Databases

Jörg Endrullis

VU University Amsterdam

Database Design

Database Design

- **formal model** of the relevant aspects of the real word
  - mini world
- the real world serves as **measure of correctness**
  - possible database states should correspond to the states of the real world

Database design is challenging:

- **Expertise**: requires expertise in the application domain
- **Flexibility**: real world often permits exceptional cases
- **Size**: database schema may become huge

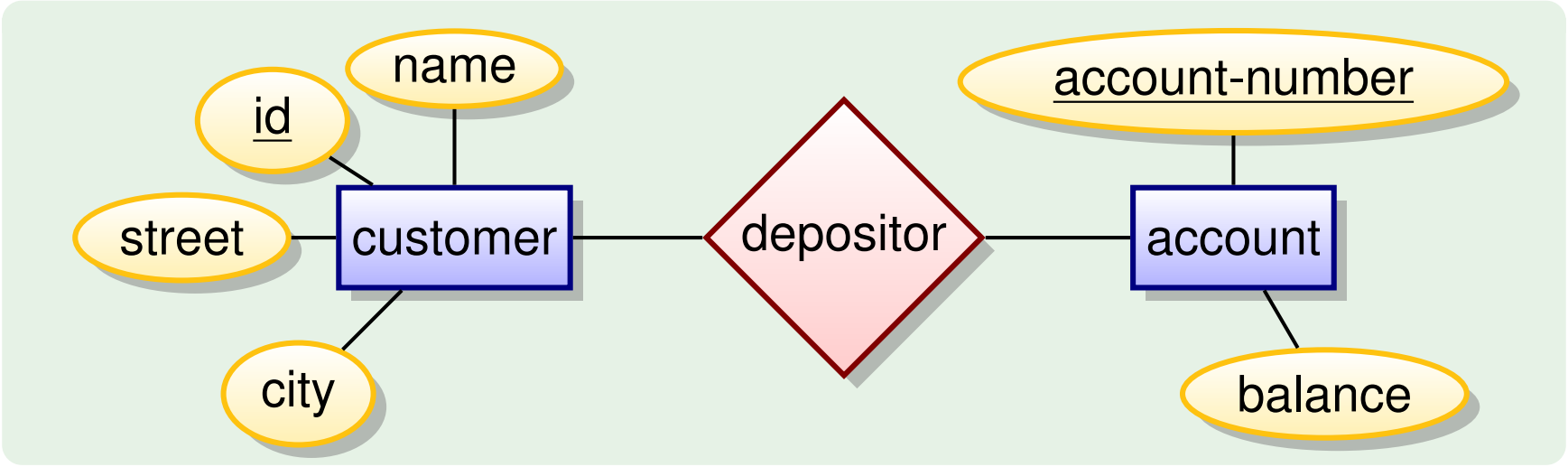
Database Design

Due to the complexity, the design is a multi-step process. . .

Three Phases of Database Design

- **Conceptual Database Design**
  - what information do we store
  - how are the information elements related to each other
  - what are the constraints?
  - e.g. E/R model or UML model
- **Logical Database Design**
  - transformation of the conceptual schema into the schema supported by the database
  - e.g. relational model
- **Physical Database Design**
  - design indexes, table distribution, buffer sizes,. . .
  - to maximise performance of the final system

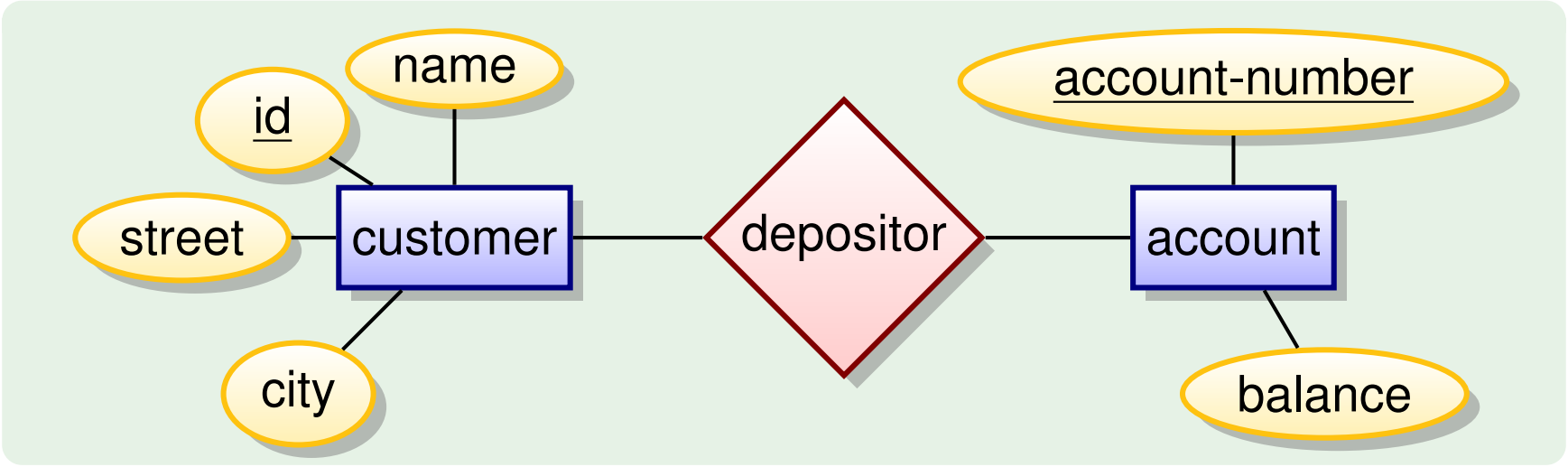
Entity-Relationship Model



The three main ingredients of entity-relationship diagrams are:

- **Entity sets**
- **Attributes**
- **Relationship sets**

Entity-Relationship Model



- **Rectangles** represent entity sets
- **Ellipses** represent attributes
  - **Double ellipses** represent multi-valued attributes
  - **Dashed ellipses** denote derived attributes
- **Diamonds** represent relationship sets
- **Lines** link attributes and relationship sets to entity sets
- **Underline** indicates primary key attributes

Entity Sets

- **entity** is an abstract object
  - e.g.: specific person, company, event
- entities have **attributes**
  - e.g.: people have names and addresses
- **entity set** is a collection of similar entities
  - similar = sharing the same properties (attributes)
  - e.g.: set of all persons, companies, trees, holidays

Comparison with object-oriented programming:

- entity $\approx$ object
- entity set $\approx$ class

Important difference: the **E/R model is static**

- models structure of the data, not the operations
- no methods/functions associated to entity sets

Attributes

An **entity set is represented by a set of attributes**, that is, descriptive properties possessed by all entities of the entity set.

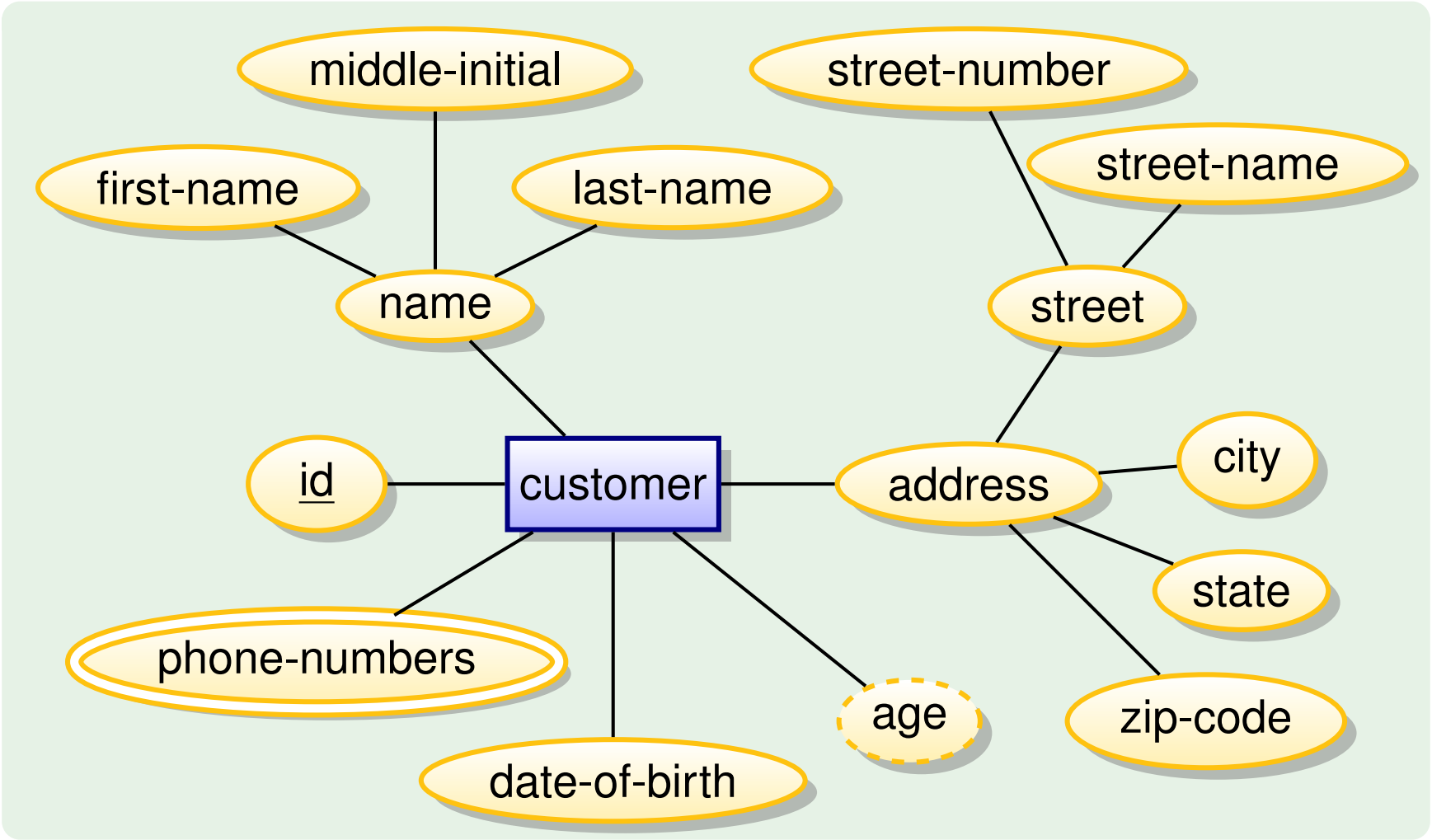
```
CUSTOMER = (ID, NAME, STREET, CITY)
LOAN = (LOAN-NUMBER, AMOUNT)
```

The **domain** is the set of permitted values for each attribute.

Attribute types

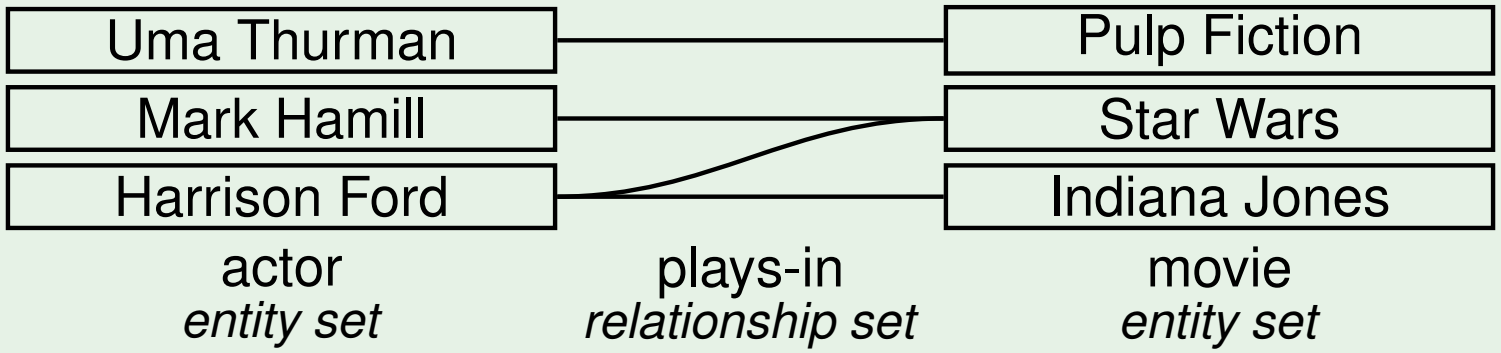
- **simple** and **composite** attributes
  - e.g. street is composed of street name and number
- **single-valued** and **multi-valued** attributes
  - e.g. single-valued: age of a person
  - e.g. multi-valued: person can have multiple phone numbers
- **derived attributes**
  - can be computed from other attributes
  - e.g. age can be computed given the date of birth

Attributes



- *name*, *address* and *street* are composite attributes
- *phone numbers* is a multi-valued attribute
- *age* is a derived attribute (derived from *date-of-birth*)

Relationship Sets

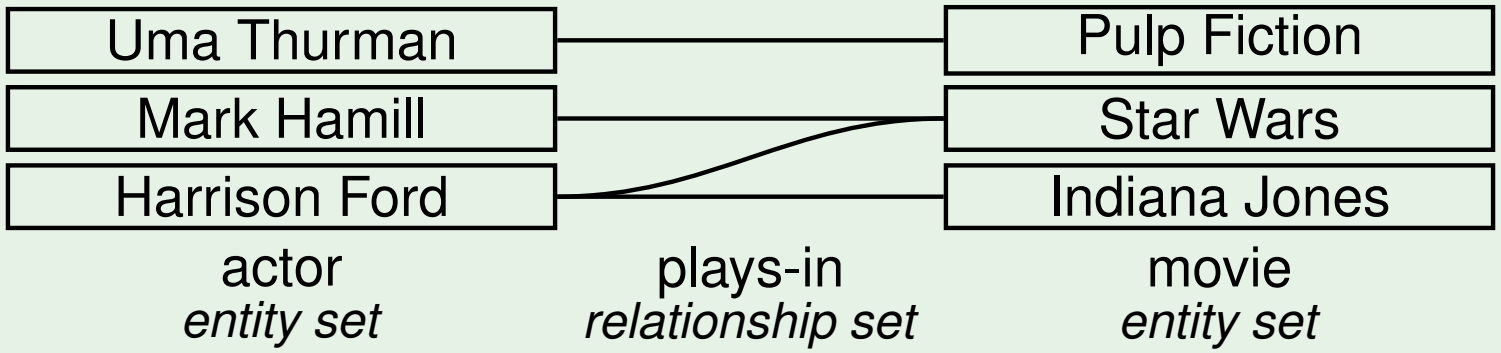


A **relationship** is an association among several entities.

