Database1

Kandahar University
Computer Science Faculty

Sayed Ahmad Sahim

DBMS 4_ER_Model

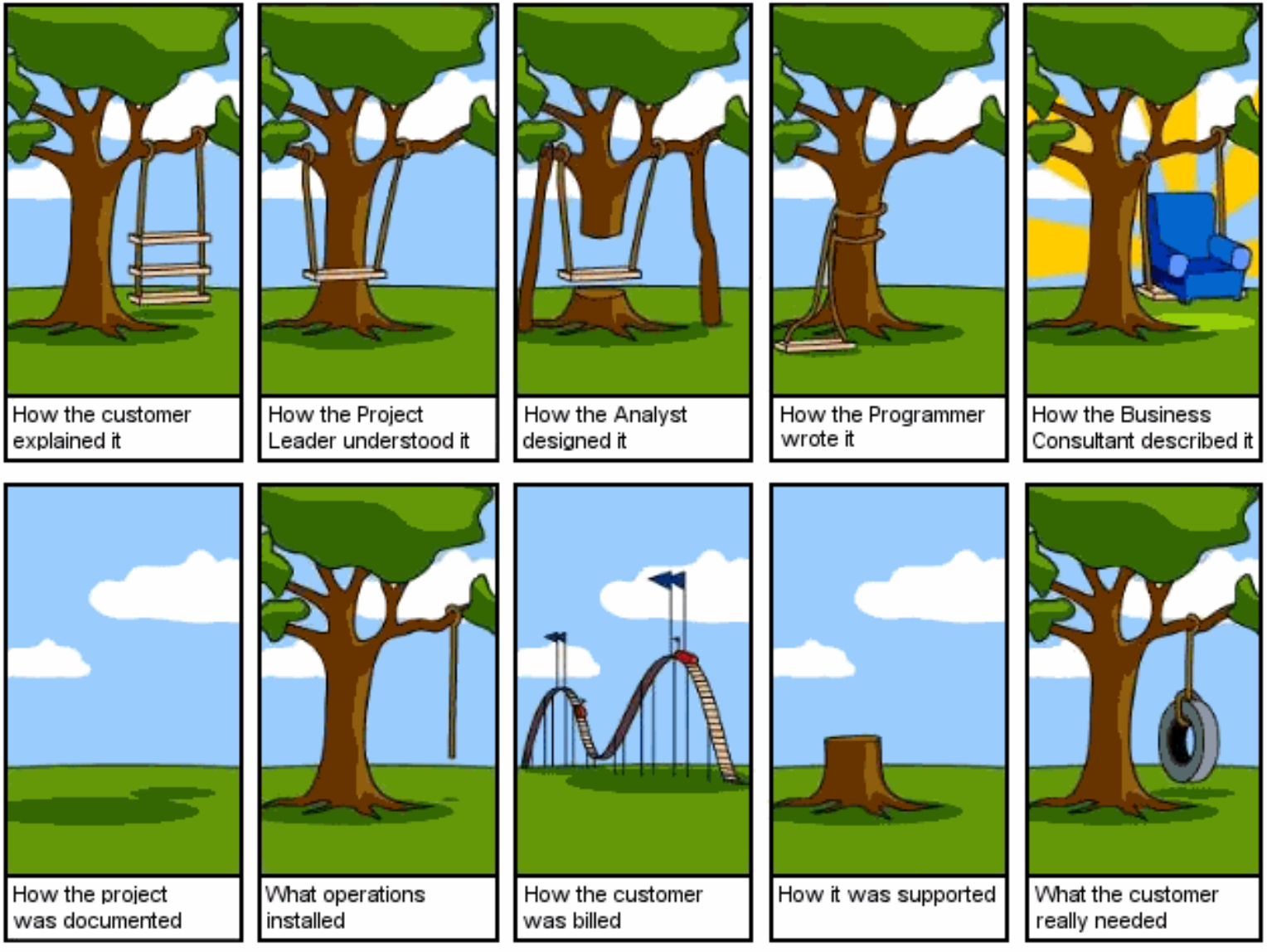
The Lecture

1. High-level motivation for the E/R model
2. Entities
3. Relations

Database Design

Database design? Why do we need it?

Database Design



Database Design

Database design: Why do we need it?

- Agree on the **domain**, the **topic**, the **focus**
- Agree on **structure of the database**
- **before** starting implementation!

Database Design

Consider issues such as:

- What **entities** to model
- which not
- How entities are **related**
- What **constraints** exist in the domain
- How to achieve **good designs**

Several formalisms exist

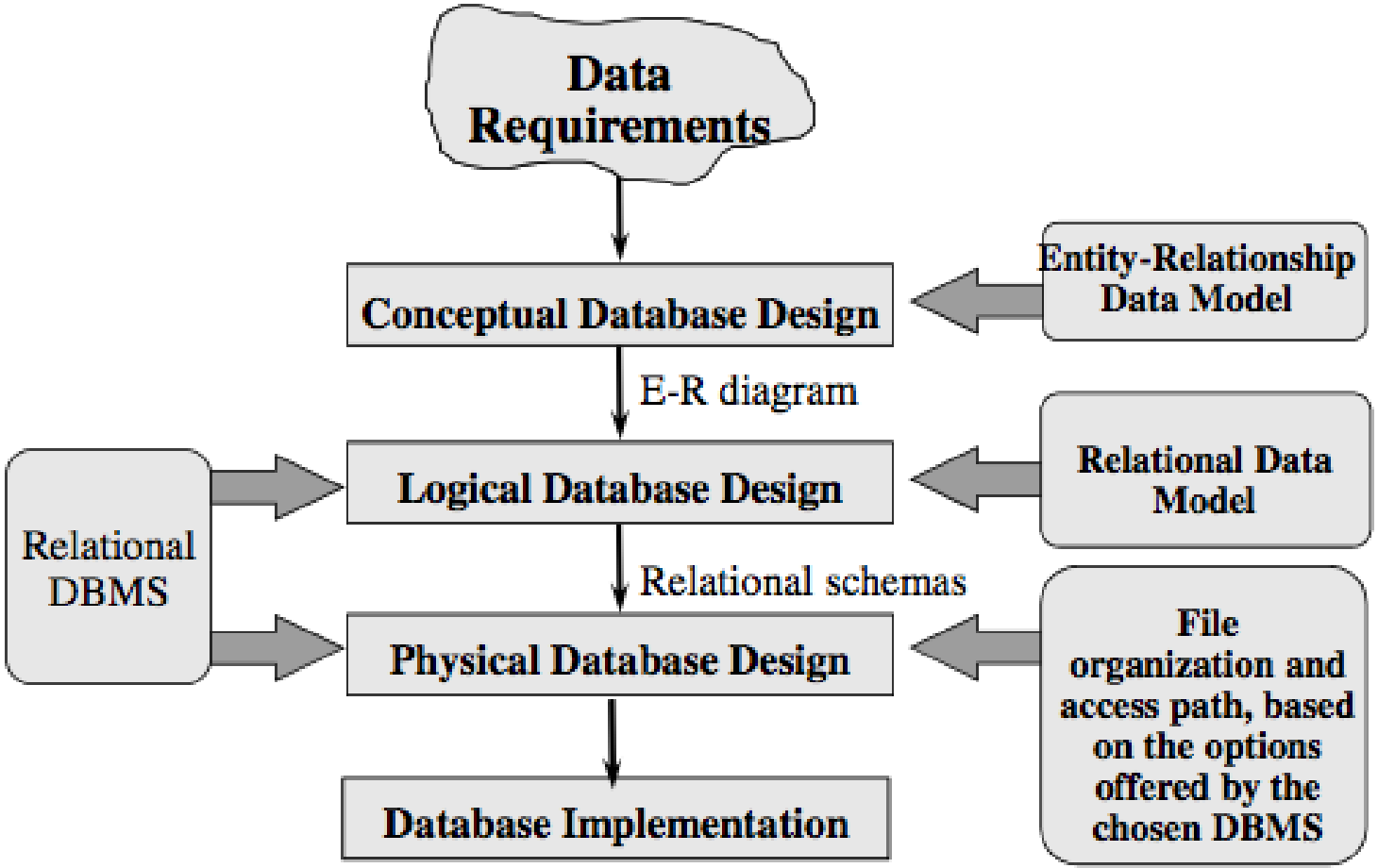
- We discuss one flavour of **Entity Relationship (E/R) Model**

Modeling languages

- **Entity Relationship (E/R) Model:** historically important and still used. An "OO-light" approach.
- **Unified Modeling Language (UML):**Different types of diagrams are useful for modeling databases $\Rightarrow$ SE lecture

Design Process Schema

1. Requirements analysis
2. Conceptual Design
3. Logical
4. Physical Design



1. Requirements analysis

To be considered:

- What is going to be stored?
- How is it going to be used?
- What are we going to do with the data?
- Who should access the data?

Technical and non-technical people are involved.

2. Conceptual Design

- A **high-level description** of the database
- **Sufficiently precise** that technical people can understand it
- **Sufficiently simple** that non-technical people can participate

3. Logical Design

More detail than the conceptual ER model, without regard to physical implementation.

- Include all **entities** and **relationships** between them.
- Specify **attributes** for each entity.
- Specify **primary key** for each entity.
- Specify **foreign keys** for relationships.
- remove **redundancy** by means of normalization.

4. Physical Design

Modelling for a specific DBMS implementation.

- Specify all **tables and columns**.
- Include **foreign keys** to identify relationships between tables.
- May include **denormalization**, depending on user requirements.
- **May be** significantly **different from the logical data model**.
- Will differ **depending on DBMS**.
- **Security** Design

Design Phases Overview

Feature	Conceptual	Logical	Physical
Entity names	X	X	
Entity relationships	X	X	
Attributes		X	
Primary keys		X	X
Foreign keys		X	X
Table names			X
Column names			X
Column data types			X

Requirement Analysis

- **discuss** the problem
- identify **core concepts**
- express the **meaning of terms and concepts** used by domain experts
- find the correct **relationships** between different concepts.

Tools

- Use case
- User Story
- Domain Dictionary

Domain model

Model

- a simplified and idealized understanding of a system
- abstraction of things in the real world
  - intended purpose

Topographic vs. topological

Domain model - Miniworld

- miniworld or universe of discourse
- represents **some aspect** of the real (or an imagined) world
- changes to the miniworld are reflected in the database

The world described by the 7th-century scholar Isidore of Seville.

Example miniworld

A **University** concerned with **students, courses, course sections, grades, and course prerequisites**

Entity-Relationship (E/R) Model

Entity-Relationship (E/R) Model

- **high level** description.
- easy to understand for **non-technical**.
- rigorous enough to be used for **system building**.

Concepts available in the model

- **entities** and **attributes** of entities.
- **relationships** between entities.
- diagrammatic notation.

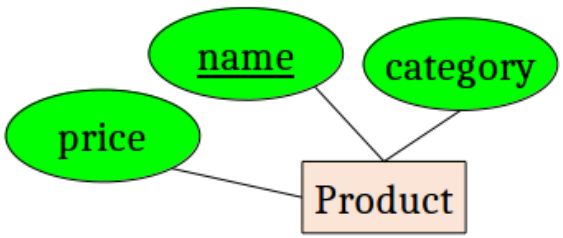
Impact of the ER Model

- The E/R model is one of the most cited articles in Computer Science
- "The Entity-Relationship model – toward a unified view of data" Peter Chen, 1976

Entities & Entity Sets

Entity set or **Entity type**

- **class** or **type** of objects
- collection of entities
  - common **attributes**
    - Ex: Persons, Products



Entity

- class instance
  - individual set of attributes
    - Ex: A specific person or product

Entity sets are shown in E/R diagrams - as **rectangles**.
They represent the sets of all possible entities.
Entities are the individual objects, which are members of **entity sets**.

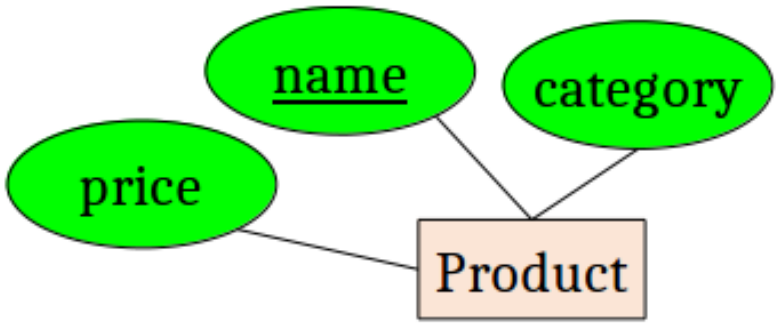
Entity Type

Entity type:

- represents some abstract "thing" of interest
- is described by its attributes

Attribute:

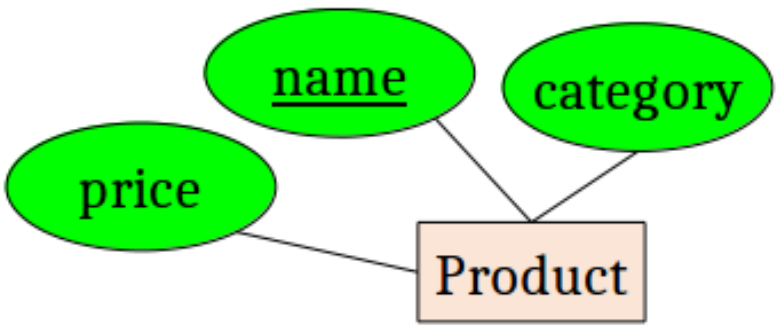
- properties characterize an entity
- values are drawn from some domain (set of meaningful values).
- represented by **ovals** attached to the entity type



Shapes are important. Colors are not.

