in a relationship type *more than once*, e.g.:

  - For a (syntactical) distinction between the different „forms of participation“, special designators are used, called *roles*. Roles are used as labels of the lines connecting the resp. entity and relationship type.
Extended Geo-DB: ER-diagram

- **city**
- **country**
- **river**
- **region**

**Relationships:**
- city_in_country
- capital_of_country
- city_at_river
- river_through_country
- source_of
- neighbour_country
- capital
- region_in_country

**Entity Types:**
- city
- country
- river
- region
Relationship types and mathematical relations

On the level of instances, a relation over the populations of the entity types involved is associated with each relationship type.

On the level of instances, a relation over the populations of the entity types involved is associated with each relationship type.
Functional relationships

In any cases, **at most one** entity on one „side“ of a relationship type may be associated with a particular entity on the other „side“ of the relationship, e.g.:

- Each city is in **exactly one** country.
- In each country, however, arbitrarily many cities may be situated.
Functional relationships (2)

- Mathematically, `city_in_country` is a function:
  
  \[
  \text{city_in_country: } \text{city} \rightarrow \text{country}
  \]

- More exactly: . . . a function from the population of city into the population of country.
In the ER-model, such restrictions of the admissible combinations can be expressed by means of so-called functionalities, annotations attached to the edges connecting entity and relationship types.

There are four different kinds of binary relationships expressible by means of functionalities:

- In this context, N resp. M stands for arbitrary integer values $\geq 0$.
- In the 'city_in_country'-example, an N:1-relationship is appropriate: There is exactly one country per city, but arbitrarily many cities per country ($N \geq 0$).
Functional relationships (4)

- Functionalities of type 1 : 1, 1 : N or N : 1 define partial functions where some of the instances of the types involved possibly are not related at all.

- In the „normal“ ER-model, total functions cannot be distinguished from partial ones – in extensions of the model there are additional graphical means for explicitly stating whether a function is partial or total.
Functional relationships (5)

- In a **1 : 1-relationship** each instance of one of the entity types involved is related to none or exactly one of the instances of the other entity type.

- An **N : M-relationship** can be considered the „normal case“ without restrictions on the number of participating entities.

- If **no functionalities** have been stated for a relationship type, then an implicit N : M functionality is assumed.

- Functionalities can be defined for relationships with more than two entities involved, too:

```
E₁  N  \(\times\)  M  \(\rightarrow\)  E₃
   \(\rightarrow\)  1  \(\rightarrow\)
     \(\rightarrow\)
E₂
```

\[ R: p(E₁) \times p(E₂) \rightarrow p(E₃) \]
If in an n-ary relationship several edges are marked by '1', then the resp. relationship type represents several partial functions:

\[ R^{(1)}: p(E_1) \times p(E_2) \rightarrow p(E_3) \]
\[ R^{(2)}: p(E_1) \times p(E_3) \rightarrow p(E_2) \]

... and analogously:

\[ R^{(1)}: p(E_1) \times p(E_2) \rightarrow p(E_3) \]
\[ R^{(2)}: p(E_1) \times p(E_3) \rightarrow p(E_2) \]
\[ R^{(3)}: p(E_2) \times p(E_3) \rightarrow p(E_1) \]
(Min,Max)-Notation

- The ER model offers another way for restricting how often an entity can participate in a relationship (cardinality).

- The (min,max) notation specifies an interval of possible participations in a relationship.

- An entity of type $E_1$ may be related to at least 2 and at most 7 entities of type $E_2$.
- Likewise, 1 is the minimum number and 3 is the maximum number of $E_1$ entities to which an entity from $E_2$ is related.

- Special cases:
  - Entities which may not participate in relationship $R$: $min=0$
  - Entities which arbitrarily often participate in $R$: $max=*$
Examples:

A man can be married to at most one woman and vice versa.

An airport lies in exactly one country. A country may have arbitrarily many airports (and maybe none at all).
In contrast to the afore mentioned functionalities, a \((\text{min, max})\) restriction is put to the other side of the relationship (close to the affected entity type):

**Reason:**
- **N:1** means that every instance of \(E_1\) is related to at most one instance of \(E_2\).
- \((0,1)\) for entity type \(E_1\) means that every instance of \(E_1\) can participate in \(R\) at most one time (but doesn’t have to participate because of 0).
- Here, both expressions achieve the same restriction w.r.t. allowed instances of the binary relationship \(R\). In case of ternary (or higher) relationship, the two notation lead to very different restrictions.
The (min, max) notation allows for differentiating whether a relationship represent a partial or even total function:

- Partial:
  - A minimum value set to 1 on the codomain side (functionality 1 before entity type E2) expresses whether a function is surjective or not:

- Total:
  - A minimum value set to 1 on the codomain side (functionality 1 before entity type E2) expresses whether a function is surjective or not:
Extended ER model: Generalization

- The concepts introduced so far have been contained in Chen’s original proposal throughout. Since then, however, various extensions have been proposed: Extended Entity-Relationship Model (EER-model)

- Most important extensions (as in object-oriented models):
  - formation of subtypes of entity types
  - Sub-/supertype relationships (type hierarchy)
  - Inheritance of attributes and of „participations" in R-types

- This means:
  - Special graphical notation for generalization of E-types:

\[ \text{is}_\text{a} \]
Generalization (2)

In this example, "Inheritance" means:
- Both subtypes inherit all attributes of the supertype, i.e., they "own" these attributes without explicit definition.
- Both subtypes participate in the relationship type capital_of', which has been explicitly defined for the supertype only.
Generalization always means that the populations of the subtypes are subsets of the population of the supertype.

This circumstance motivates the notion 'is_a'-relationship:

"Every country is a region."

Quantifier over instances!
in general:
This form of the 'is_a'-notation just means some form of subset formation, i.e. overlapping, incomplete subdivision.

in the example: special case disjoint generalization
(empty intersection of the populations)