That is, a **relationship** is a tuple $(e_1, e_2, \ldots, e_n)$ of entities.

- (Mark Hamill, Star Wars) is a relationship
- (Harrison Ford, Indiana Jones) is a relationship

Relationship Sets



A **relationship set** is a set of relationships of the same kind.

That is, a **relationship set** is a set of tuples $(e_1, e_2, \ldots, e_n)$ where $e_1 \in E_1, \ldots, e_n \in E_n$ are from entity sets $E_1, \ldots, E_n$.

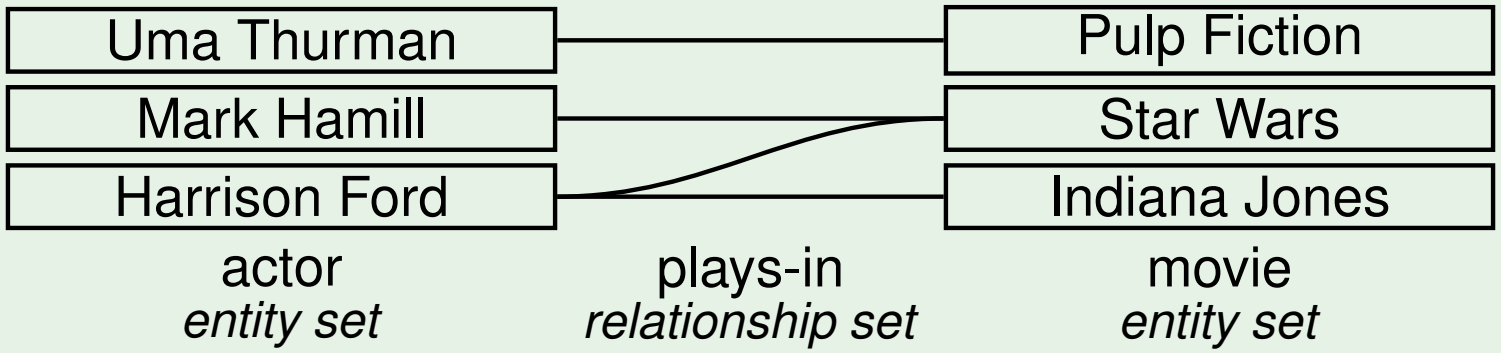
Example of a relationship set

$\{$ (Uma Thurman, Pulp Fiction), (Mark Hamill, Star Wars), (Harrison Ford, Star Wars), (Harrison Ford, Indiana Jones) $\}$

The elements of a relationship set are relationships:

- (Mark Hamill, Star Wars) is a relationship

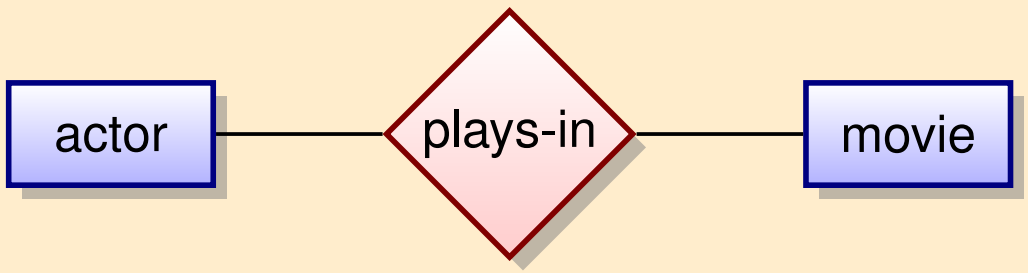
Relationship Sets



A **relationship set** is a set of relationships of the same kind.

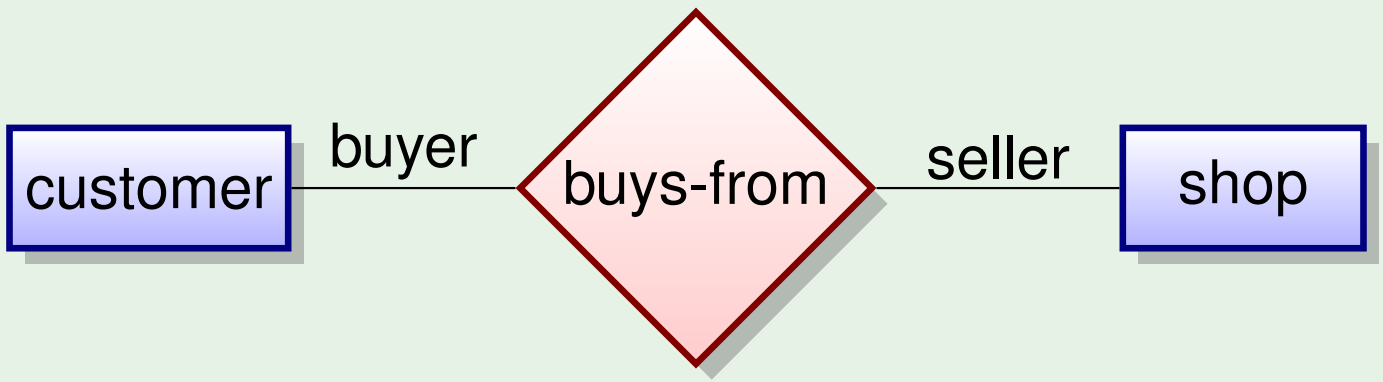
That is, a **relationship set** is a set of tuples $(e_1, e_2, \ldots, e_n)$ where $e_1 \in E_1, \ldots, e_n \in E_n$ are from entity sets $E_1, \ldots, E_n$.

A relationship set *plays-in* between entity sets *actor* and *movie* is indicated as follows in E/R models:



Relationship Sets and Role Names

The relationship set connections can be annotated with **role indicators**.

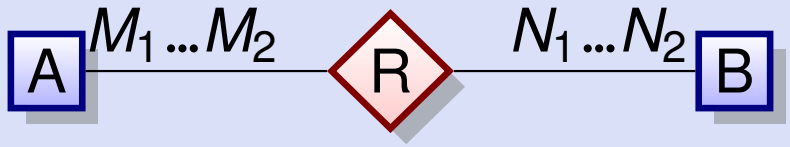


Role indicators improve readability!

Cardinality Limits

Cardinality limits express the number of entities to which another entity can be associated via a relationship set.

There are many different notations. **We use the UML notation:**

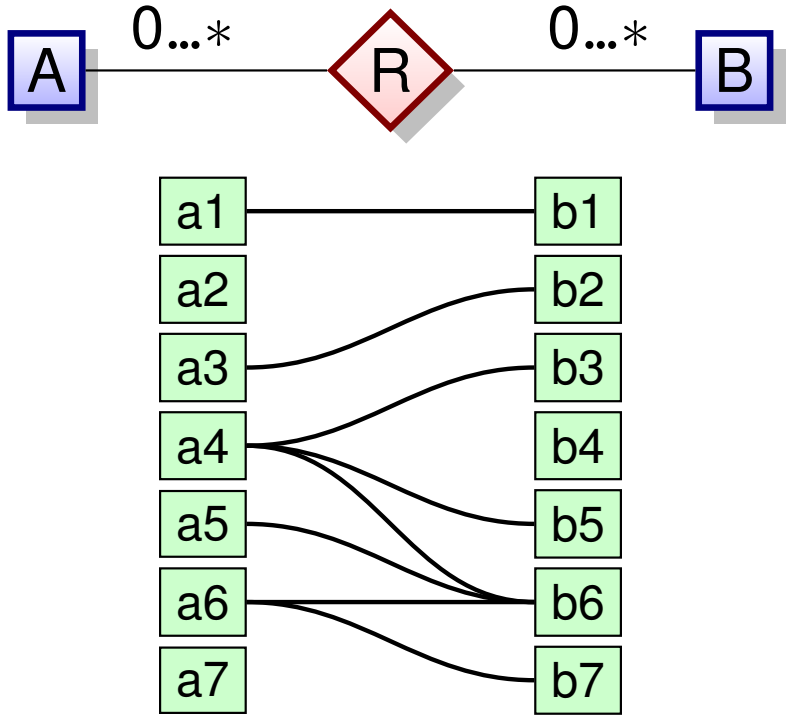


- Every entity *a* from *A* is connected to at least N_1, and at most N_2 entities in *B*.
- Every entity *b* from *B* is connected to at least M_1, and at most M_2 entities in *A*.

Typical cardinality constraints

0…1 = zero or one	0…∗ = any number
1…1 = precisely one	1…∗ = at least one

Cardinality Limits: Many-to-Many

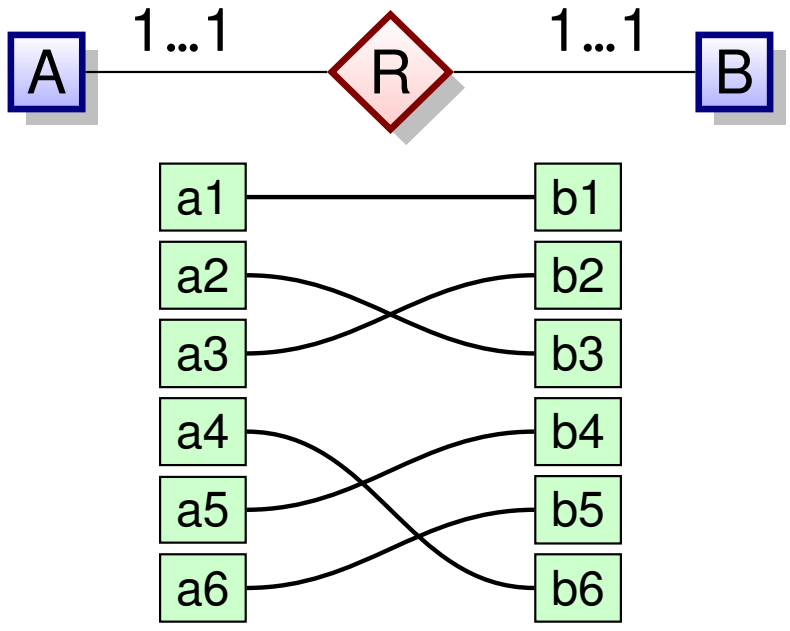


This describes a **many-to-many** relationship set:

- the entities may be connected arbitrarily
- every *a* in *A* can be linked to an arbitrary number of *B*'s
- every *b* in *B* can be linked to an arbitrary number of *A*'s

If the cardinalities are not given, the **default is many-to-many**.

Cardinality Limits: One-to-One

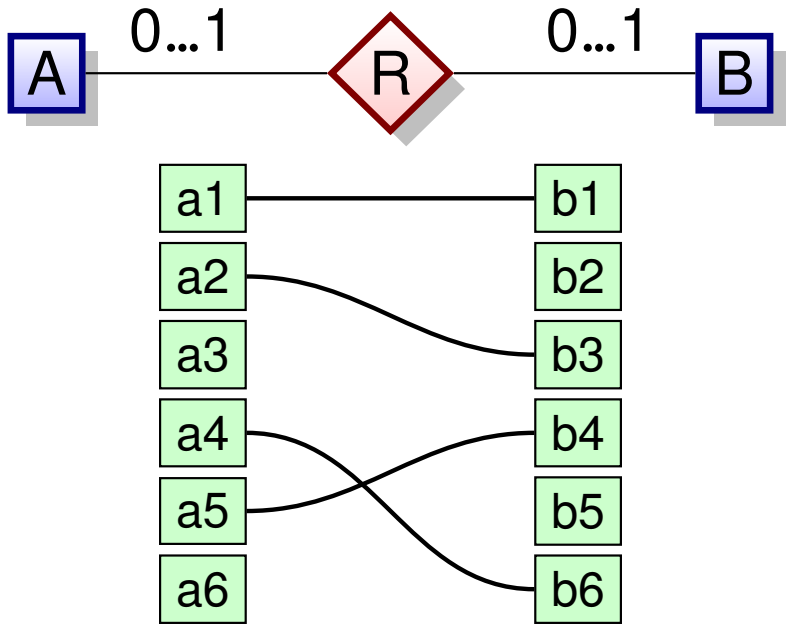


This describes a **one-to-one** relationship set:

- every *a* in *A* is connected to precisely one *b* in *B*
- every *b* in *B* is connected to precisely one *a* in *A*

Note that this corresponds to a bijective function from *A* to *B*.