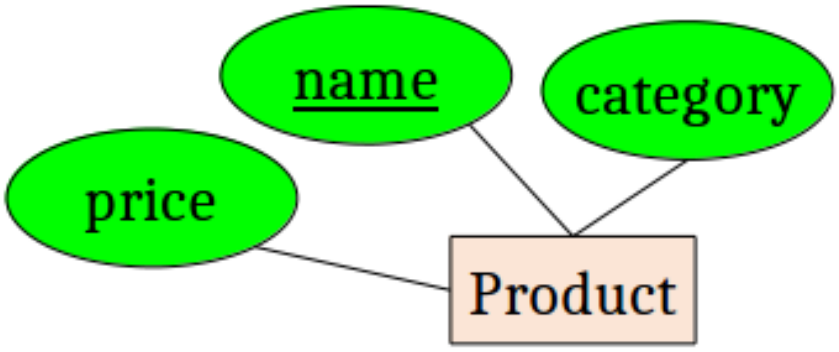
Entity Sets and Key Attributes

- **Key**
  - an attribute or a collection of attributes whose value(s)
  - **uniquely identify** an entity instance in the entity set.
  - **minimal**
- Keys are represented by **underlining the name**.
- Keys are determined by the designers
  - from the meaning of the attributes



The E/R model forces us to designate a **single primary key**, though there may be multiple candidate keys.

Key Example



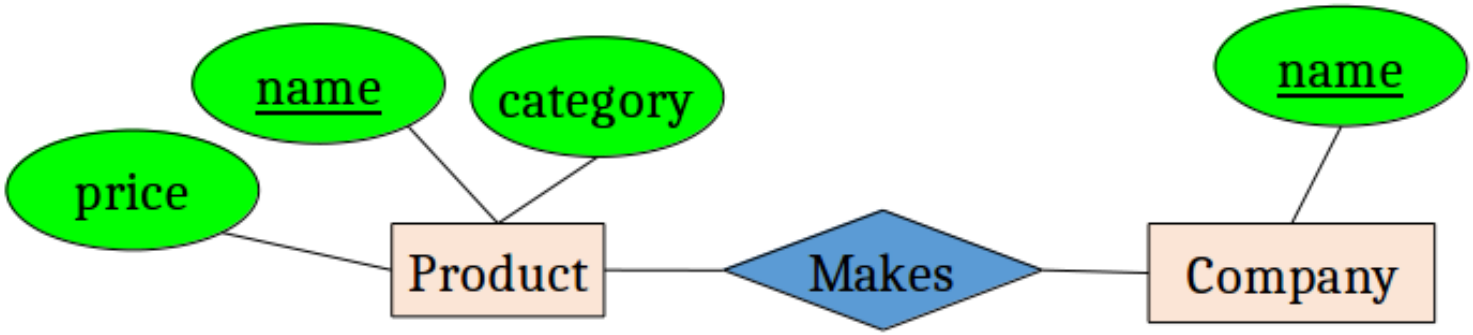
Here, name, category is not a key (not minimal).

If it were, what would it mean?

The R in E/R: Relationships

What is a relationship?

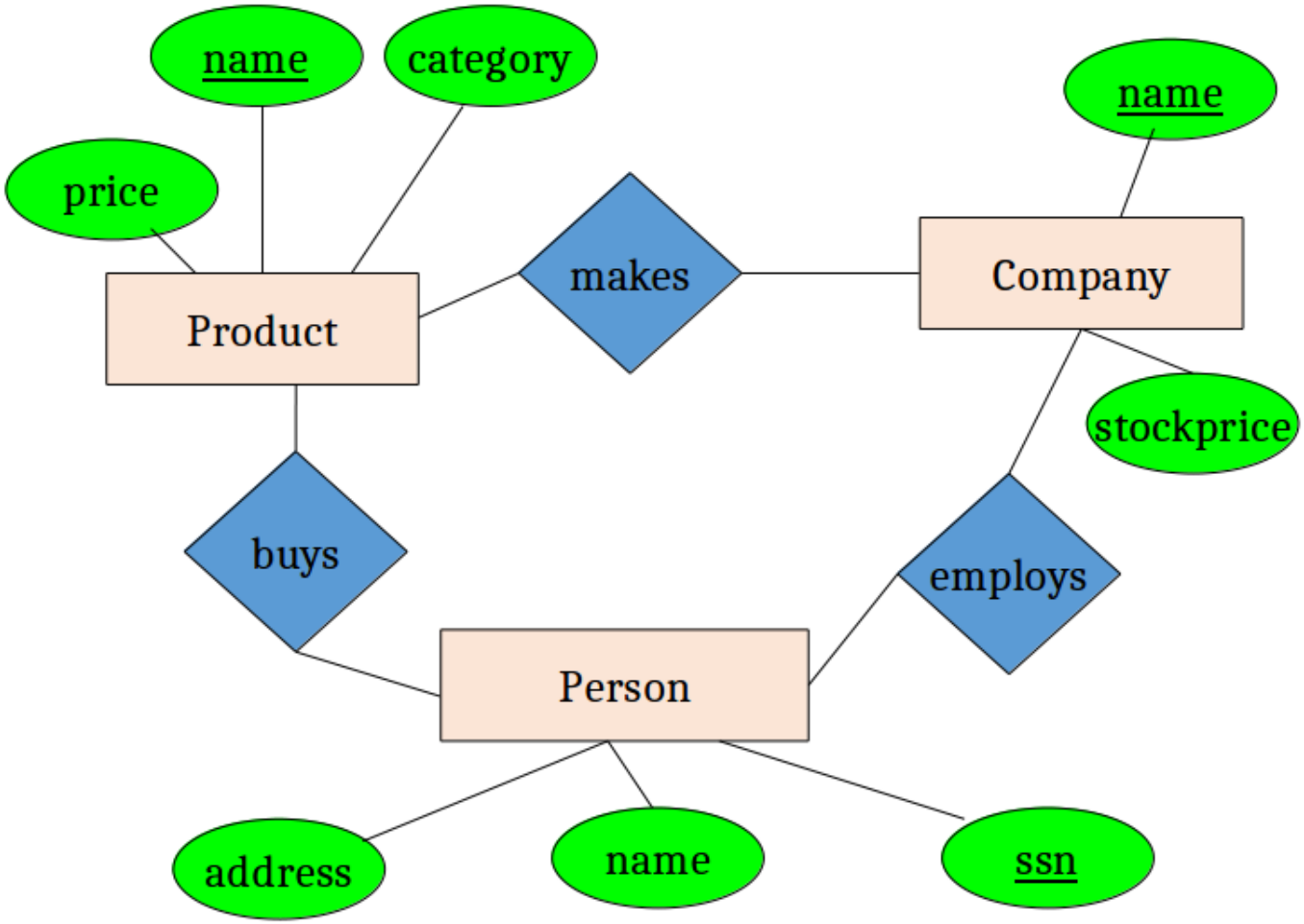
- A **relationship** is an association between entities



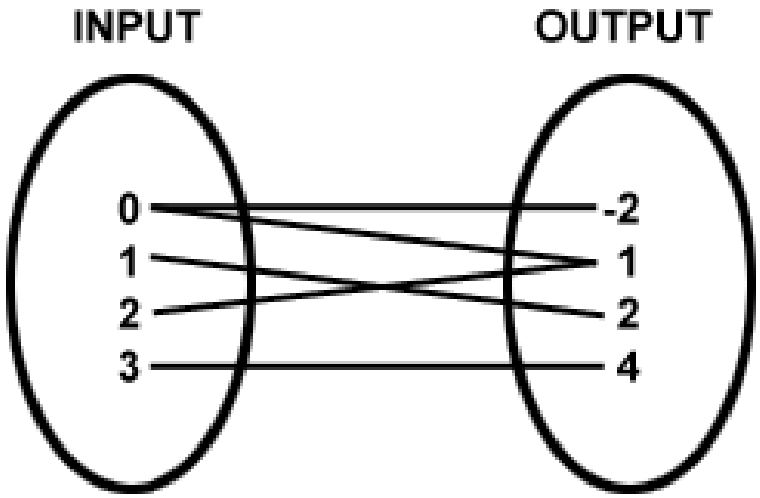
- A **relationship set** is a set of relationships of the same type

Relationship types are shown in E/R diagrams - as **diamonds**.
They represent the sets of all possible relationships.

The R in E/R: Relationships



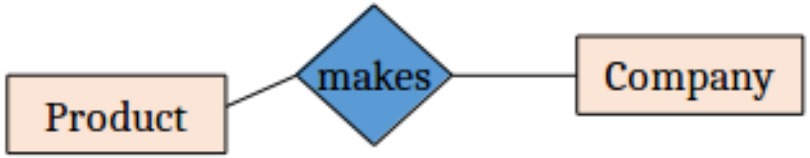
Relationships and Sets



- Let I(nput) and O(utput) be Sets
  - $\boldsymbol{I = \{0, 1, 2, 3\}, O = \{-2, 1, 2, 4\}}$
- $\boldsymbol{I \times O}$ is the set of all possible pairs $\boldsymbol{(i, o)}$
- $\mathcal{R} = \{(0, -2), (0, 1), (1, 2), (2, 1), (3, 4)\}$
- $\mathcal{R} \subseteq \ I \times O$

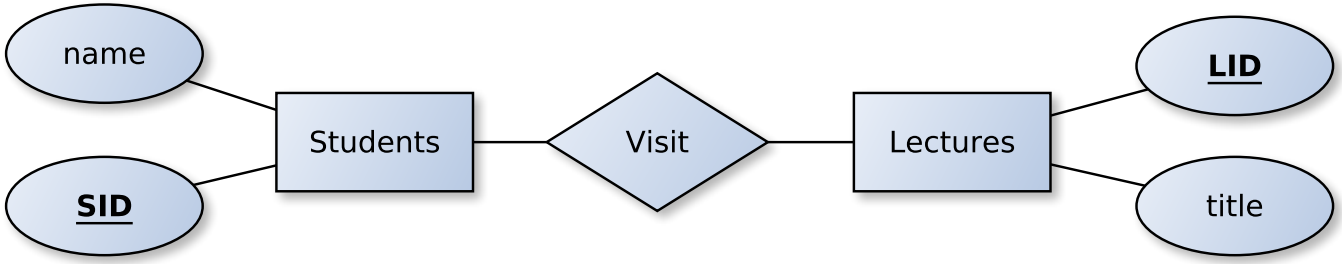
What is a Relationship

- Let *Product* and *Company* be sets
- ***Product*** $\times$ ***Company*** is the set of all possible pairs
- ***Makes*** $\subseteq$ ***Product*** $\times$ ***Company***
- (p,c) $\in$ ***Makes*** $\iff$ Product 'p' is made by Company 'c'



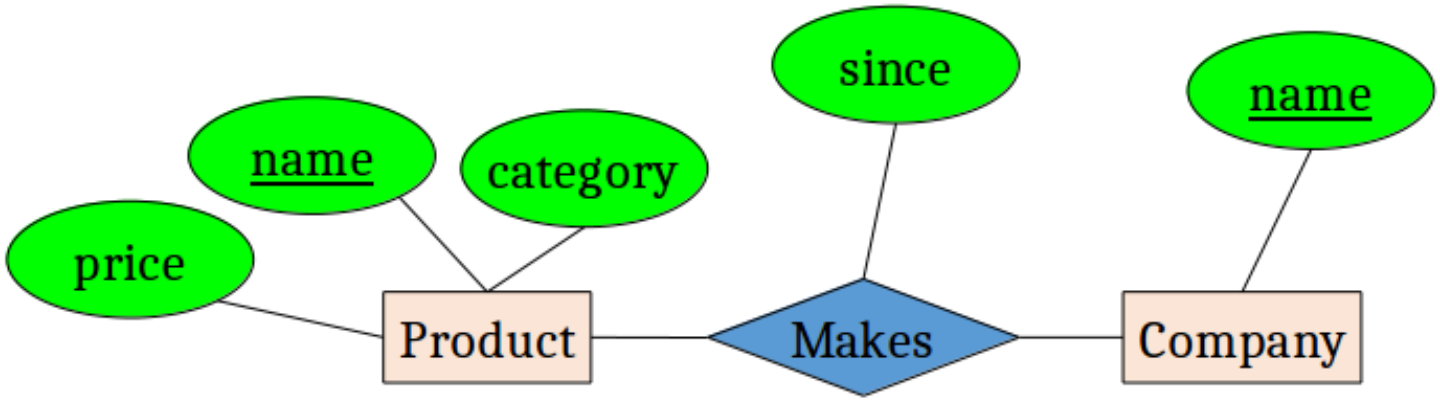
Concepts so far

- **entity sets**: Rectangles
- **relationship sets**: Diamonds
- **attributes**: Ovals
- **keys**: underlined



Relationships and Attributes

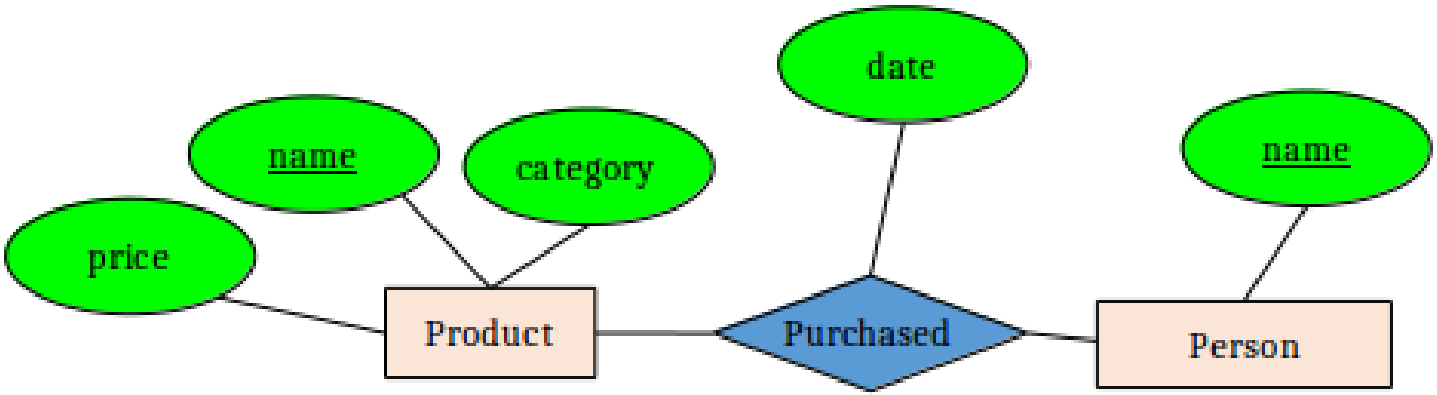
- Relationships may have attributes as well.



- e.g. "since" records when company started making a product
  - "since" is implicitly unique per pair here! Why?
  - Why not "how long"?

How to model a Relationship

Q: What does this say?

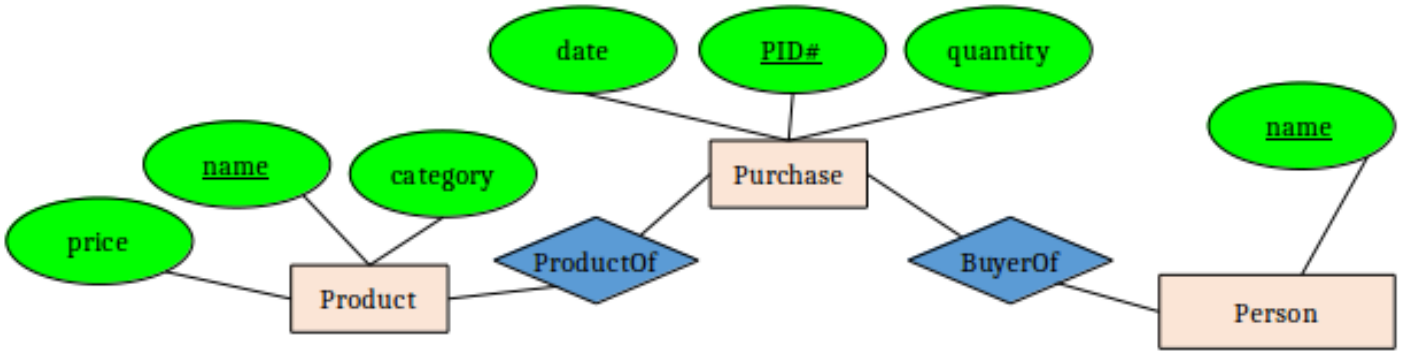


A: A person can only buy a specific product once (on one date)

Design decision

Modeling something as a **relationship makes it unique**
What if not appropriate?

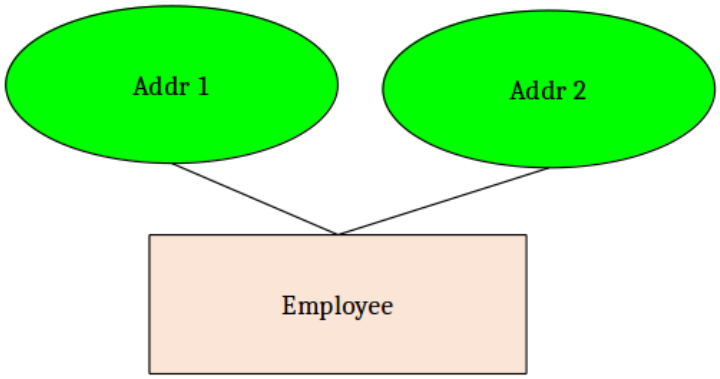
How to model a Relationship

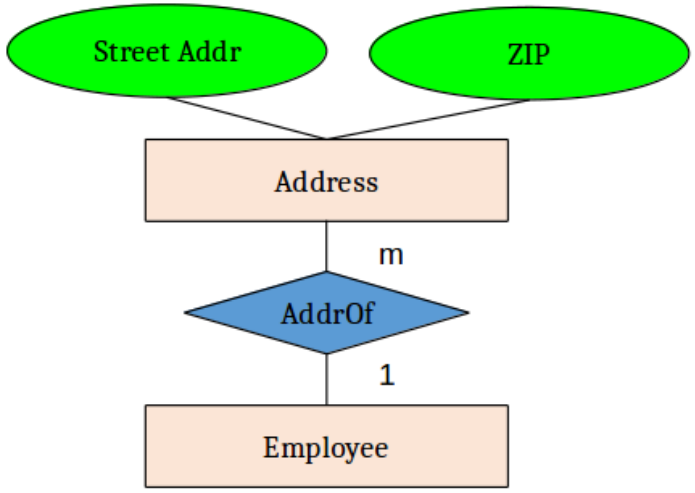


Design decision

We can **model a relationship as an entity**.
For example, to permit multiple instances of each entity combination

Entity vs. Attribute





In general, when we want to record several values, we choose new entity.

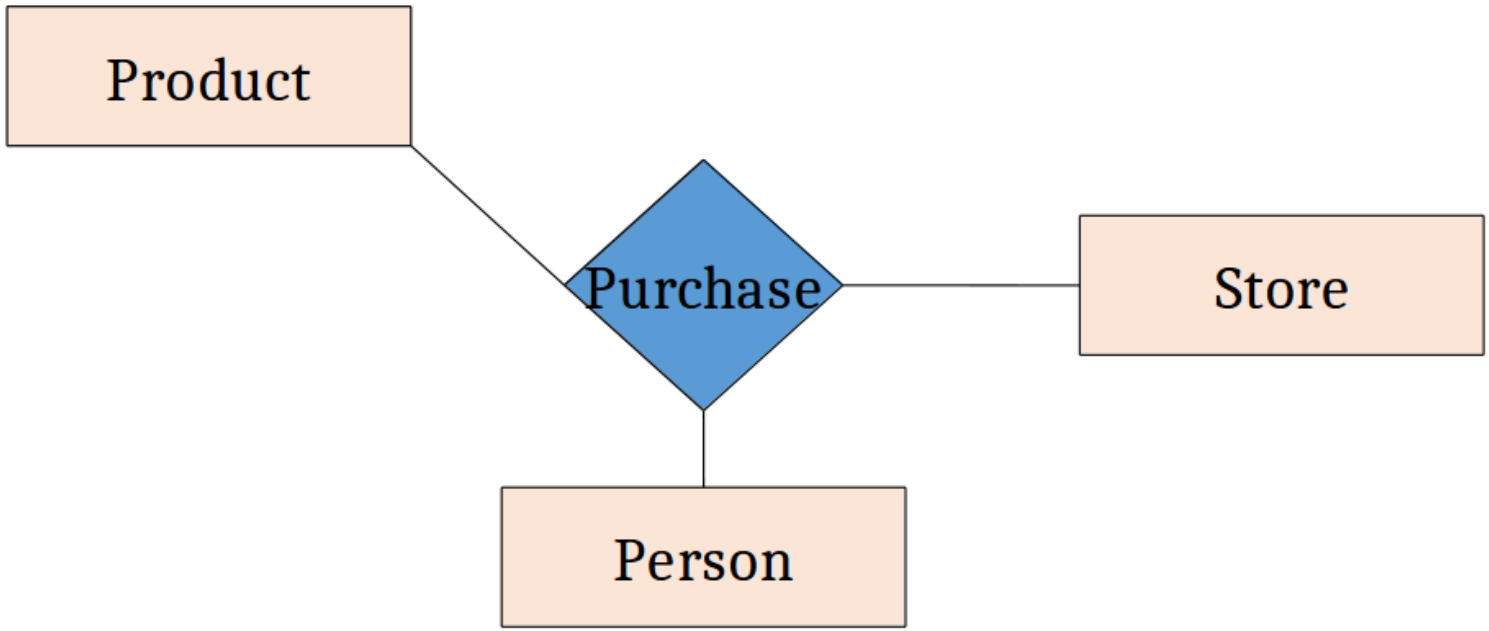
N-ary relationship - Degree

- Degree : the number of participating entity types.
  - Degree 2: binary
  - Degree 3: ternary
  - Degree n: n-ary

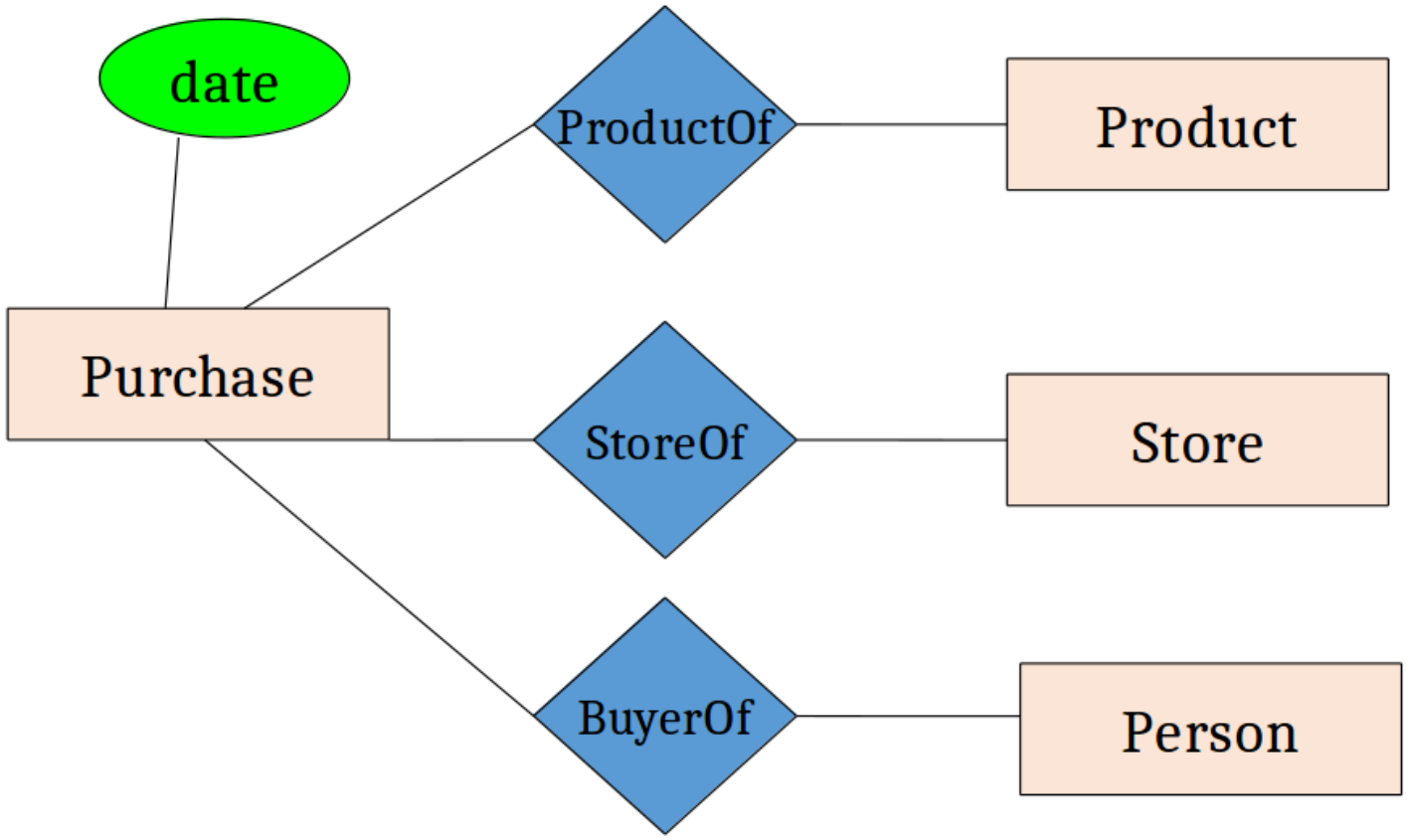
Binary relationships are very common and widely used.

Ternary Relationships

How could we model a purchase relationship between **buyers**,**products** and **stores**?

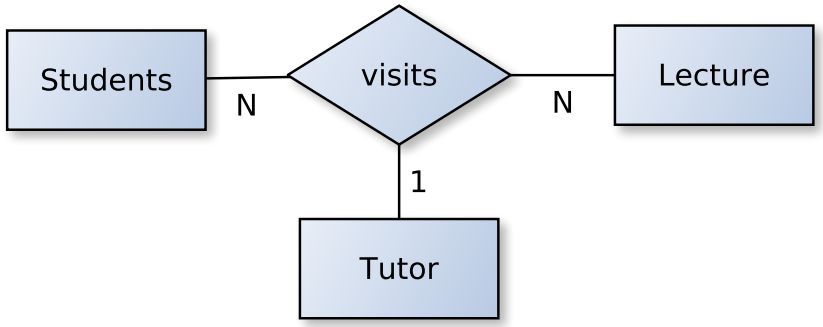


Converting a Ternary Relationship to Binary



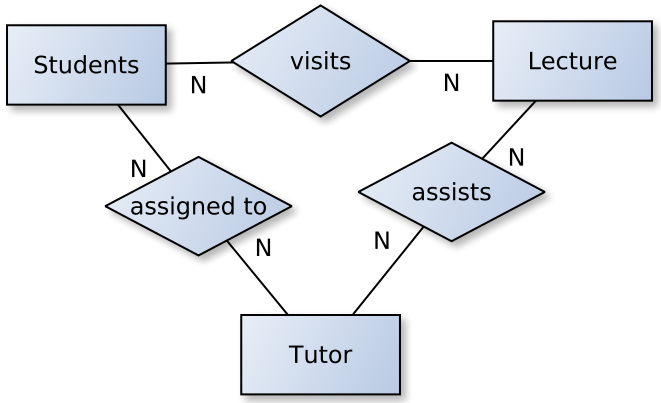
N-ary Relationship to Binary Example

- Each student has a tutor assigned for a certain lecture:



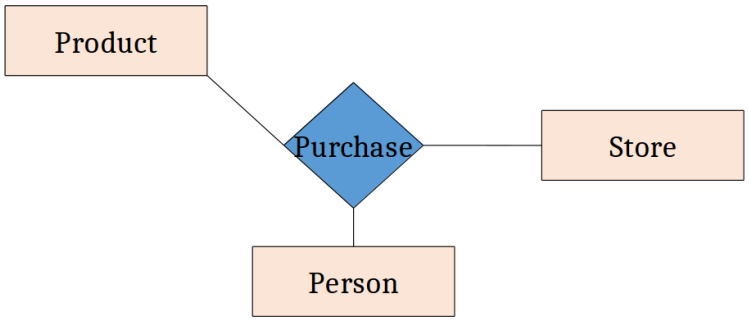
- This relationship is ternary , can we replace it by binary relationships?