Cardinality Limits: Zero or One-to-Zero or One

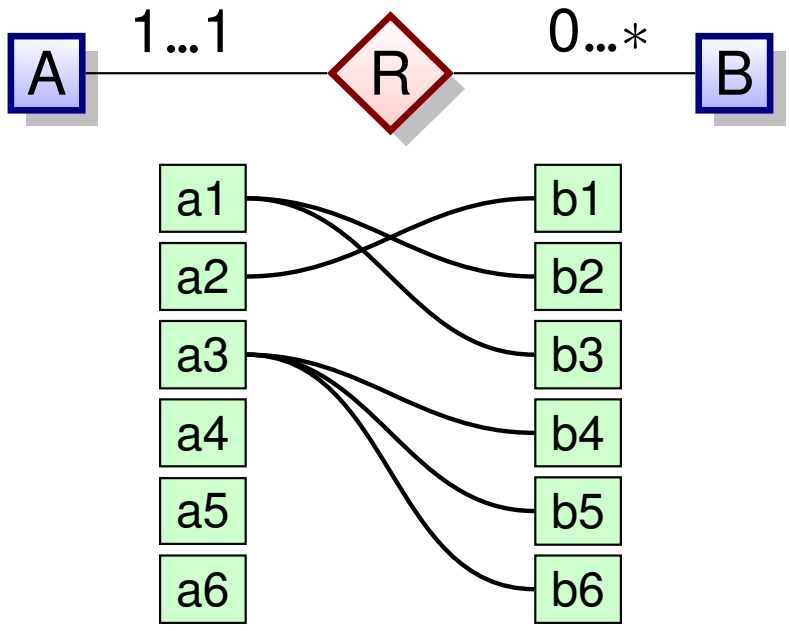


This describes a **zero or one-to-zero or one** relationship set:

- every *a* in *A* is connected to at most one (= 0 or 1) *b* in *B*
- every *b* in *B* is connected to at most one (= 0 or 1) *a* in *A*

Confusingly, this is sometimes also called one-to-one.

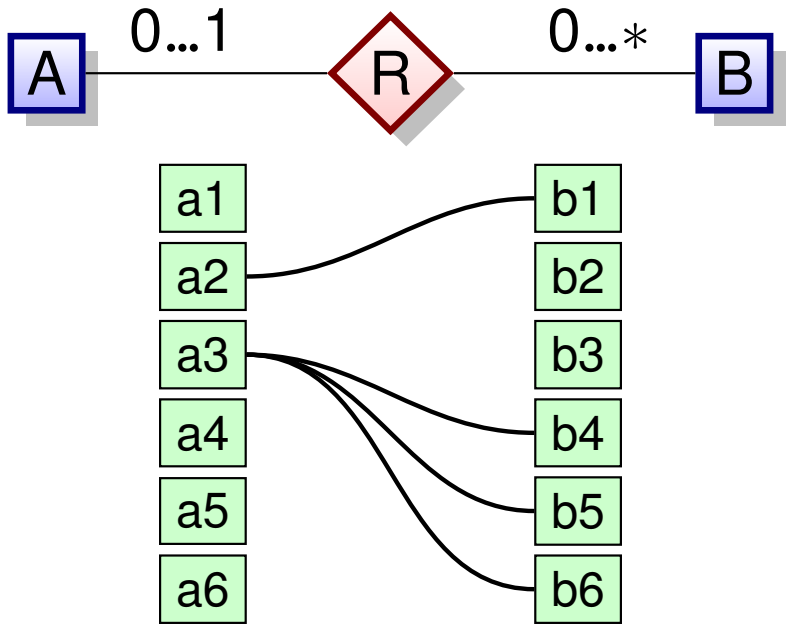
Cardinality Limits: One-to-Many



This describes a **one-to-many** relationship set:

- every *a* in *A* is related to an arbitrary number *b*'s in *B*
- every *b* in *B* is connected to precisely one *a* in *A*

Cardinality Limits: Zero or One-to-Many

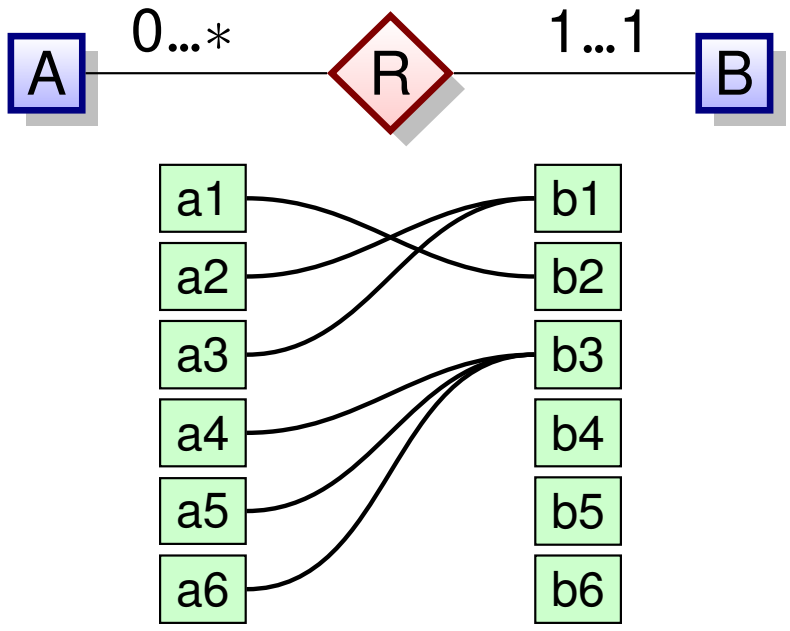


This describes a **zero or one-to-many** relationship set:.

- every *a* in *A* is related to an arbitrary number *b*'s in *B*
- every *b* in *B* is connected to at most one *a* in *A*

Confusingly, this is sometimes also called one-to-many.

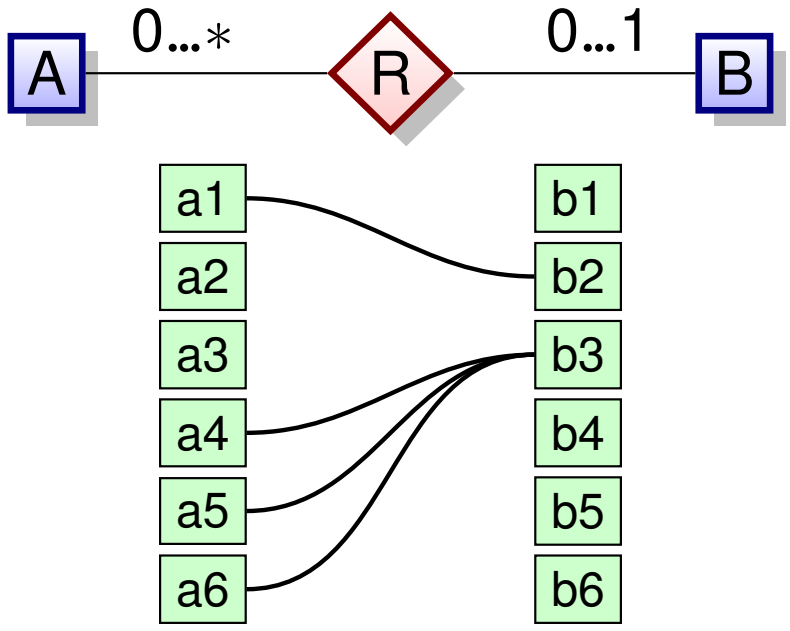
Cardinality Limits: Many-to-One



This describes a **many-to-one** relationship set:

- every *b* in *B* is related to an arbitrary number of *a*'s in *A*
- every *a* in *A* is connected to precisely one *b* in *B*

Cardinality Limits: Many-to-Zero or One



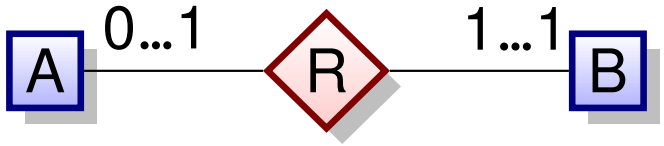
This describes a **many-to-zero or one** relationship set:

- every *b* in *B* is related to an arbitrary number of *a*'s in *A*
- every *a* in *A* is connected to at most one *b* in *B*

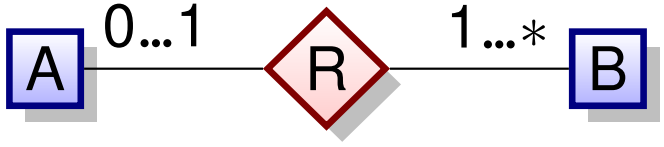
Confusingly, this is sometimes also called many-to-one.

Express the following Cardinality Limits

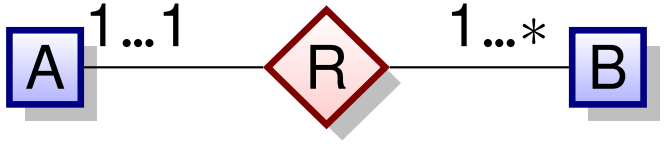
Every *a* in *A* is connected to precisely one *b* in *B*,
and every *b* in *B* is connected to at most one *a* in *A*.



Every *a* in *A* is connected to one or more *b* in *B*,
and every *b* in *B* is connected to at most one *a* in *A*.



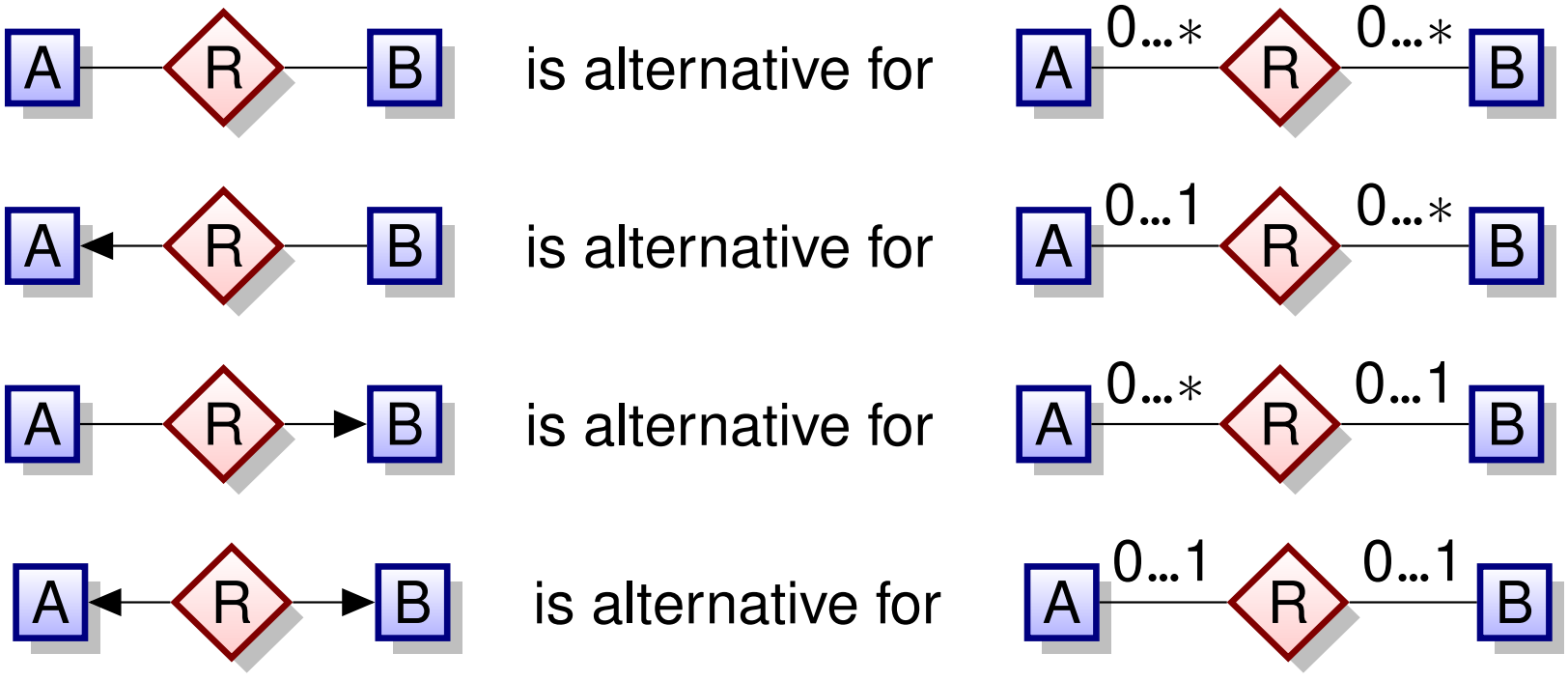
Every *a* in *A* is connected to one or more *b* in *B*,
and every *b* in *B* is connected to precisely one *a* in *A*.



Cardinality Limits: Alternative Notations

There are many different notations for E/R models.

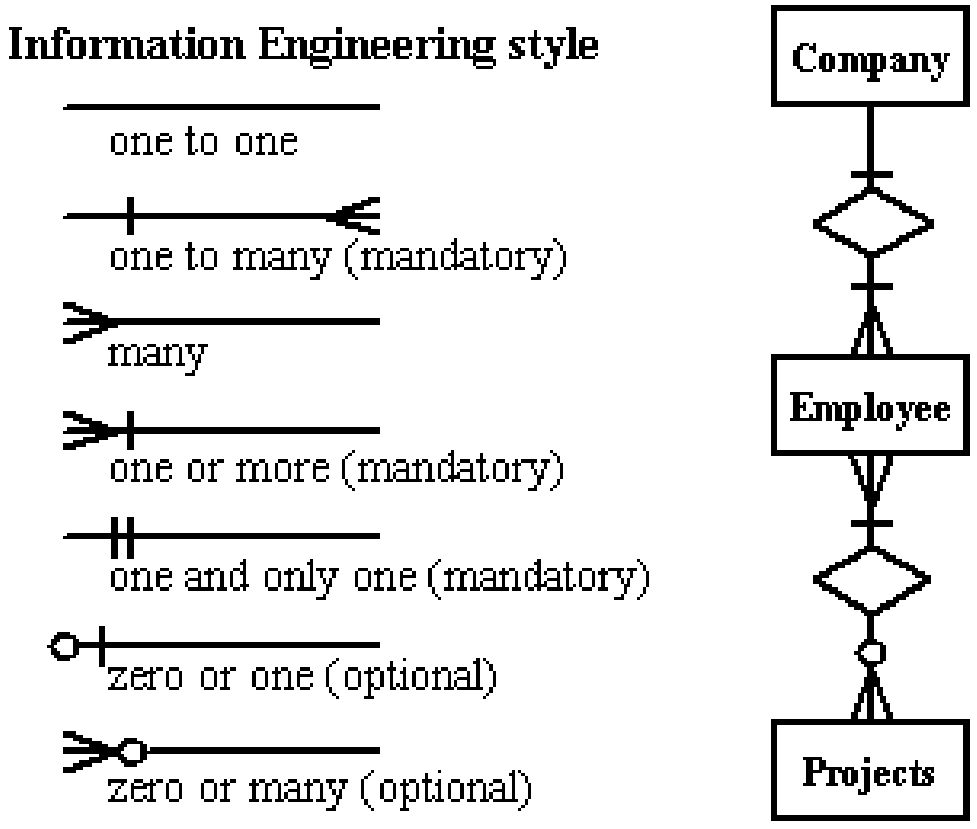
For example:



Cardinality Limits: Alternative Notations

There are many different notations for E/R models.