N-ary Relationship to Binary Example

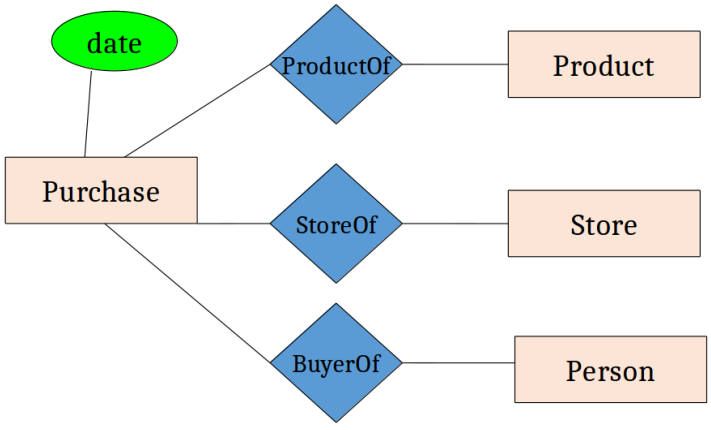


- does this model replace our ternary relationship?
- No, examine this example:
  - Karl visits the Math and DB lectures
  - Stefanie tutors the Math and DB lectures
  - Karl is assigned to Stefanie in Math, but not in DB

Decision N-ary or Entity?



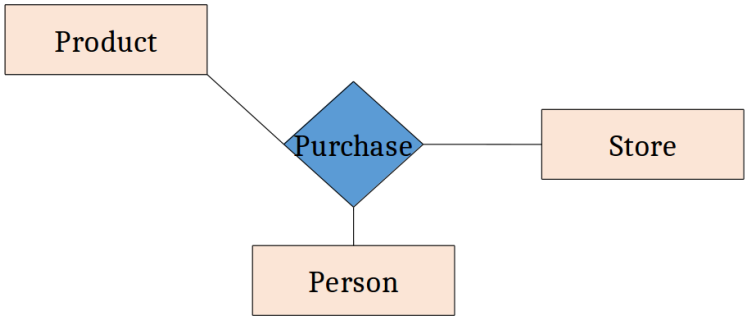
A) ternary Relationship



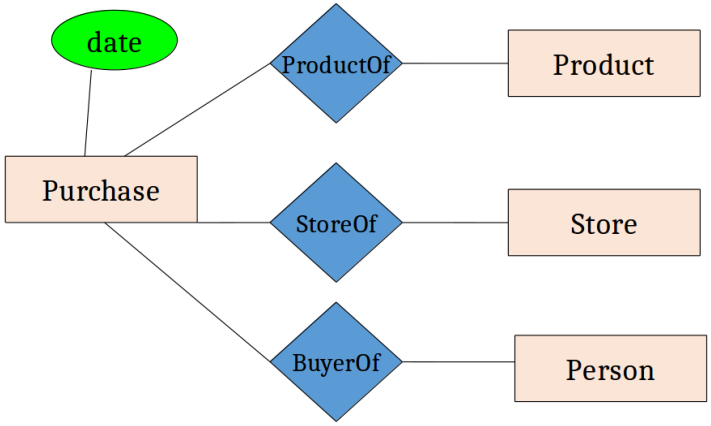
B) Entity + Binary

Should we use a **single n-ary** relationship or a **new entity** with binary relations?

Decision N-ary or Entity?



A) ternary Relationship



B) Entity + Binary

- (A) is useful
  - when a relationship really is between multiple entities
    - Ex: A three-party legal contract
- (B) is useful
  - if we want to have multiple relationship instances per entity combination
  - if we want to add details (constraints or attributes) to the binary relationships

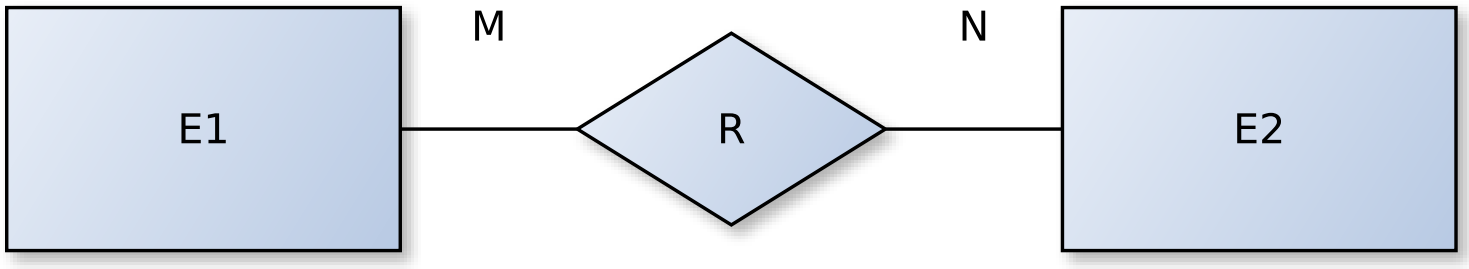
Constraints on Binary Relationship Types

E/R model and the relational model are logical representations of real world scenarios.

In order to make a **relationship type** an accurate model of our miniworld concept:

- we may impose certain **constraints**
- that **limit** the possible corresponding relationship sets.

Cardinality Ratio



Cardinality Ratio

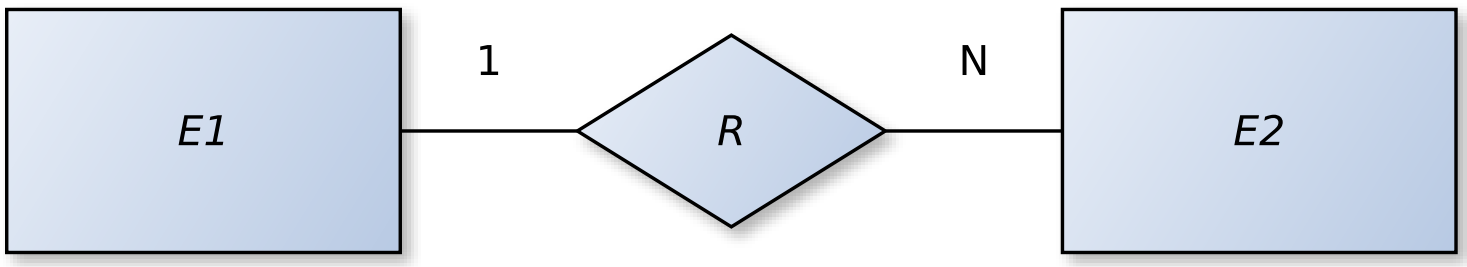
Specifies **maximum number** of relationship instances that an entity can participate in.

Four types:

- one-to-one (1:1)
- one-to-many (1:N)
- many-to-one (N:1)
- many-to-many (M:N)

Cardinality Diagram Notation

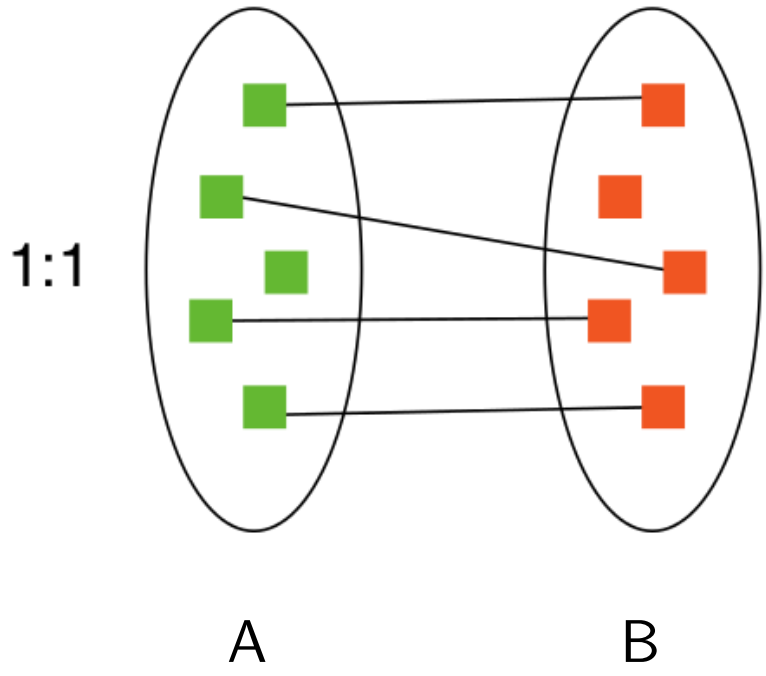
- We indicate cardinality by writing the
  - **max. number** of associated entities
    - next to the associated entity
    - on the line.



- alternative notations use special types of arrows
- or indicate a range like (0,N), (0,1) or (1,N)

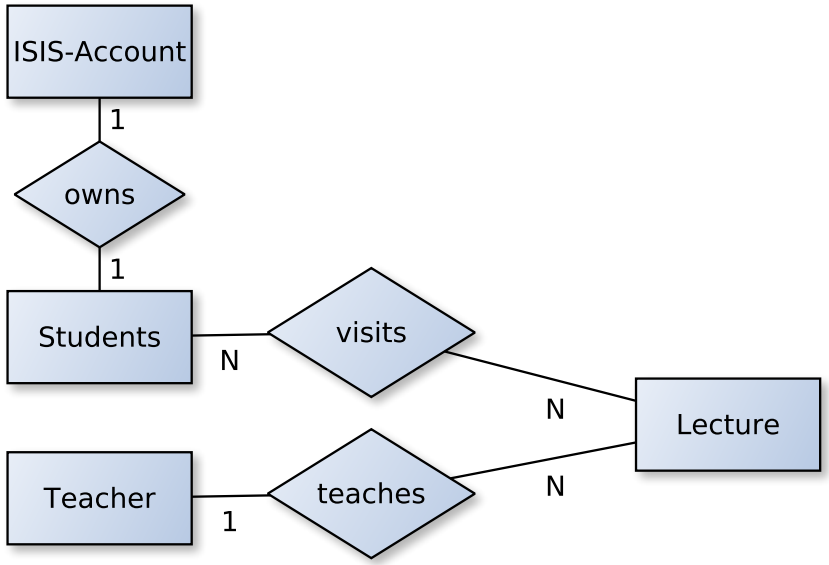
One to one (1:1)

- given relationship R
- each entity in A is related to 0 or 1 entities in B and vice versa:



Constraint: No instance of A may particpate in more than one instance of R; similarly for instances of B.

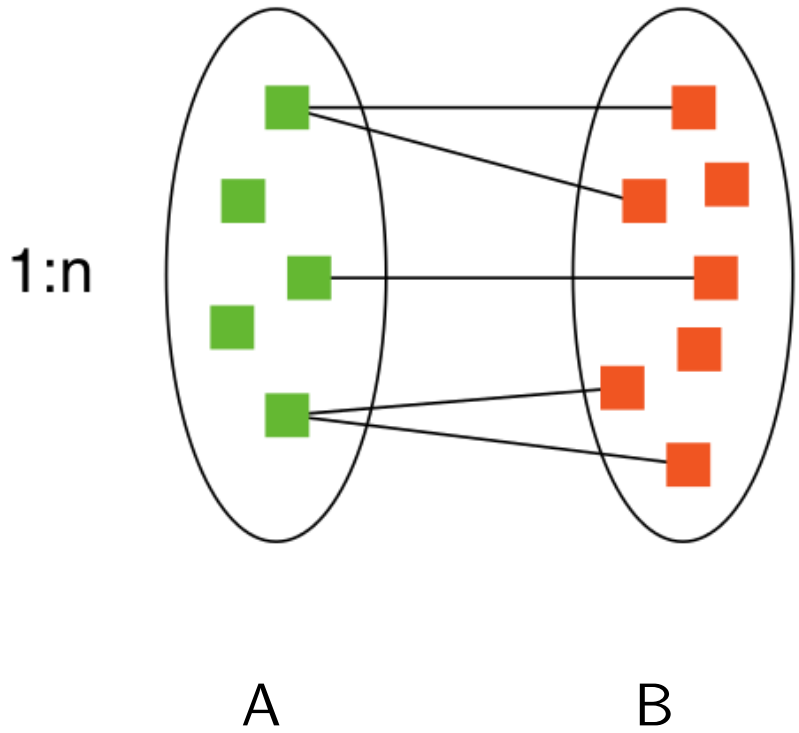
Example - One to one



- *students owning ISIS accounts*

One to many (1:N)

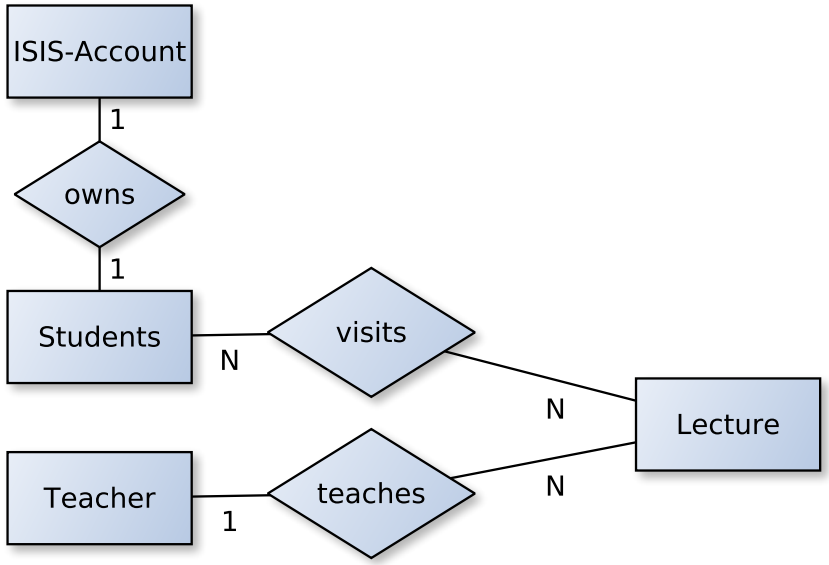
- given relationship R
- each entity in A is related to 0 or more entities in B
- each entity in B is related to 0 or 1 entities in A



Constraint: No instance of B may particpate in more than one instance of R; instances of A are under no such restriction.

Example - One to many

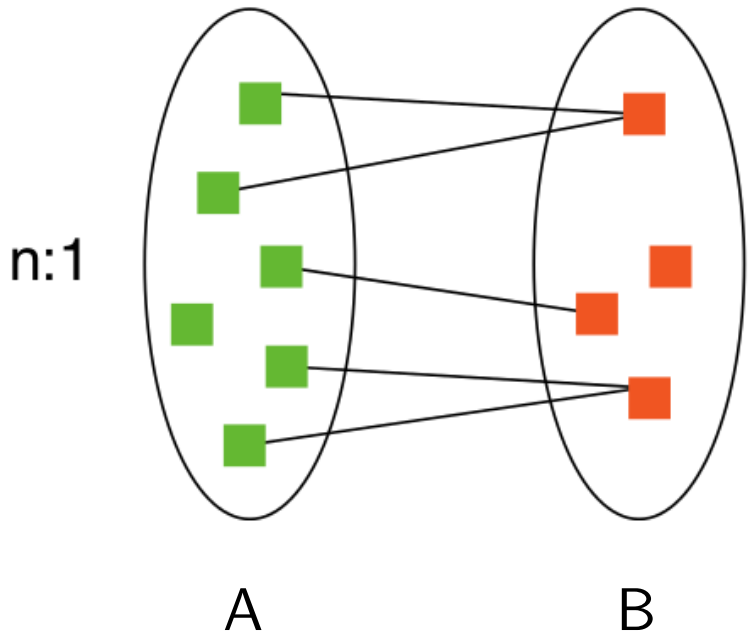
Constraint



- *teachers teaching lectures*

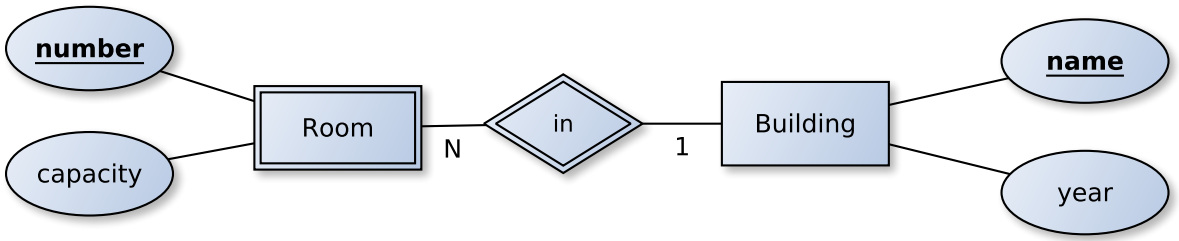
Many to one

- given relationship R
- each e. in A is related to 0 or 1 in B
- each e. in B is related to 0 or more e. in A



Constraint: No instance of A may particpate in more than one instance of R; instances of B are under no such restriction.

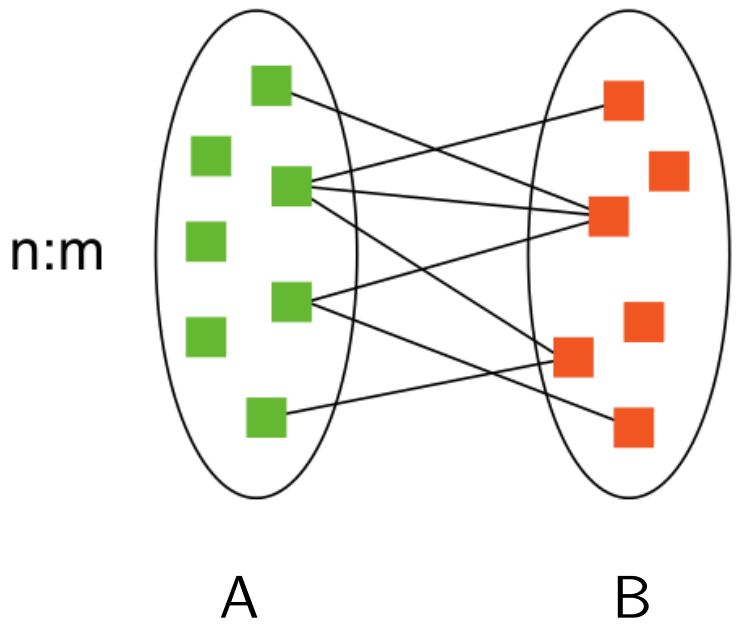
Example - Many to one



- *rooms in building*

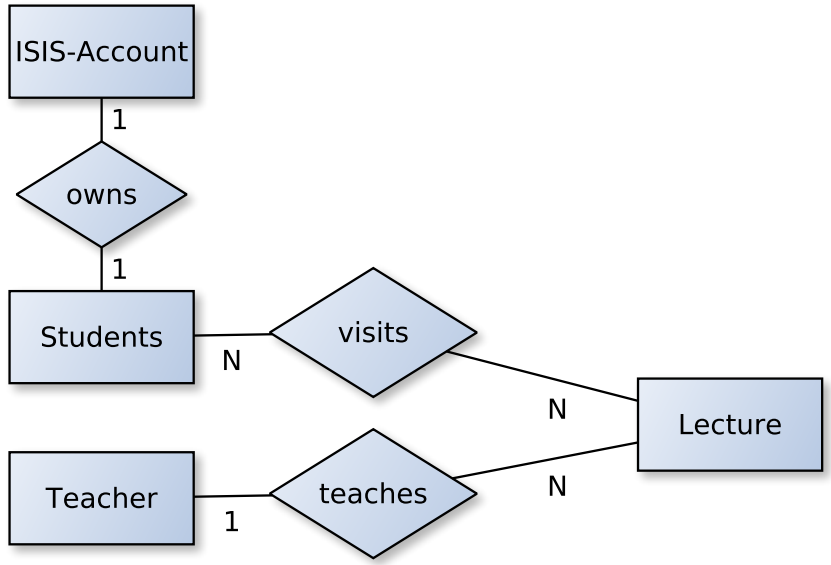
Many to many (N:M)

- given relationship R
- each entity in A is related to 0 or more entitities in B
- and vice versa



Constraint: Absence of Constraint. No restriction!

Example - Many to many



- *students visiting lectures*

Determine cardinality ratio design

Given a binary relationship R, that relates entity types A and B, respectively, the questions you should ask:

- May a given entity of type B be related
  - to multiple entities of type A?
- May a given entity of type A be related
  - to multiple entities of type B?

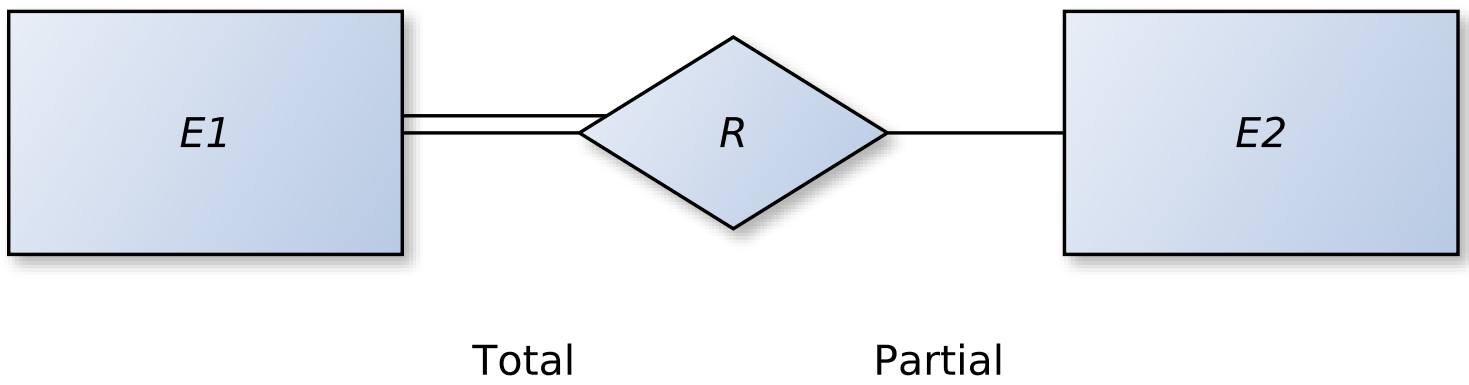
The pair of answers you get maps into the four possible cardinality ratios as follows:

- (yes, yes) $\Rightarrow$ M:N
- (yes, no) $\Rightarrow$ N:1
- (no, yes) $\Rightarrow$ 1:N
- (no, no) $\Rightarrow$ 1:1

Participation Constraint

An entity type may **participate** in a relation either **totally** or **partially**.

- **totally**
  - Every entity instance has to participate in the relationship.
- **partially**
  - Not all entities have to paticipate.

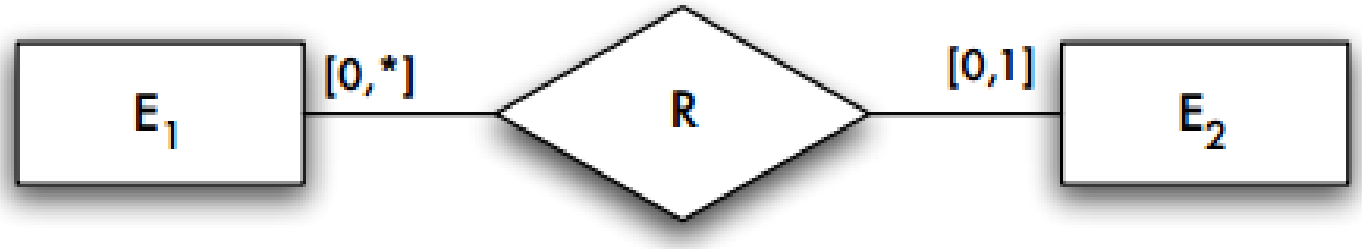


Structural Constraints

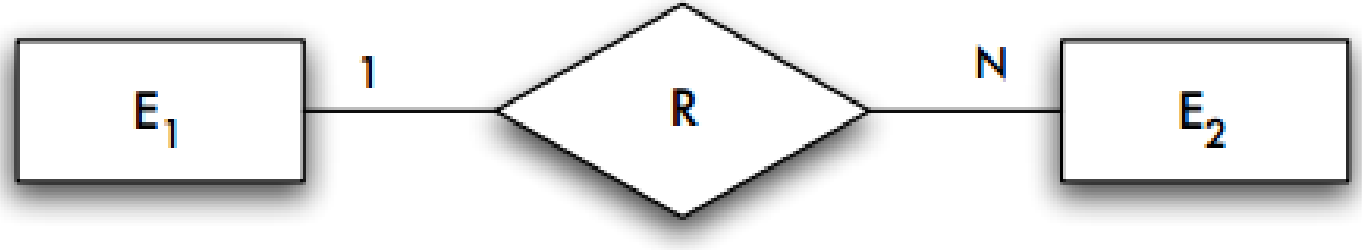
- **Cardinality Ratio** and **Participation Constraints** are together called **Structural Constraints**.

- They are called **constraints** as the **data must satisfy them** to be consistent with the requirements.

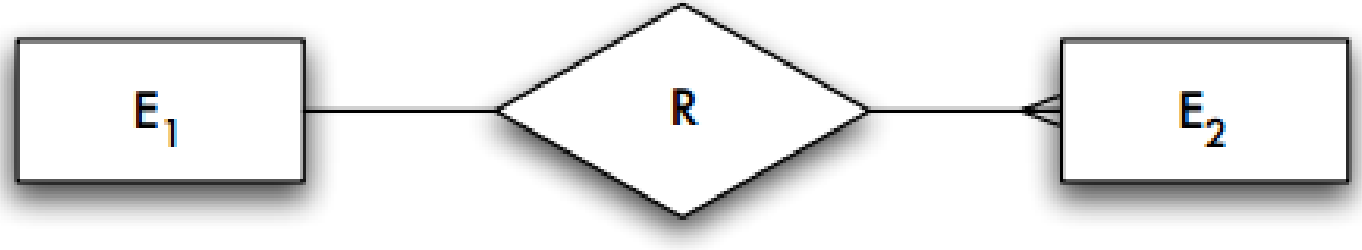
Alternative Notations for Cardinality



(a) [min,max]-Notation



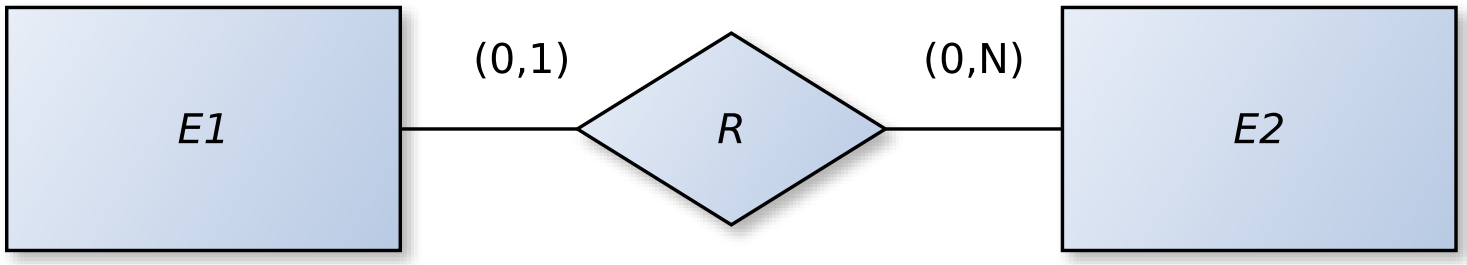
(b) Chen-Notation



(c) Krähenfußnotation

(Min,Max) Notation

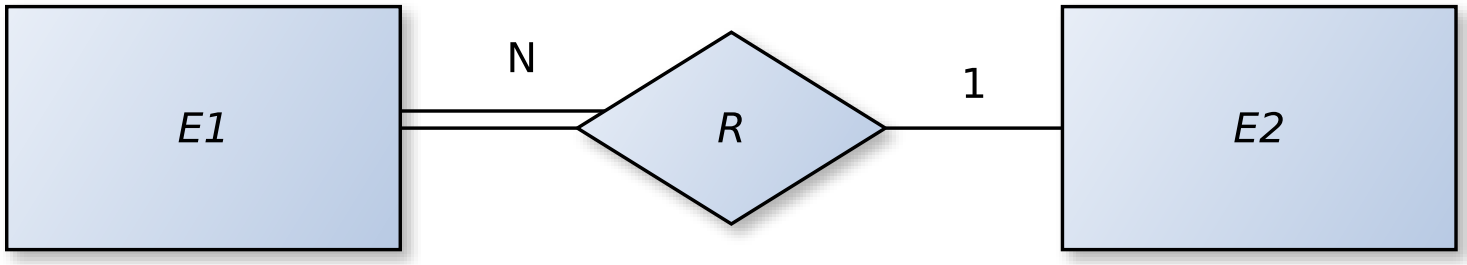
Min-Max notation: pair of numbers (m,n) placed on the line connecting an entity to the relationship.