For example: Information engineering style

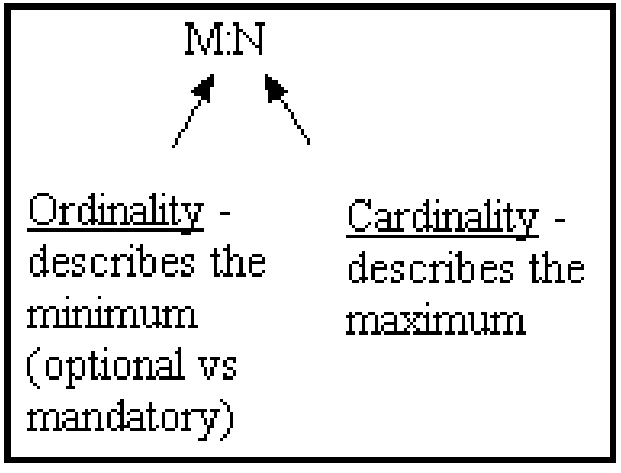


Cardinality Limits: Alternative Notations

There are many different notations for E/R models.

For example: Chen style

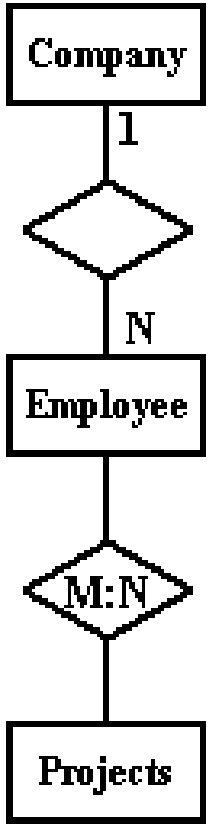
Chen style



1:N (n=0,1,2,3...)
one to zero or more

M:N (m and n=0,1,2,3...)
zero or more to zero or more
(many to many)

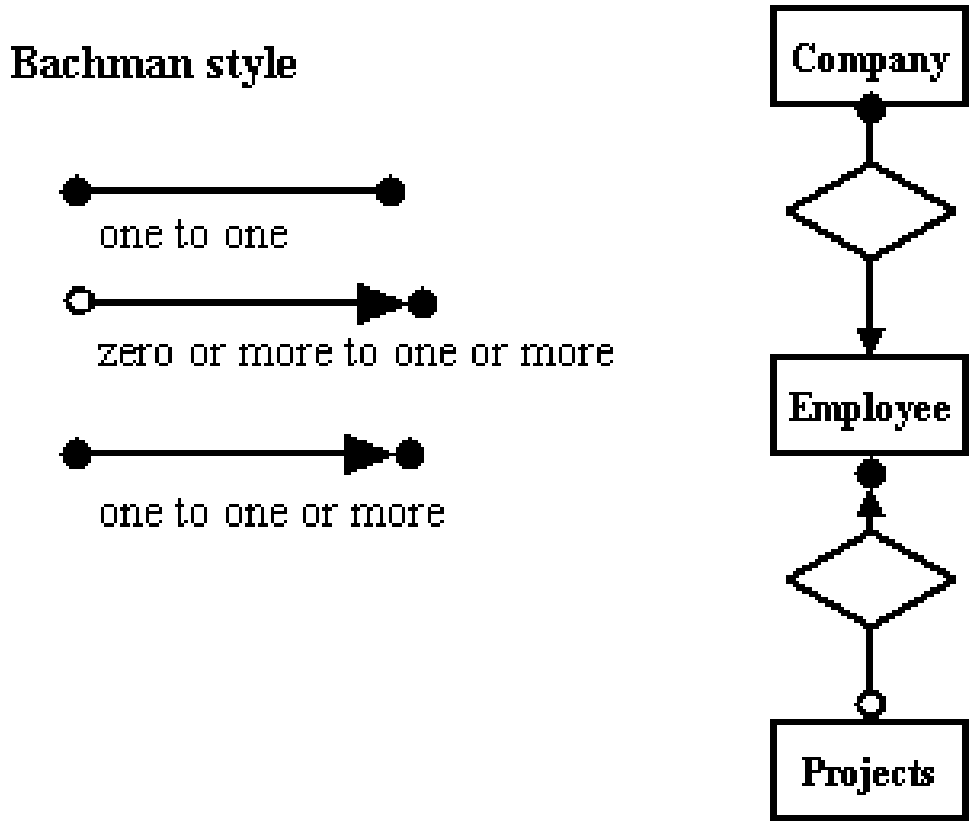
1:1
one to one



Cardinality Limits: Alternative Notations

There are many different notations for E/R models.

For example: Bachman style



Cardinality Limits: Alternative Notations

There are many different notations for E/R models.

For example: Martin style

Martin style

1 - one, and only one (mandatory)

* - many (zero or more - optional)

1...* - one or more (mandatory)

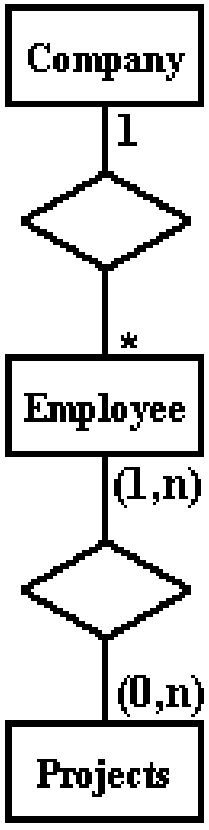
0...1 - zero or one (optional)

(0,1) - zero or one (optional)

(1,n) -one or more (mandatory)

(0,n) - zero or more (optional)

(1,1) - one and only one (mandatory)

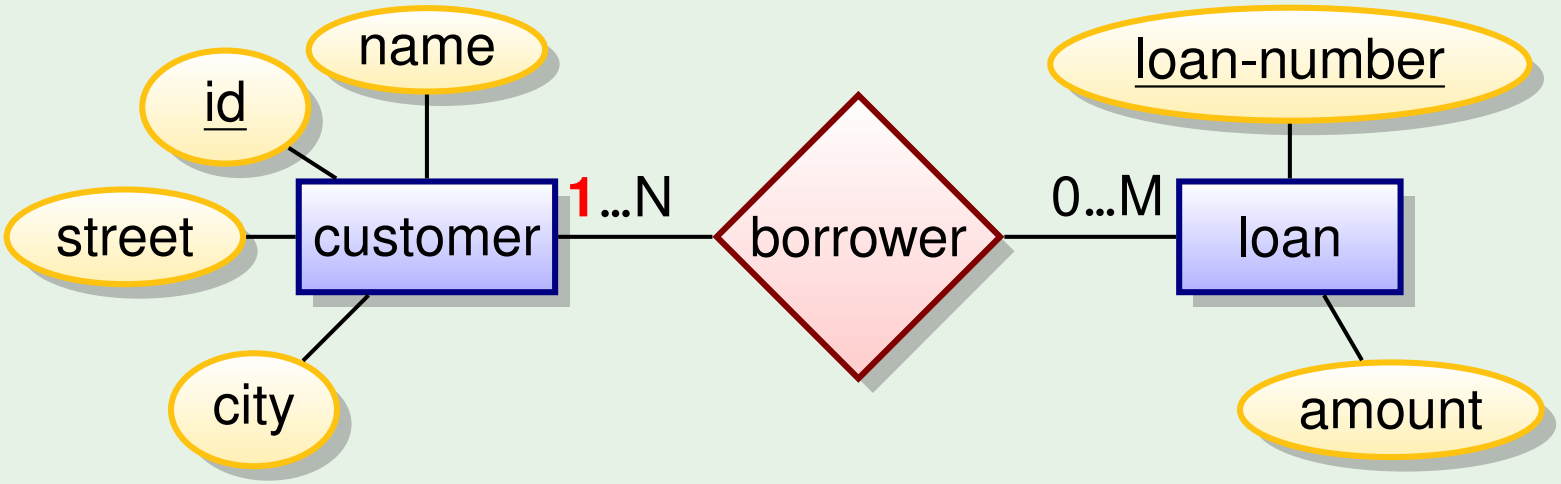


Total Participation

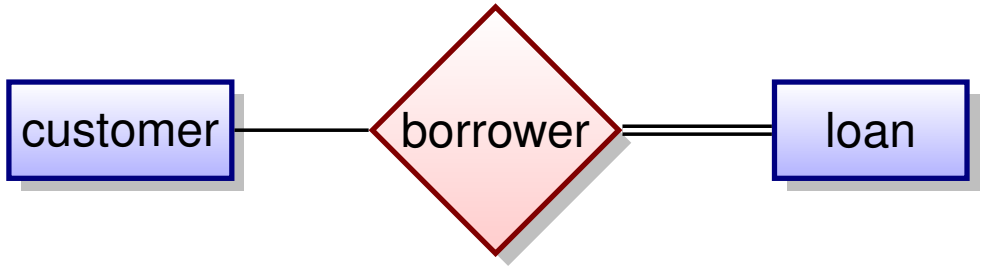
Total participation means that every entity in the entity set participates in at least one relationship in the relationship set.

- e.g. every loan must be belong to at least one customer

Partial participation means that entities may not participate in any relationship in the set.



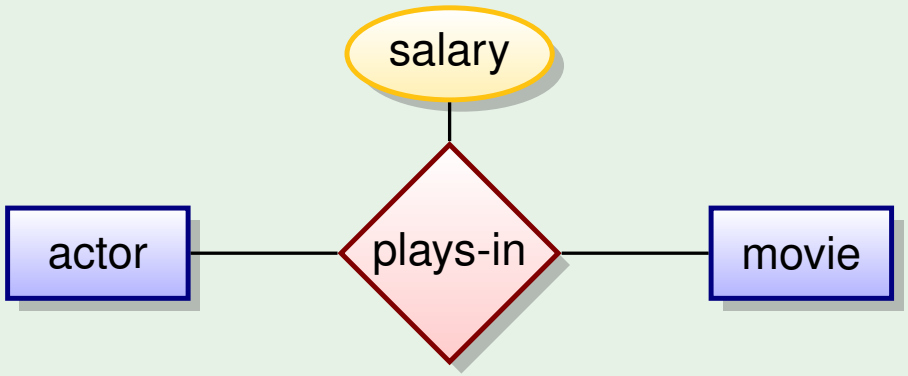
Alternative notation:

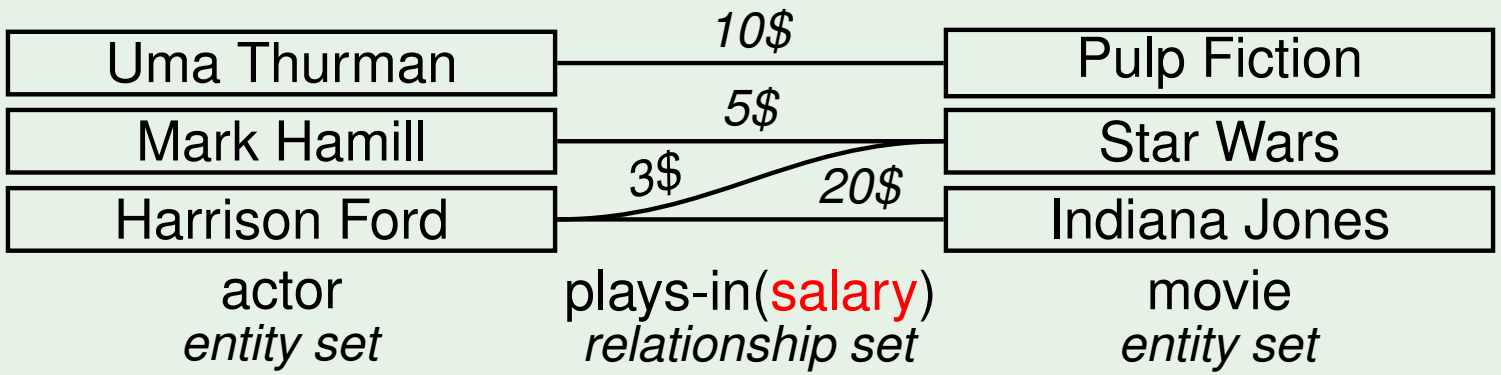


Relationship Sets with Attributes

An **attribute** can also be property of a relationship set.

The *plays-in* relationship set between the entity sets *actor* and *movie* may have the attribute *salary*.

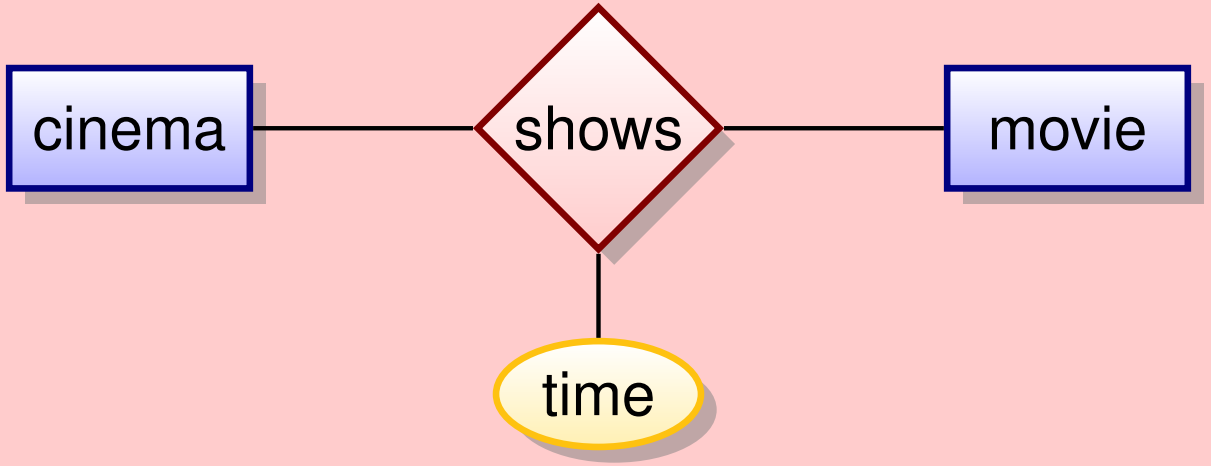




The value of the relationship attributes is **functionally determined** by the relationship $(e_1, \ldots, e_n)$.

Relationship Sets with Attributes

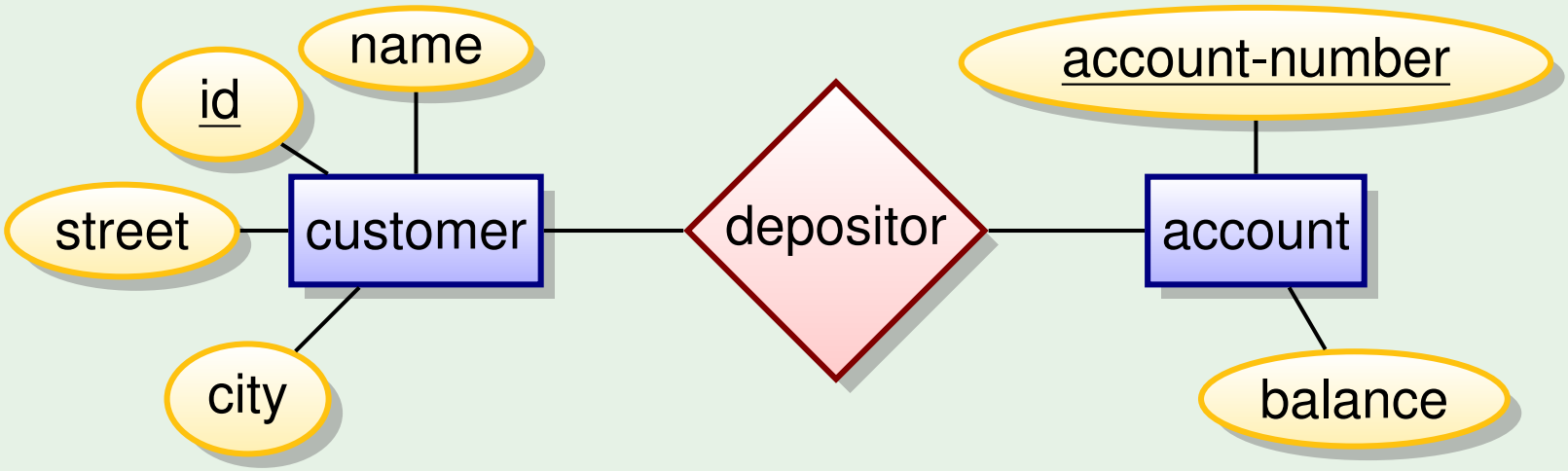
Consequences of the Semantics



Suppose a cinema shows a movie twice a day (at 3pm and 6pm).

Can this information be stored in the given schema?

Cardinalities affect the E/R Design



Assume that we want to record the date of the last access of a customer to an account. We call this attribute *access-date*.

If the relation from customer to account is **many-to-many**:

- then *access-date* must be an attribute of *depositor*

If the relation from customer to account is **one-to-many**:

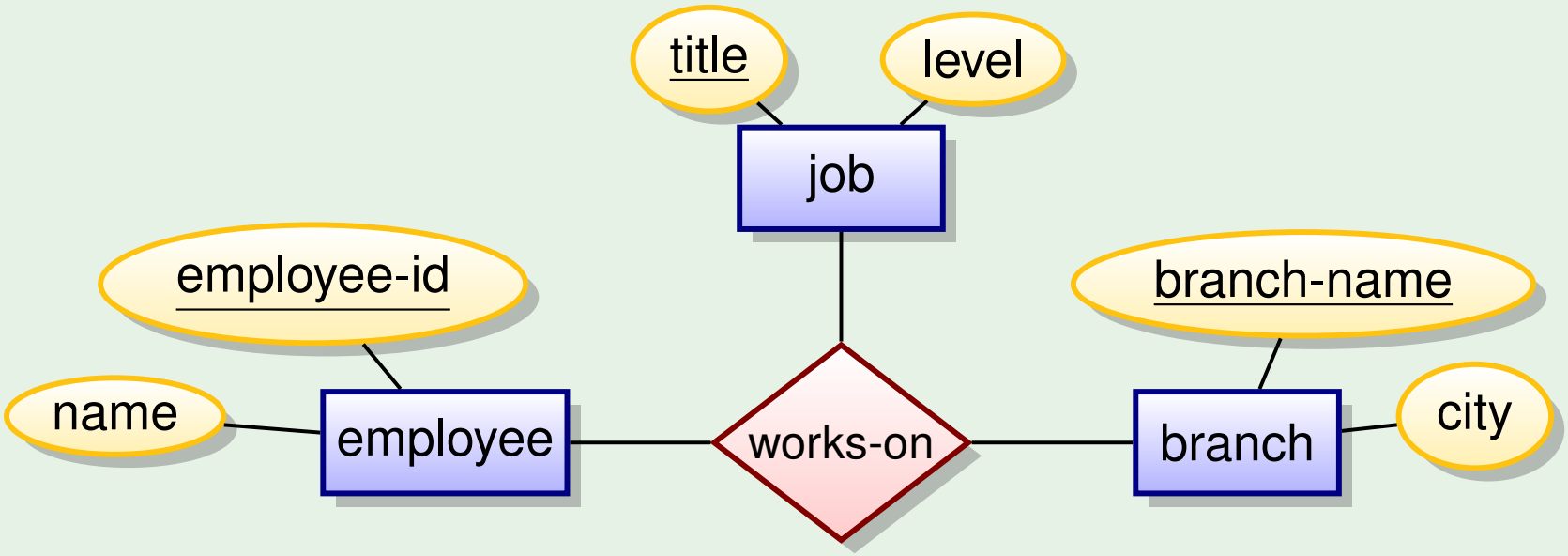
- then *access-date* can be an attribute of *account*

Degree of a Relationship Set

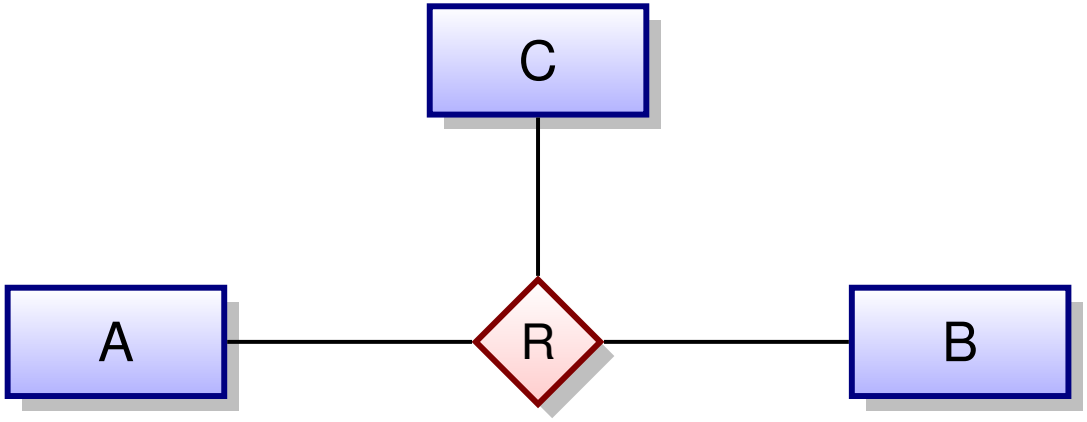
The **degree** of a relationship set refers to the number of entity sets participating in the relationship.

- relationship sets of degree 2 are called **binary**
- relationship sets of degree 3 are called **ternary**

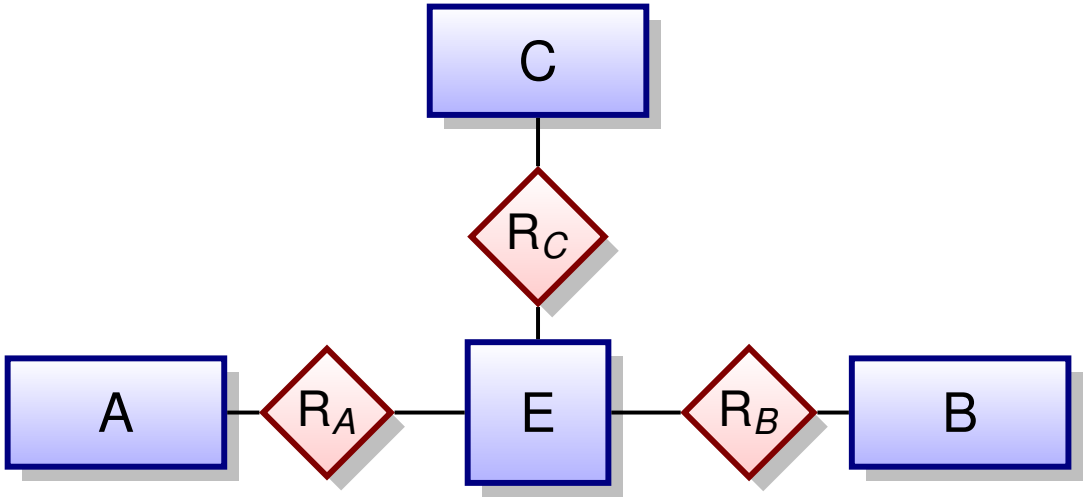
Example for a ternary relationship set *works-on*: an *employee* might work on different *jobs* at different *branches* of a company.



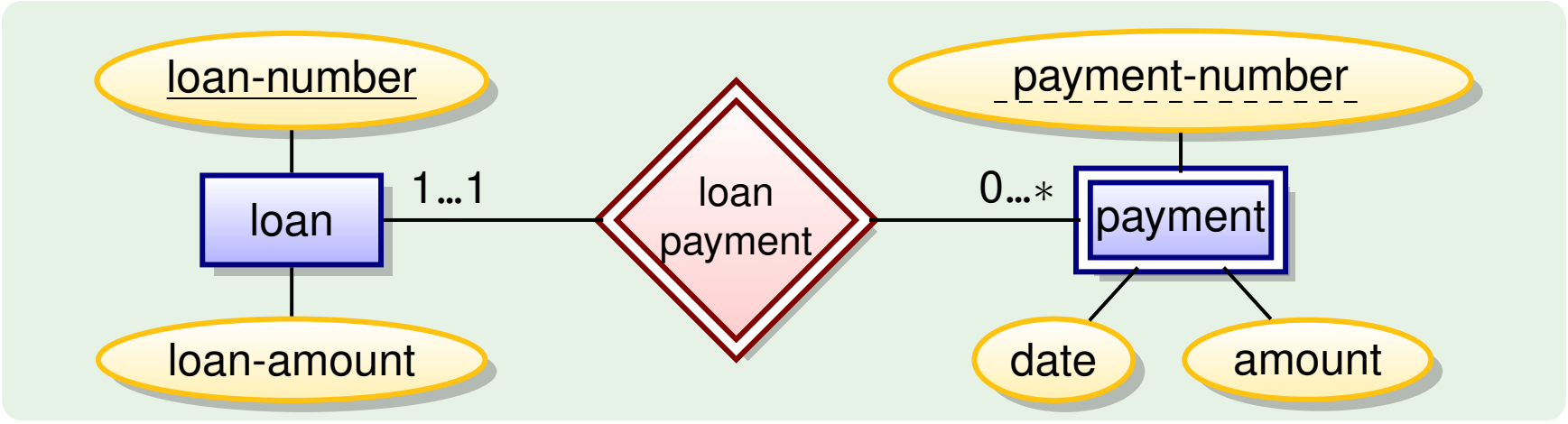
Degree of a Relationship Set



Non-binary relationship sets can be represented using binary ones by creating an artificial entity set.