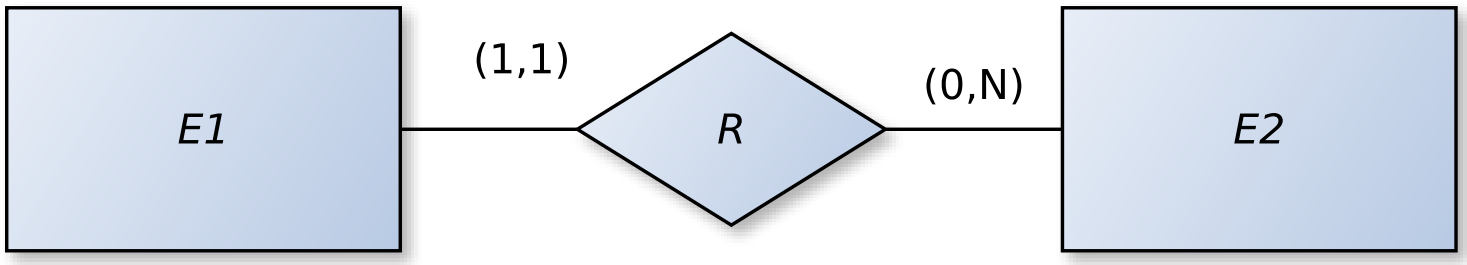
Min-Max notation: encapsulates **cardinality ratio and participation constraints**.

- m: the minimum number of times a particular entity must appear in the relationship set
  - 0 implies partial participation
  - 1 implies total participation
- n: similarly, the maximum number of times a particular entity can appear in the relationship set

Comparing the Notations

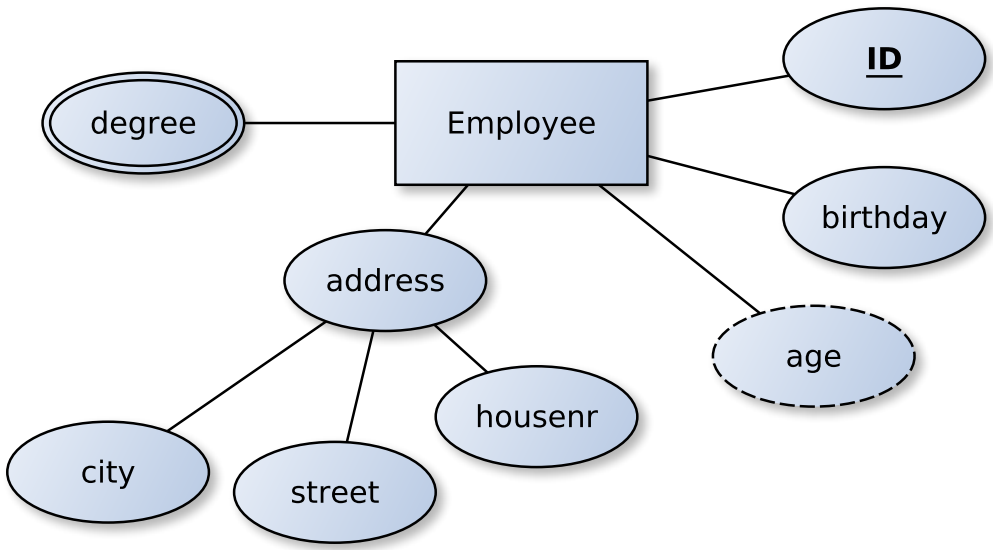


is equivalent to



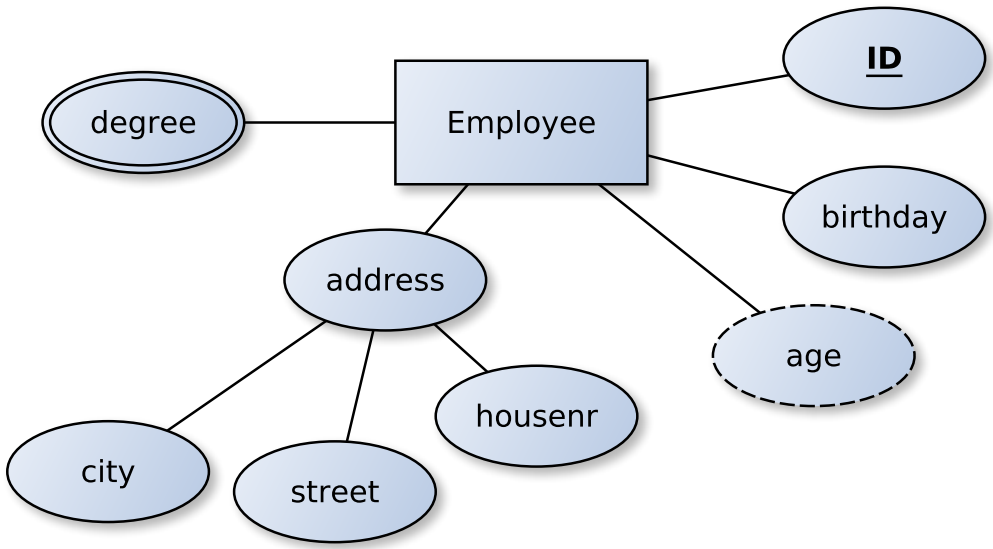
Attributes in Depth

An **entity type has attributes**. **Entities are described** by a **set of attributes**.



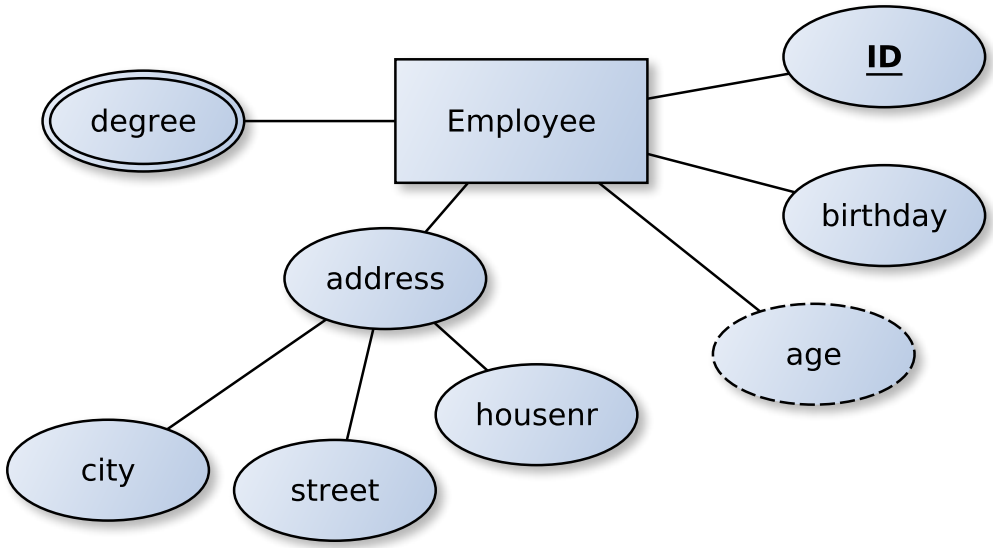
Types of Attributes 1

- **Simple Attributes**
  - atomic or indivisible values.
- **Composite attributes:**
  - non-atomic attributes. can be represented by a tree of ovals
- **Derived attributes:**
  - can be derived from other attributes. Are shown as a dashed oval.



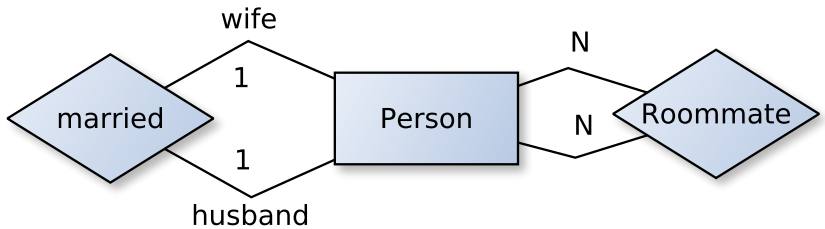
Types of Attributes 2

- **Single Value Attributes**
  - single value
  - Shown as a oval.
- **Multivalued attributes:**
  - can take a set of values.
  - Shown as a double oval.



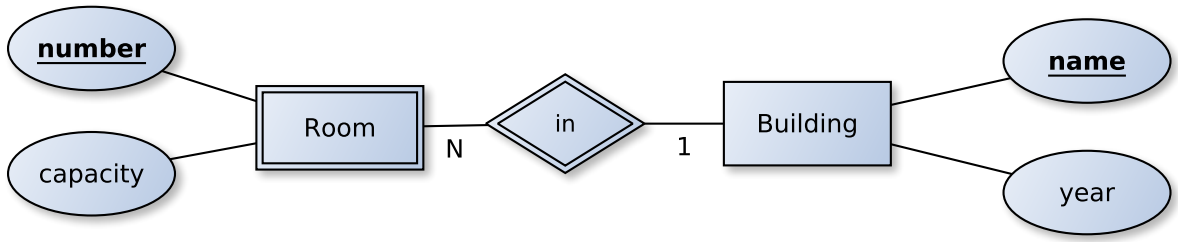
Roles in Relationships

- An entity may participate **more than once** in a relationship.
- we **label edges** to dinstinguish roles



- *married* relationship distinguishes roles husband and wife
- *roommate* relationship does not need roles.

Weak Entity Types



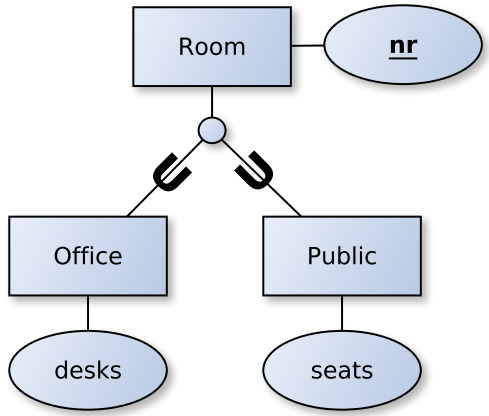
- A key of E may not come completely from its E attributes
  - but from the keys of other sets, linked to E by relationship sets.
- we call this a **weak entity type** and represent it with a double rectangle
- the connecting **identifying relationship** (1-N or 1-1) is represented with a double diamond.

Extensions to ER

- Basic ER Diagrams are a powerful modeling tool,
- to model additional concepts
- we introduce the **is-a** relation, **extended ER model (EER)**
  - Subclasses (Specialisation)
  - Superclasses (Generalisation)

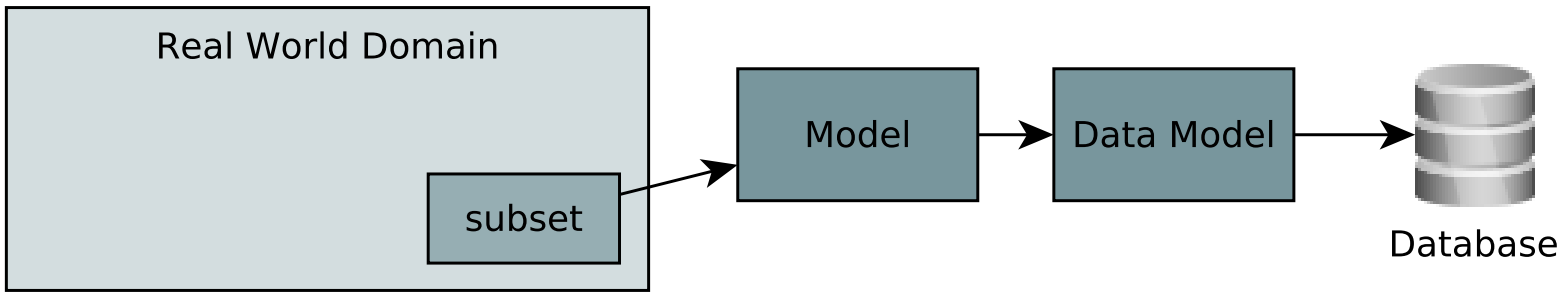
is-a Relationship

- The **is-a** relationship is a connection between entity types, which indicates
  - one entity type , which generalizes a concept (superclass)
  - one or many entity types, that specialize it (subclasses)
- a subclass inherits all attributes and relationships of a superclass



- we indicate **is-a** relationships with a line labeled with a *subset*-symbol.
- In the case of more than one subclass, these lines end at a circle connected to the superclass.

From E/R to to Relational Schema



E/R diagrams

- E/R model and the relational model are logical representations of real world scenarios.
- An E/R model can be converted to a relational database schema.

Entity/Relationship vs other models

	RA	**E/R**	**OO**
"thing" to be modeled	Tuple	Entity type	Object
set of similar "things"	Relation	Entity set	Class
relationship	Tuple?	Relationship type	Object?
set of similar relationships	Relation?	Relationship set	Class?
property of a "thing"	Attribute	Attribute	Field

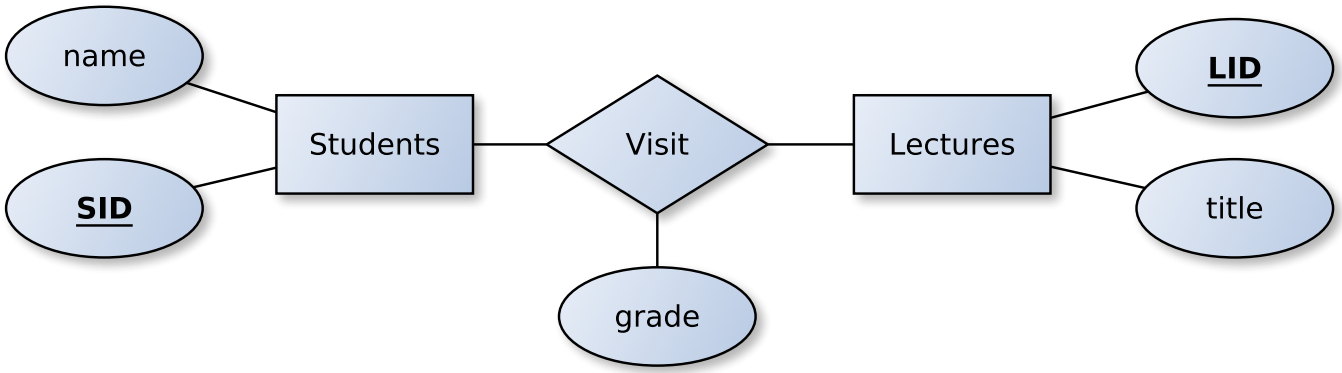
ER-to-Relational Mapping

1. Strong entity types
2. Weak entity types
3. 1:1 Relationships
4. 1:N and N:1 Relationships
5. M:N Relationships
6. Multi-valued Attributes
7. n-ary Relationships

Elmasri &Navathe: Fundamentals of Database Systems, Chap. 7

Translating Strong Entity Types

- **Strong entity types** correspond to **relations** (tables)

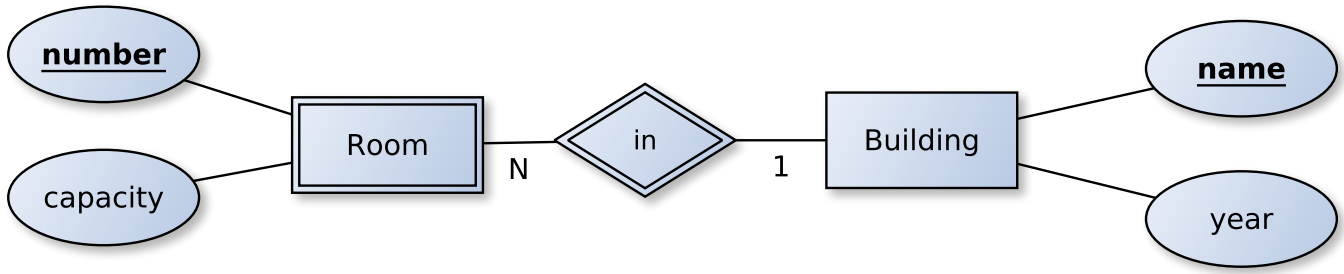


Student (SID, name) *Lecture (LID, title)*

- Attributes map to columns
- each leaf of a composite attribute tree maps to a column
- key attributes to key columns

Translating Weak Entity Types

- **mapped to relations** like strong entity types.



Room (<u>number</u>, <u>name</u>, capacity) *Building (<u>name</u>, year)*

- referenced key attributes become part of the weak entities key columns

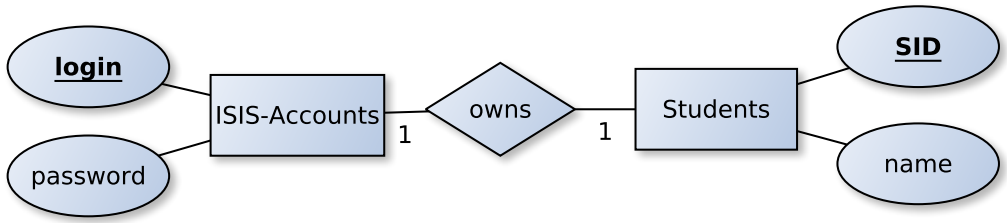
Translating Relationship Types

there are three approaches:

1. foreign key approach
2. merged relation approach
3. relationship relation approach

Foreign Key Approach

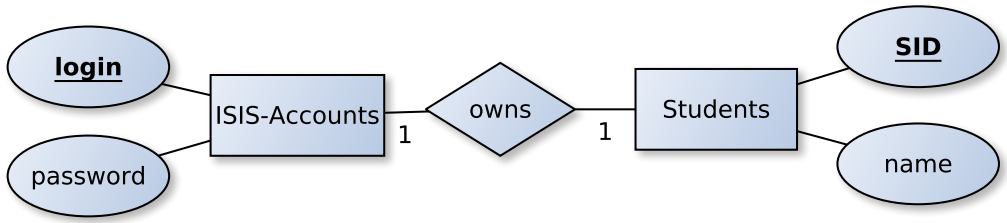
- Identify the entities S and T participating in the relationship type R
- Include the key of T as a foreign key in S (or vice versa).
- Simple attributes of R are included in S



Student (SID, name, login) *ISIS-Account (login, password)*

Merged Relation Approach

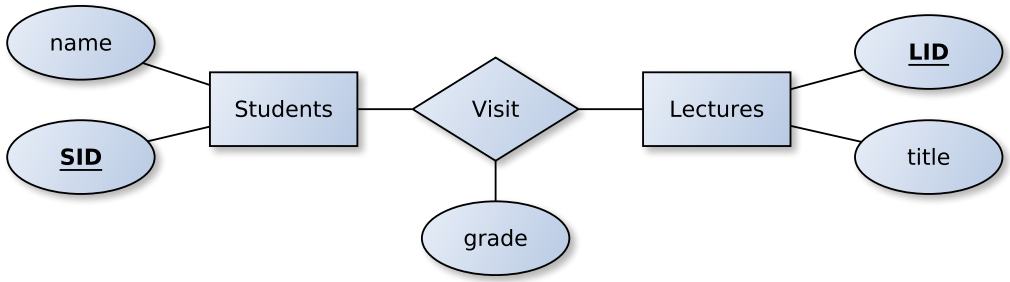
- completely integrates participating entities in one relation
- use it for 1 to 1 relationships, that are total
- if they are not total, null values have to be used



Student (SID, name, login, password)

Relationship Relation Approach

- introduce a new relation to model the relationship
- also called cross-referencing
- the only possible way to represent N to M relationships.

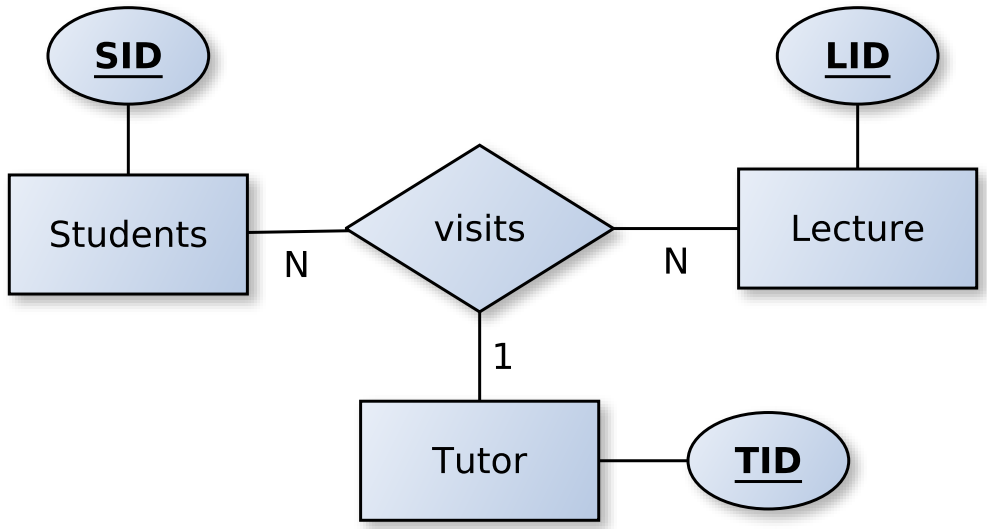


Student (<u>SID</u>, name) *Lecture (<u>LID</u>, title)*

Visits (<u>SID</u>, <u>LID</u>, grade)

Translating Nonbinary Relationships

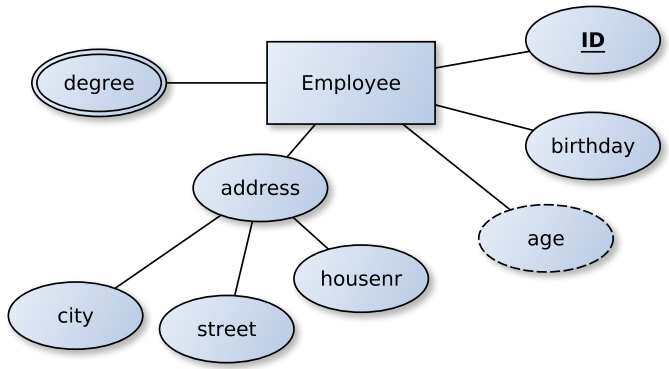
- create a relationship relation
- add the keys of all participating entity types as foreign keys
- add the attributes of the relationship type.



(SID, LID, TID)

Translating Multivalued Attributes

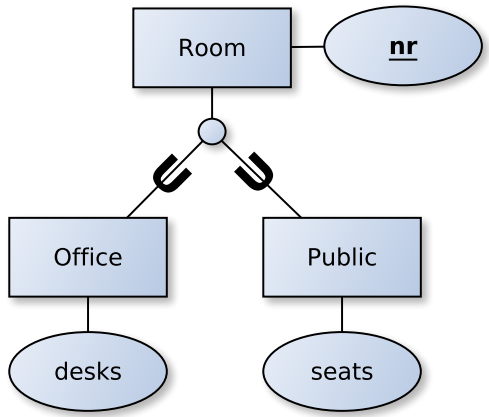
- For a relation S, create a new relation R for each multivalued attribute it has.
- Use the key of S as a foreign key in R



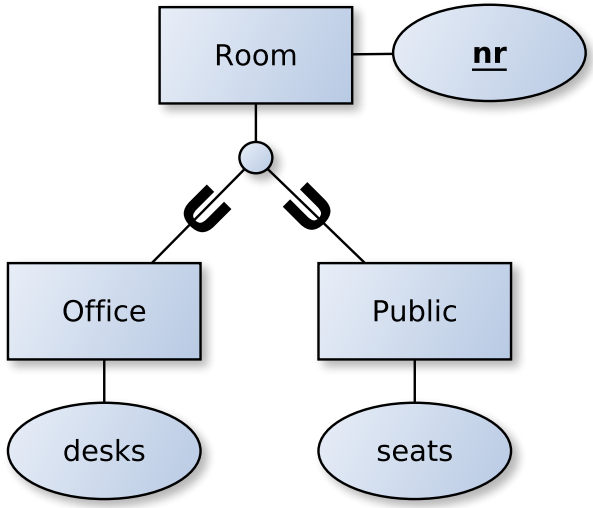
Degree (EMPID, name)

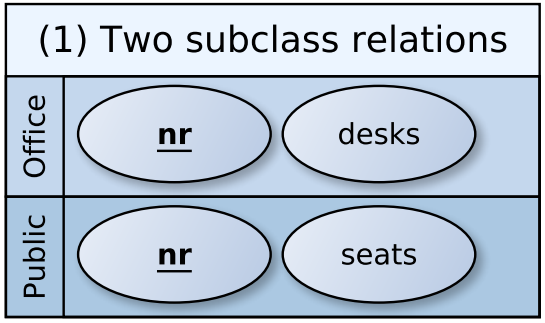
Translating is-a Relationships

- There are three main ways to translate is-a relationships:
  1. **Subclass relations:** create a relation for each subclass
  2. **Superclass and subclass relations:** create a relation with all common attributes and one for each specialization
  3. **One relation approach:** add an attribute that indicates the subclass and all subclass attributes



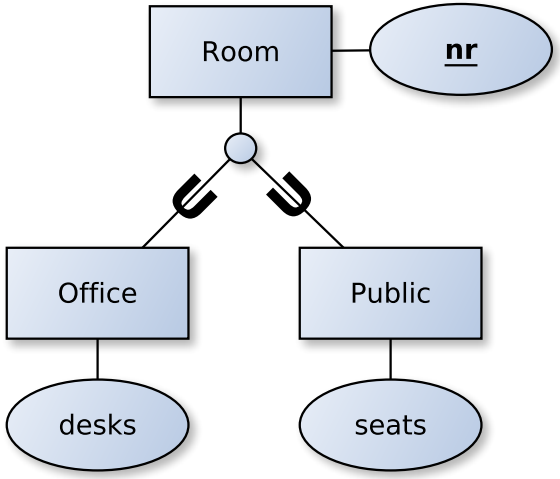
is-a: Subclass Relations

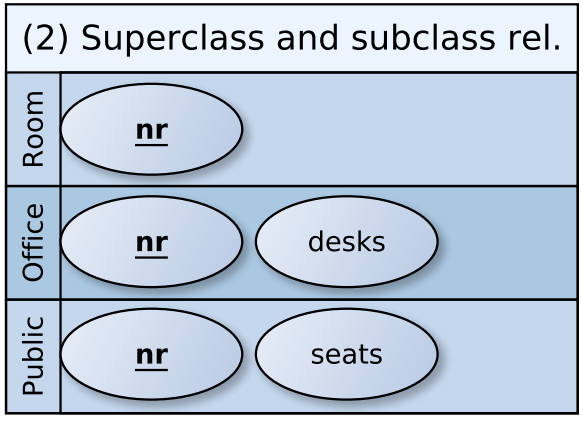




- **Pro:** all attributes of a subclass are found in one table
- **Contra:** no relation with all the rooms