Weak Entity Sets

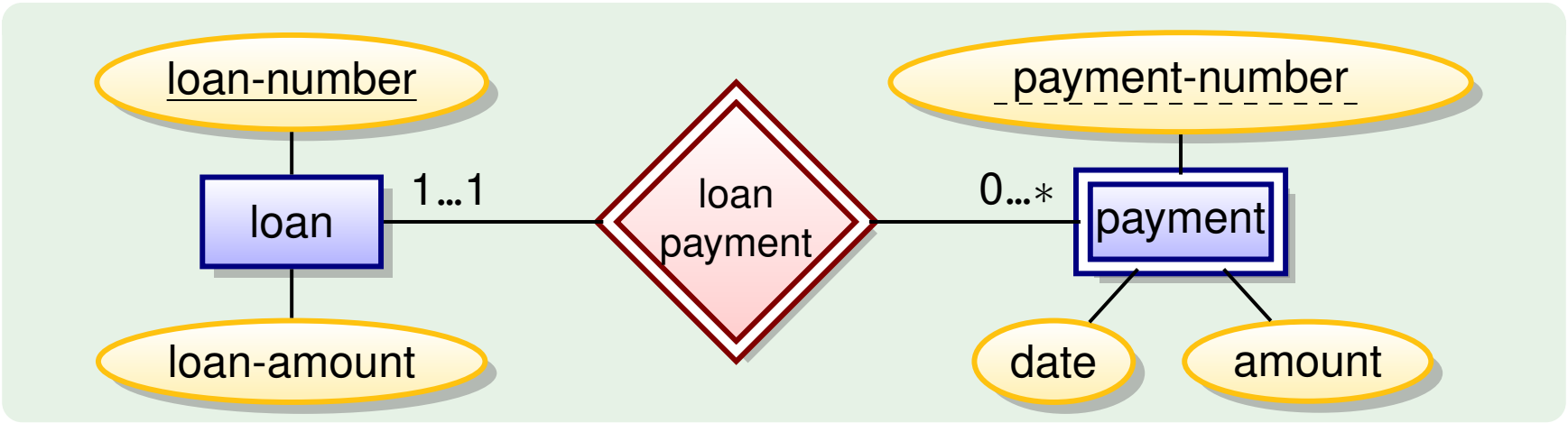


There can be multiple payments with equal _payment-number_

- the _payment-number_ is **not a key**
- payments must always be associated to precisely one loan
- the _payment-number_ identifies a payment uniquely only in combination with the loan-number of the associated loan

In other words, the **discriminator** _payment-number_ is unique among all payments for a certain loan.

Weak Entity Sets



A **weak entity set** is an entity set without a primary key.

- The existence of a weak entity set depends on the existence of an **identifying entity set**.
- There must be a total, one-to-many relationship set from the identifying entity set to the weak entity set. This **identifying relationship** is depicted by a double diamond.
- The **discriminator** is a partial key, it distinguishes the weak entity only in combination with the identifying entity.
- Primary key of the weak entity set is a combination of the discriminator and primary key of the identifying entity set.

Weak Entity Sets

Modelling with Weak Entity Sets

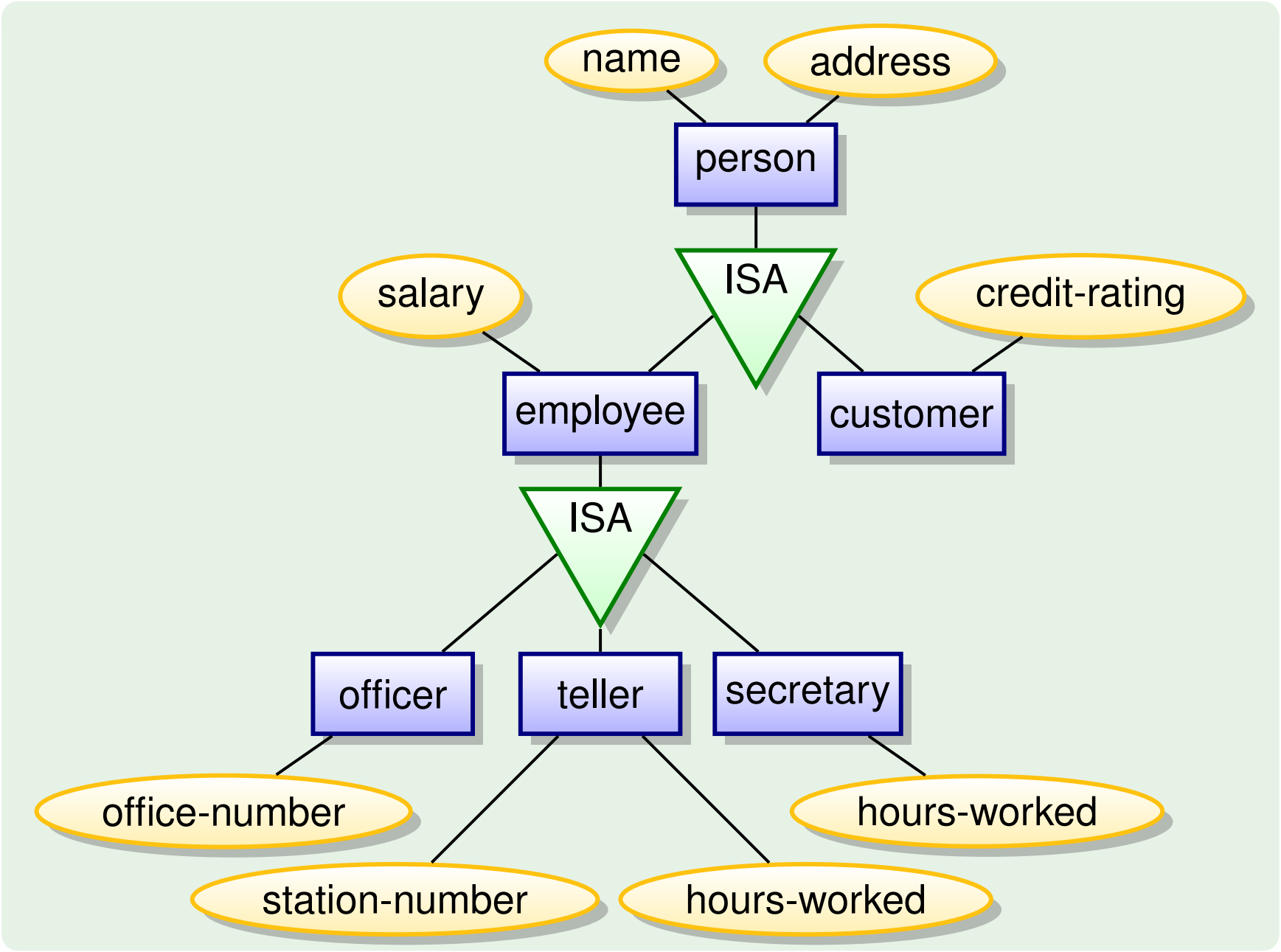
Model a set of online quizzes (multiple-choice tests).

- each quiz is identified by a title
- each question within a quiz is numbered
- each possible answer to a question is referenced by a letter
- for each question and answer the associated text is stored
- answers are classified into correct and incorrect ones

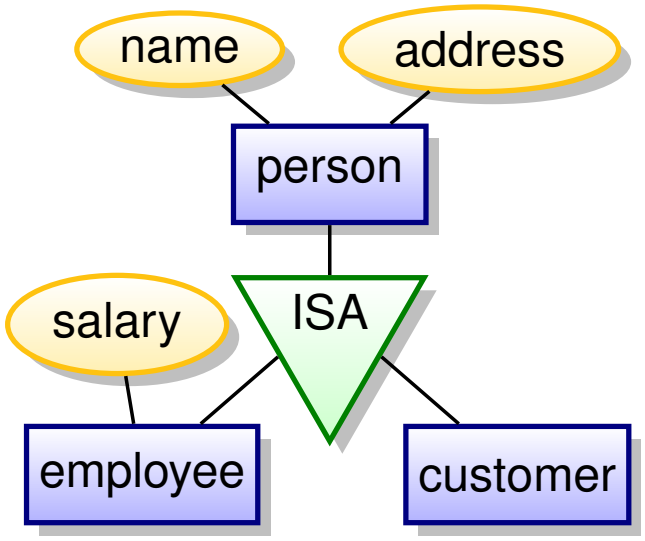
Develop an E/R diagram.

What is the complete key for each of the entity sets?

IS-A, ‘Inheritance’



IS-A, ‘Inheritance’



Lower-level entity sets are subgroups of the of higher-level entity sets:

- e.g. an employee ‘is a’ person

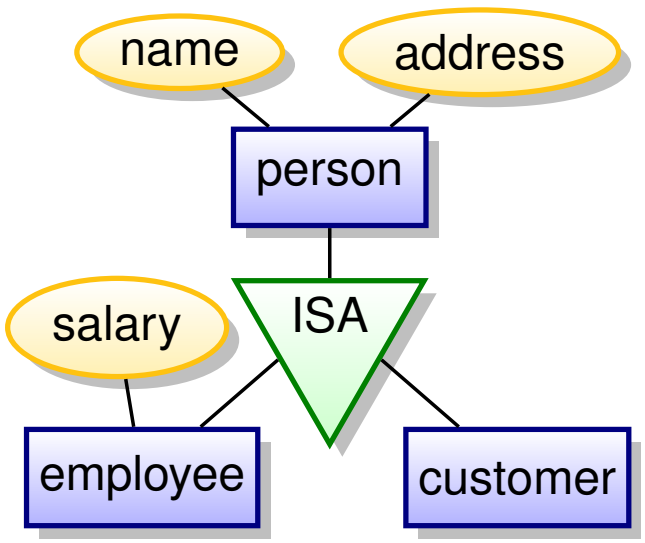
Lower-level entity sets **inherit all attributes and relationships** of the higher-level entity sets.

- e.g. an employee has attributes *name*, *address* and *salary*

Design Principle: Specialisation

- top-down design process
- identify subgroups within an entity set
- these subgroups become lower-level entity sets which may have attributes or participate in relationships that do not apply to the higher-level entity sets

IS-A, ‘Inheritance’



Lower-level entity sets are subgroups of the of higher-level entity sets:

- e.g. an employee ‘is a’ person

Lower-level entity sets **inherit all attributes and relationships** of the higher-level entity sets.

- e.g. an employee has attributes *name*, *address* and *salary*

Design Principle: Generalisation

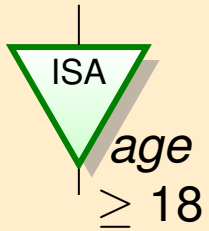
- bottom-up design process
- combine a number of entity sets that share common features into a higher-level entity set

- specialisation and generalisation are both ‘is a’-relations

IS-A, 'Inheritance'

Membership constraints

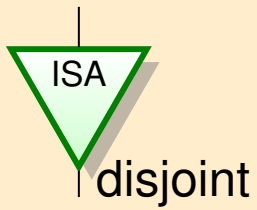
- **value-based**: assigns an entity to a specific subclass based on attribute values
  e.g. a *person* of age ≥ 18 is an *adult*



- default is **user-defined**: manual assignment to subclasses

Disjointness constraints

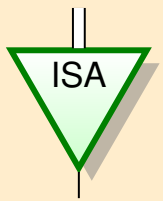
- **disjoint**: an entity can belong to at most one subclass; e.g. a *fruit* can be an *apple* or a *pear*, but not both



- default is **overlapping**: can belong to multiple subclasses

Completeness constraints

- **total specialisation (generalisation)** constraint: each superclass entity must belong to a subclass; e.g. a *person* is either a *minor* or an *adult*

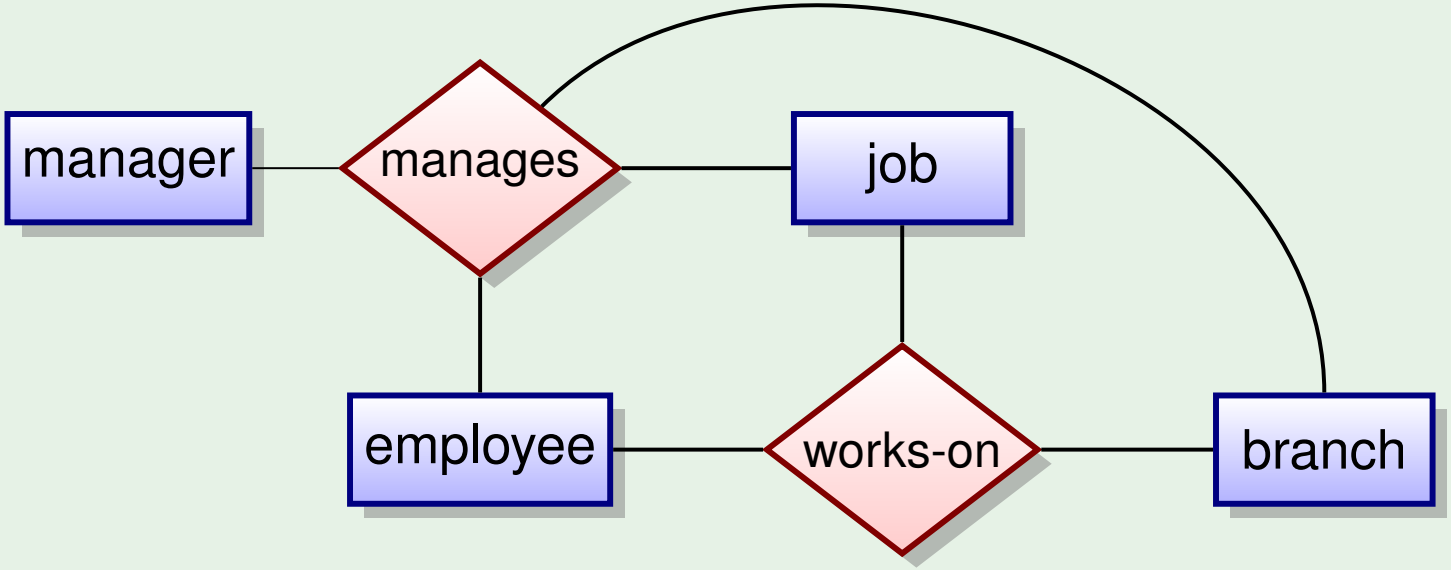


Aggregation

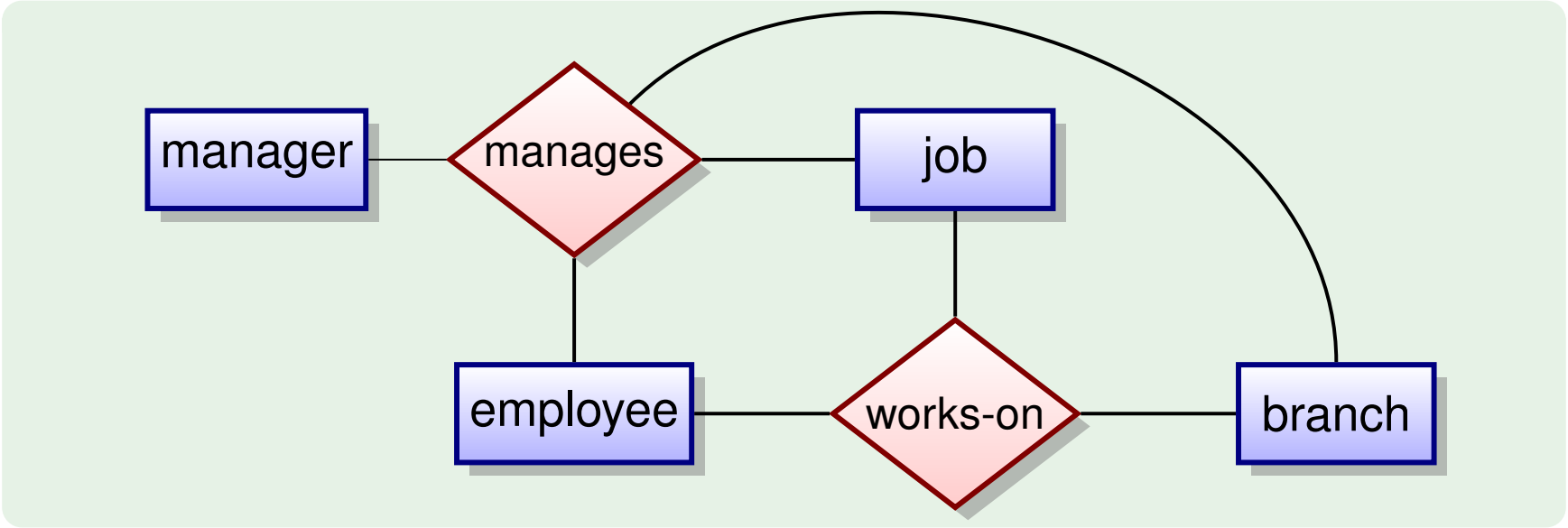
Consider the *works-on* relation we have seen before.

We now want to express that a task performed by an employee might have a manager assigned to it.

- E/R model has no relations between relations



Aggregation



However, this design is not good:

- **does not capture:** every *manages* relationship corresponds to a *works-on* relationship;
- information is represented **redundant/overlapping**;
- **we cannot discard the works-on relationship set:** some *works-on* relationships may not correspond to any *manages* relationship.

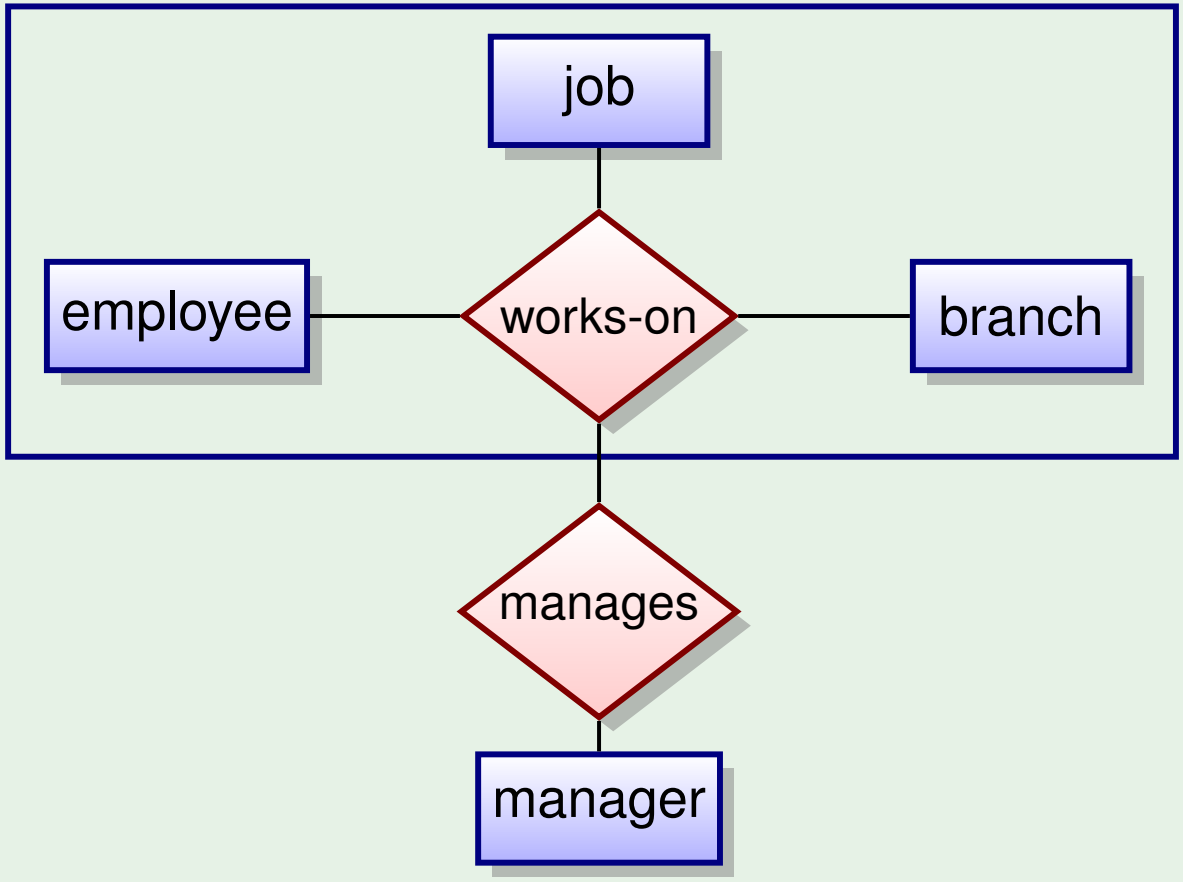
The solution is to eliminate redundancy using **aggregation**!

Aggregation

Aggregation:

- treat relationship set as an abstract entry

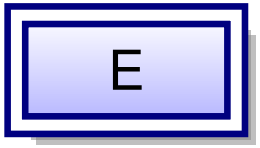
  abstraction of a relationship into a new entry
- allows relations between relations



Entity-relationship Models Summary

entity set

attribute



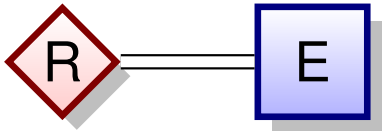
weak entity set

multi-valued attribute

relationship set

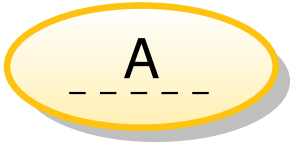
derived attribute

identifying
relationship set
for weak entity set



total participation
of entity set
in relationship

primary key



discriminating
attribute of
weak entity set

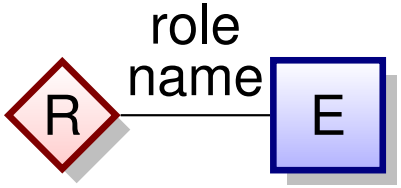
Entity-relationship Models Summary

many-to-many relationship

many-to-one relationship

one-to-one relationship

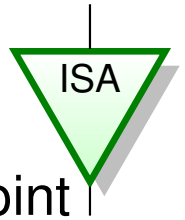
cardinality limits



role indicator

ISA (generalisation or specialisation)

total generalisation

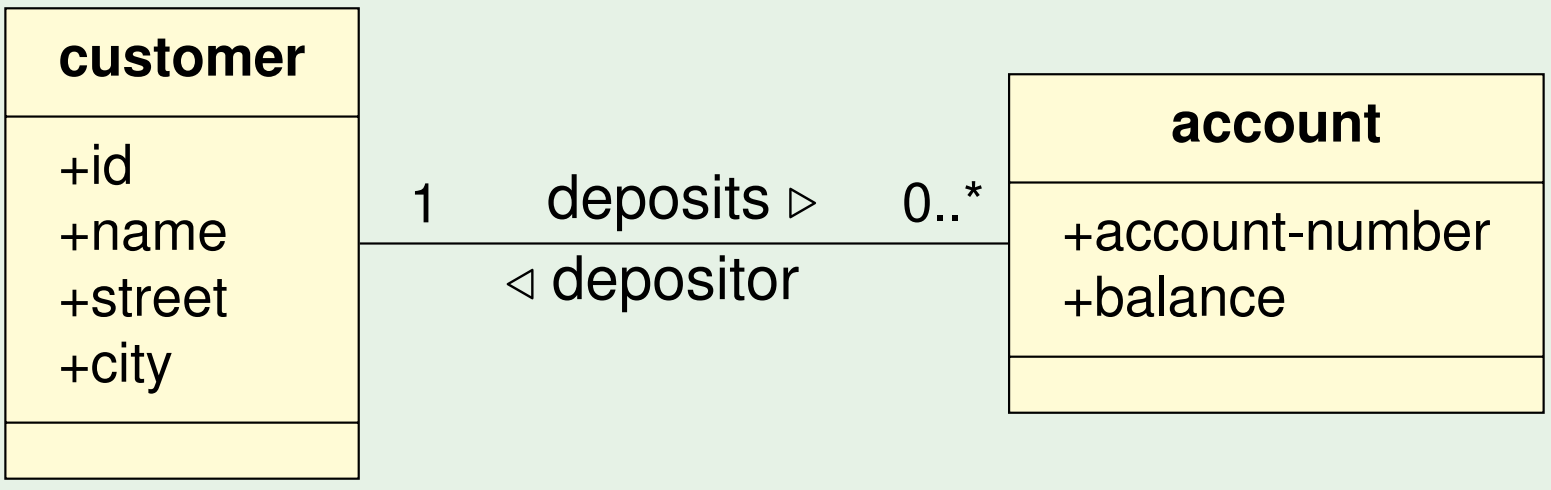


disjoint generalisation

Unified Modelling Language

UML = Unified Modeling Language

Example Schema as UML Class diagram

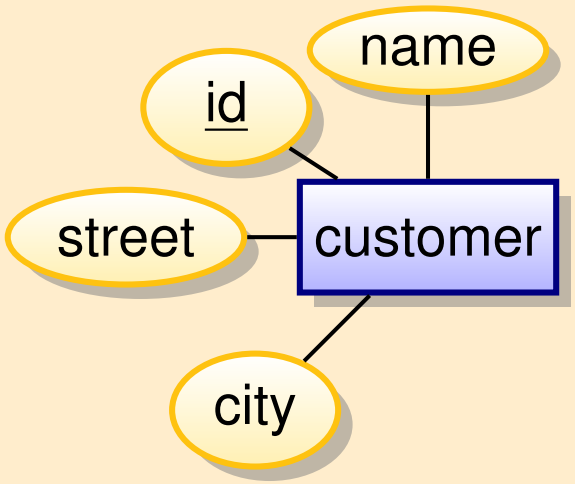


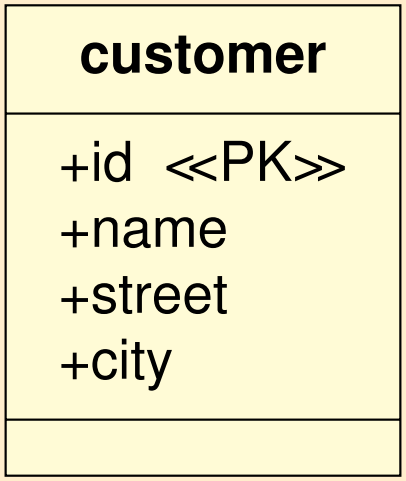
- UML diagrams are similar to E/R diagrams

  However, there are important differences!

E/R Models vs. UML Class Diagrams

Entity Sets and Attributes

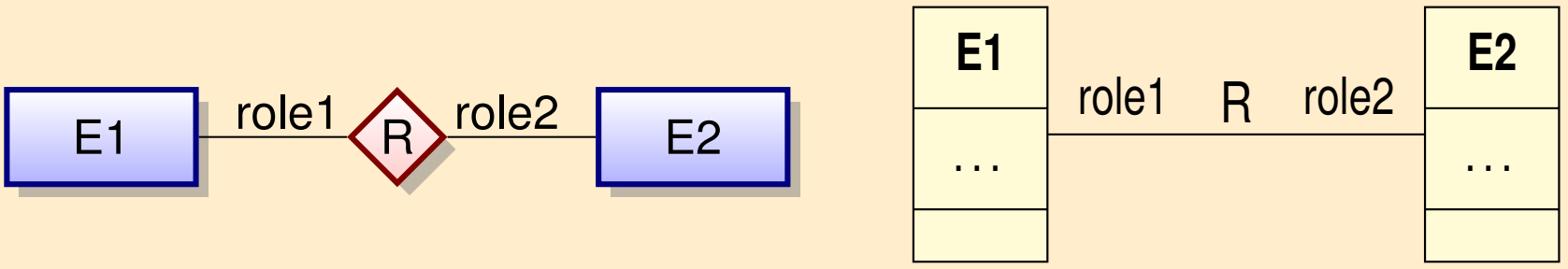




- In UML attributes are shown within the box of the entity set rather than as separate ellipses in E/R models.

E/R Models vs. UML Class Diagrams

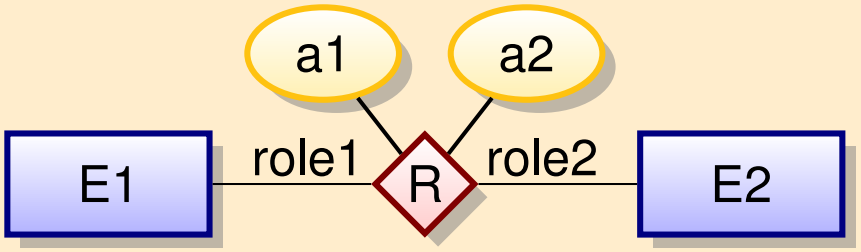
Binary Relationships

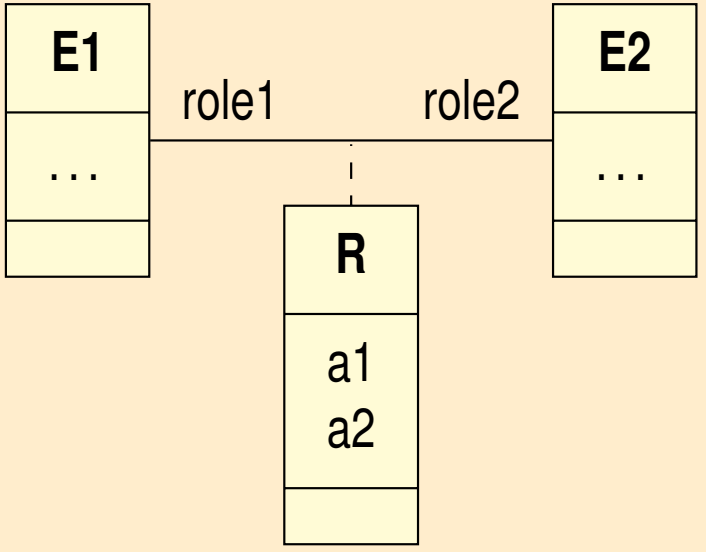


- In UML binary relationship sets are represented by a line connecting the entity sets. The name of the relationship set is written adjacent to the line.

E/R Models vs. UML Class Diagrams

Binary Relationships with Attributes

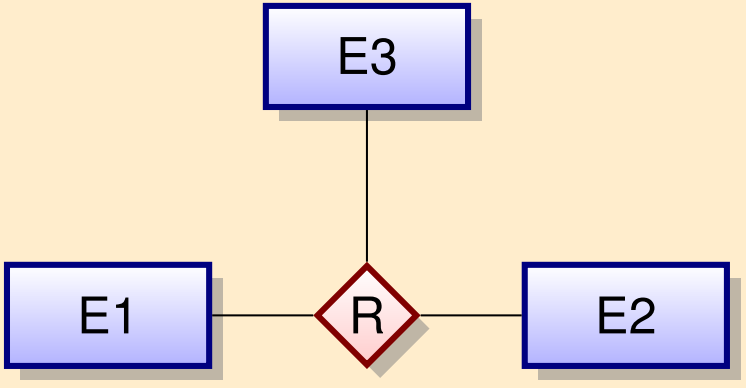


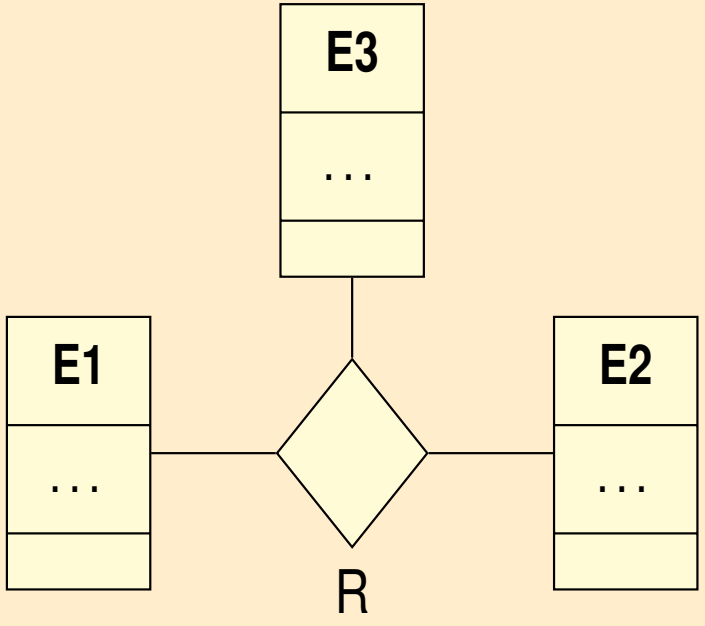


- If the relationship set has attributes, then the name of the relationship set is written in a box together with the attributes of the relation.
- The box is then connected using a dashed line to the line corresponding to the relationship set.

E/R Models vs. UML Class Diagrams

Non-Binary Relationships

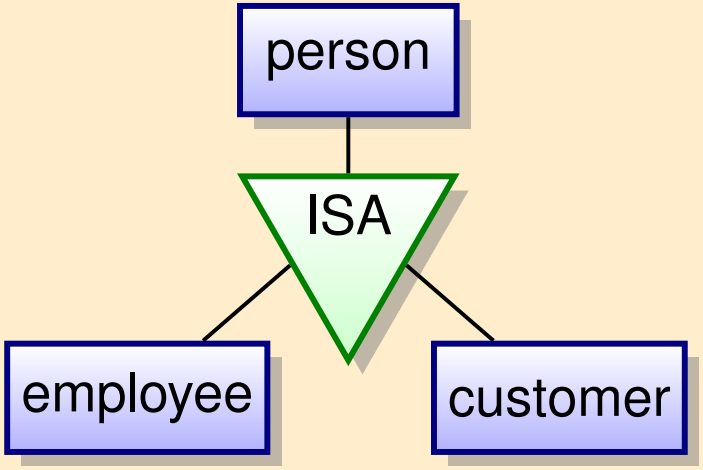


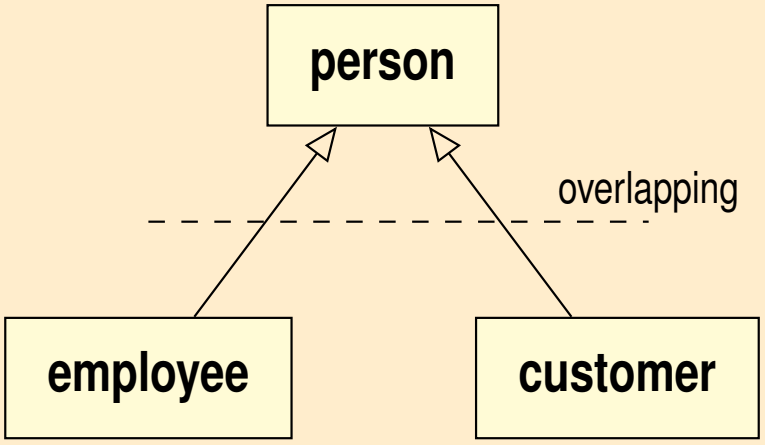


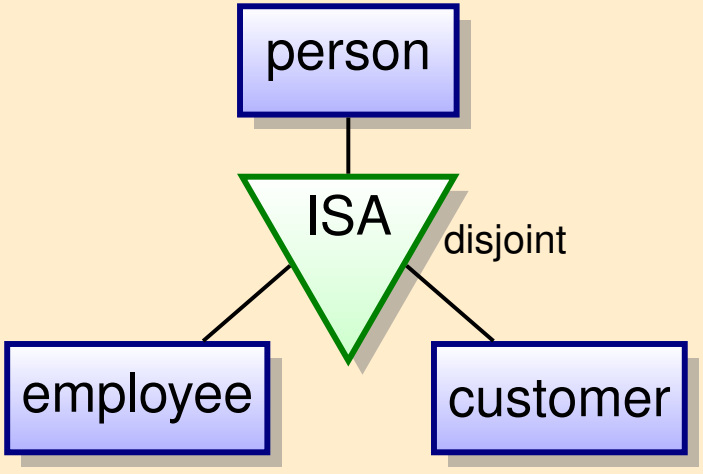
- Non-binary relationship sets are drawn using a diamond.

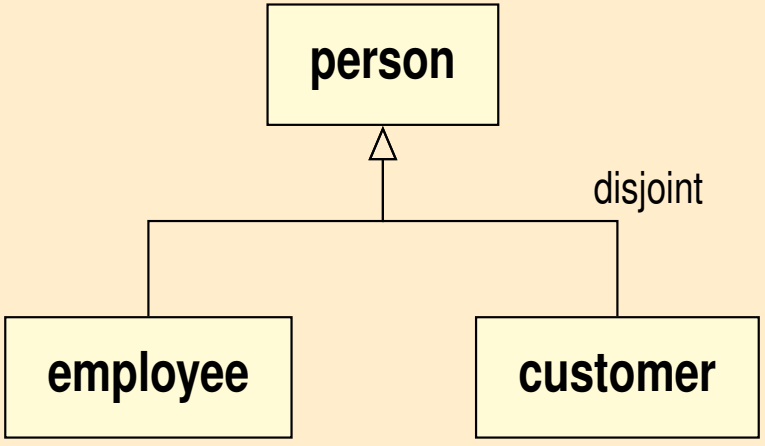
E/R Models vs. UML Class Diagrams

Generalisation and Specialisation



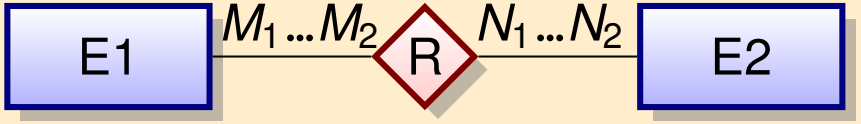


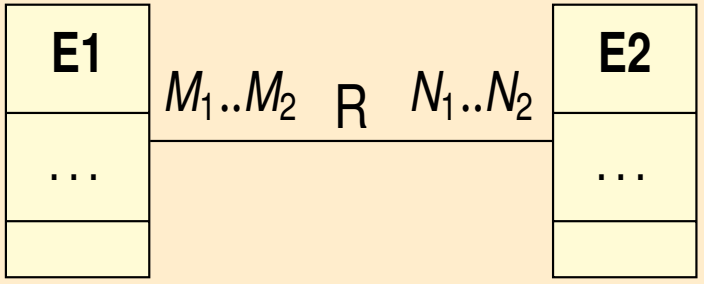




E/R Models vs. UML Class Diagrams

Cardinality Limits





The cardinalities indicate that:

- each E2 entity is related to $\geq M_1$ and $\leq M_2$ entities in E_1
- each E1 entity is related to $\geq N_1$ and $\leq N_2$ entities in E_2

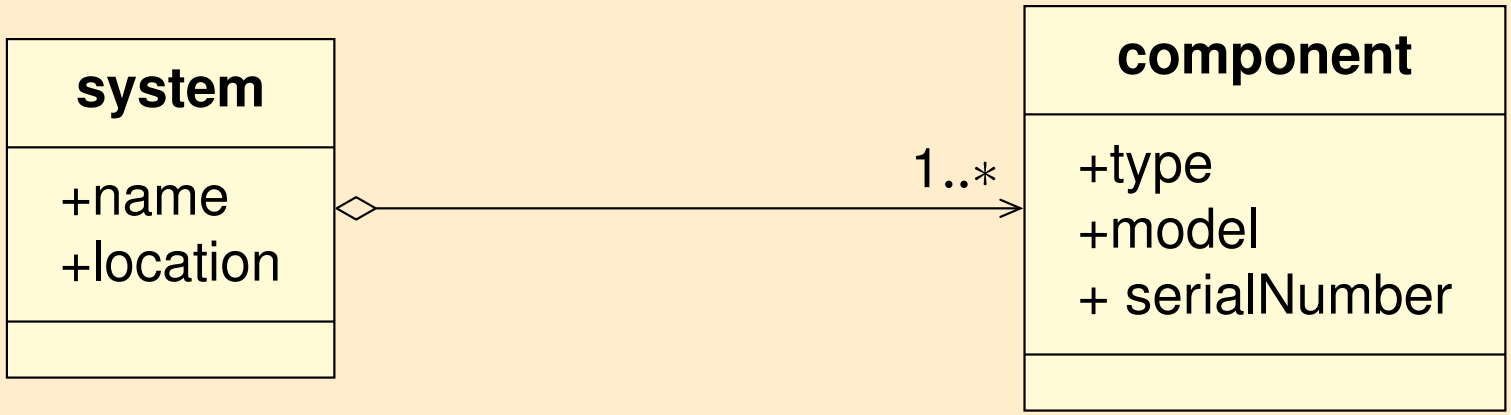
In UML we have the following abbreviations:

- 1 stands for 1..1
- $*$ stands for 0..$*$

Often better to write fully 1..1 and 0..$*$.

UML: Aggregation and Composition

Aggregation in UML



- **Aggregation:** system is a collection of components
- **Composition:** if the diamond would be filled black, it would mean that every component belongs to one system (1..1)

It is important to note the difference with E/R models:

- In E/R aggregation allows to treat relations as entities.
- Composition in UML is similar to weak entities in E/R.

  However, composition in UML says nothing about keys.

Differences: E/R Models vs. UML Class Diagrams

- visual differences — no big deal
- **keys**:
  - E/R supports keys (underlining)
  - UML has no standard for indicating keys

    Some people underline, others write PK after the attribute.
- **aggregation**: means something very different
  - in E/R: treating a relationship set as an entity
  - in UML: a part-whole relation
    (non-exclusive form of composition)
- **weak entities**:
  - in E/R: weak entities are entities without own key
  - in UML: composition is similar, but says nothing about keys

Data Modelling: Objectives

After completing this chapter, you should understand:

- **Three phases of database design**
  - Conceptual, Logical, Physical, and what these are useful for
- **Basic E/R concepts**
  - entities, attributes, relationships, ‘is a’, weak entity sets, aggregation
  - cardinality/participation constraints
- **How UML corresponds with and differs from E/R**
  - differences: basic syntax, aggregation, key specifications
- **How to make a conceptual model given a scenario**
  - in both UML and E/R