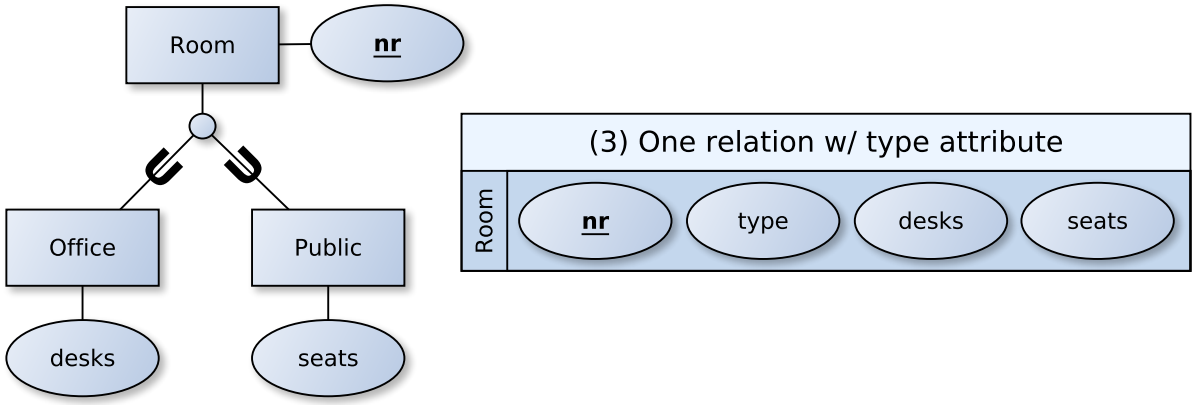
is-a: Super and Subclass Relations





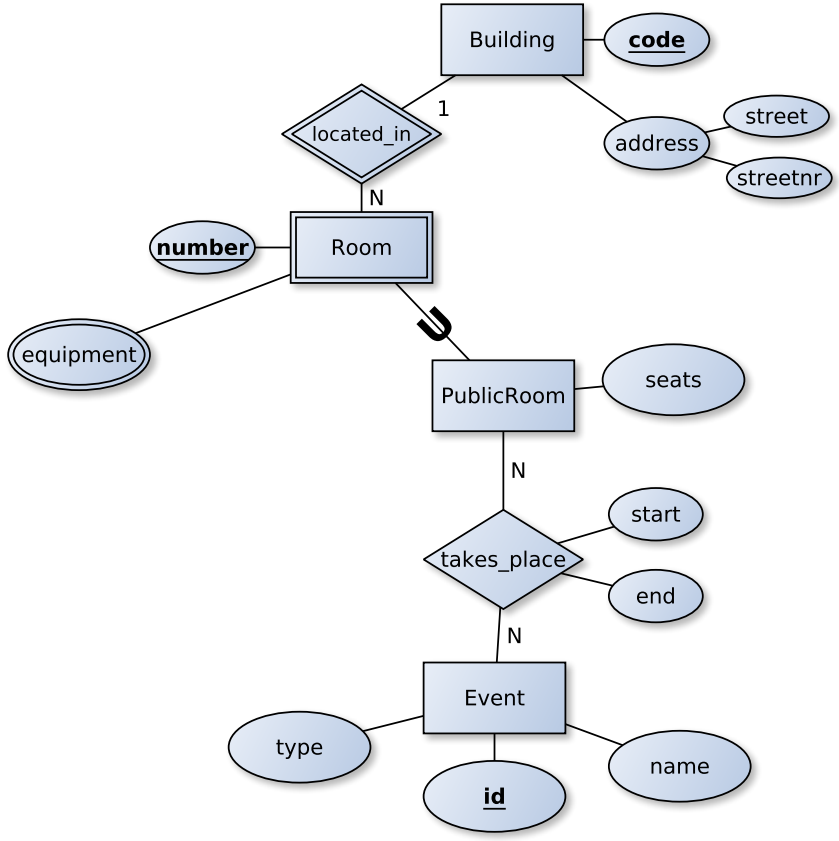
- **Pro:** all rooms in one relation
- **Contra:** attributes of subclasses are scattered across relations

is-a: One Relation

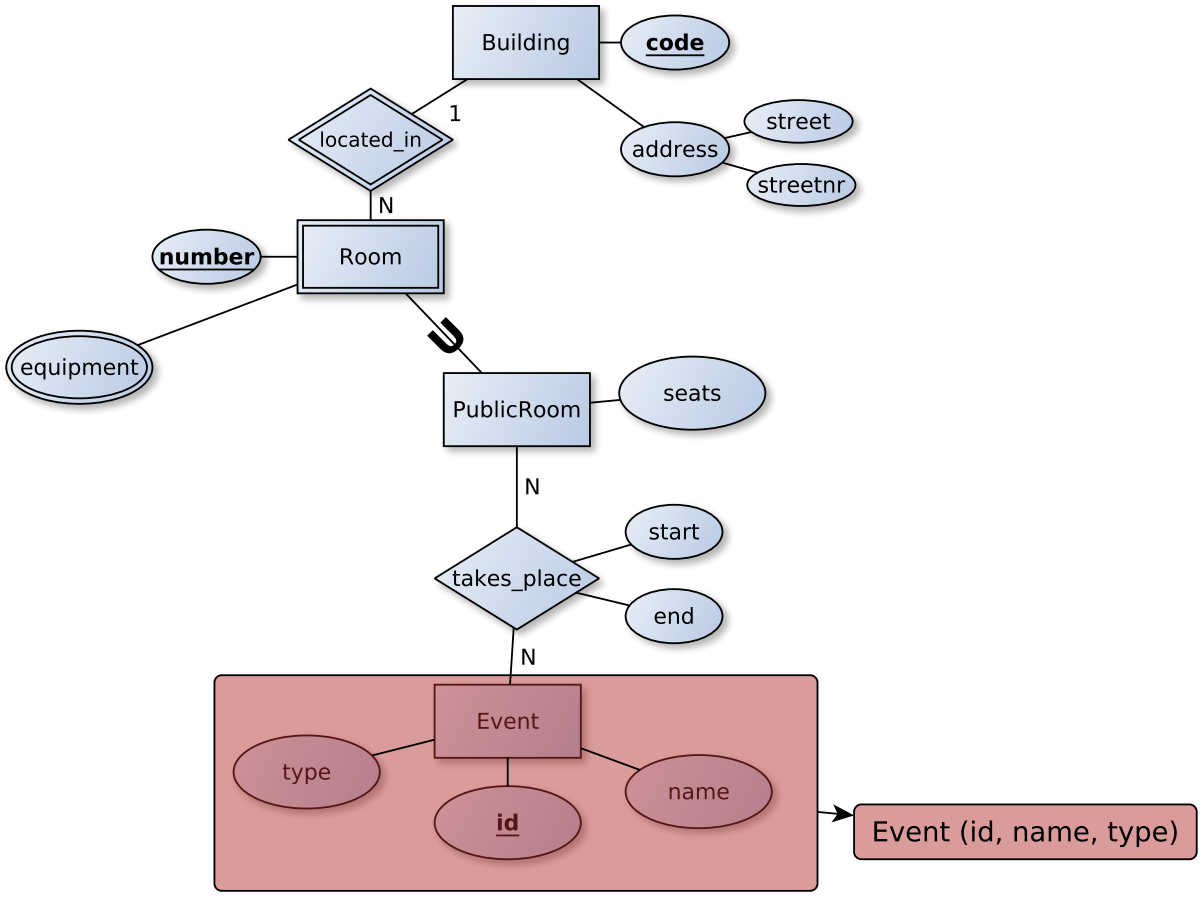


- **Pro:** everything in one table
- **Contra:** introduces NULL values. gets complicated in complex cases.

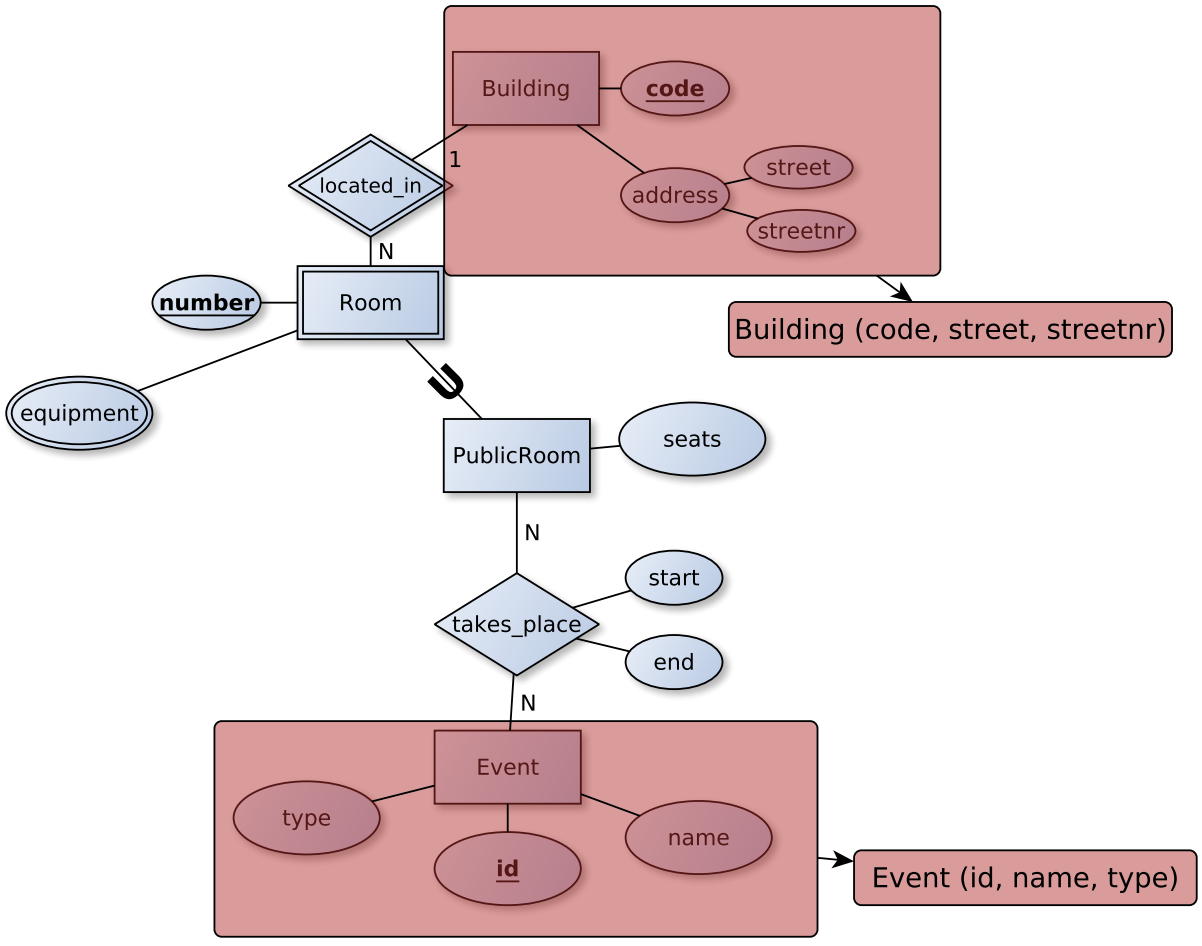
Translation Example



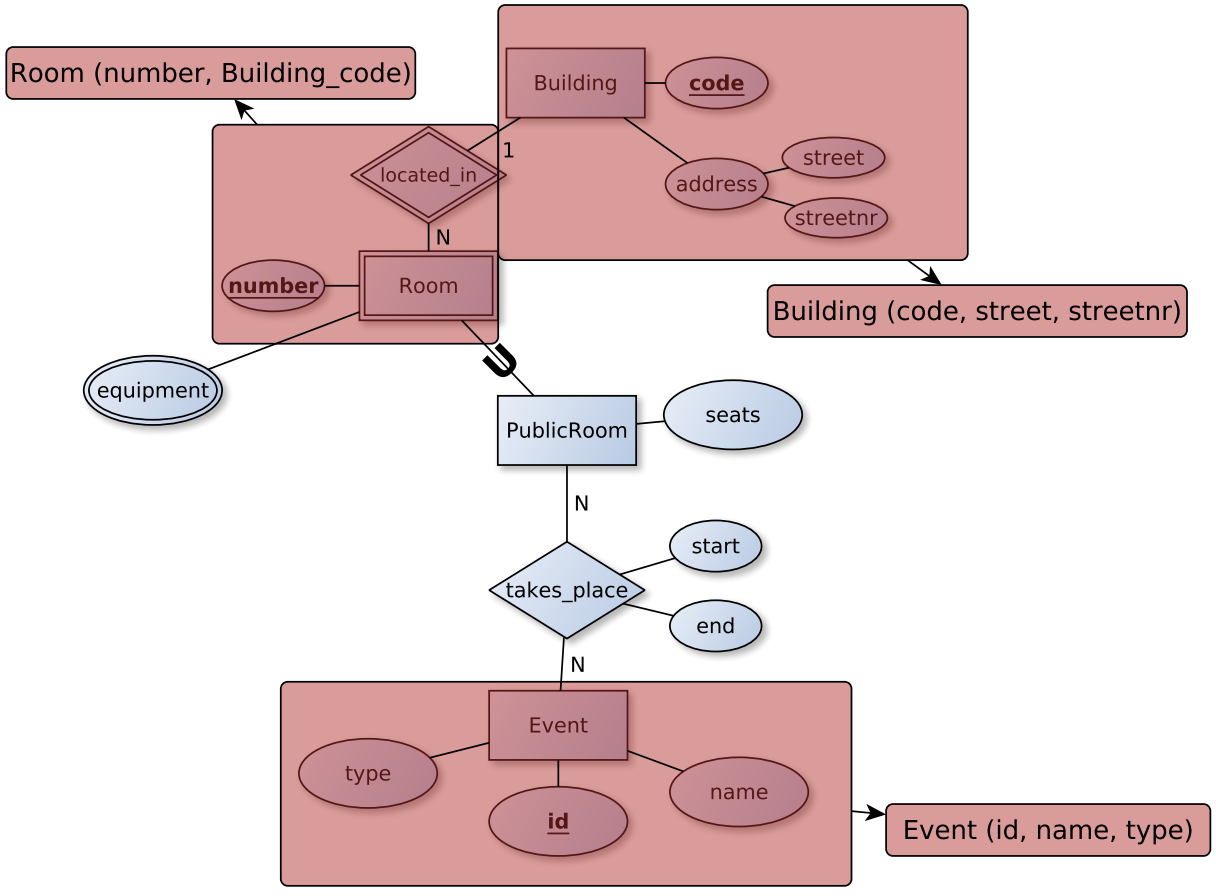
1: Entity types to relations



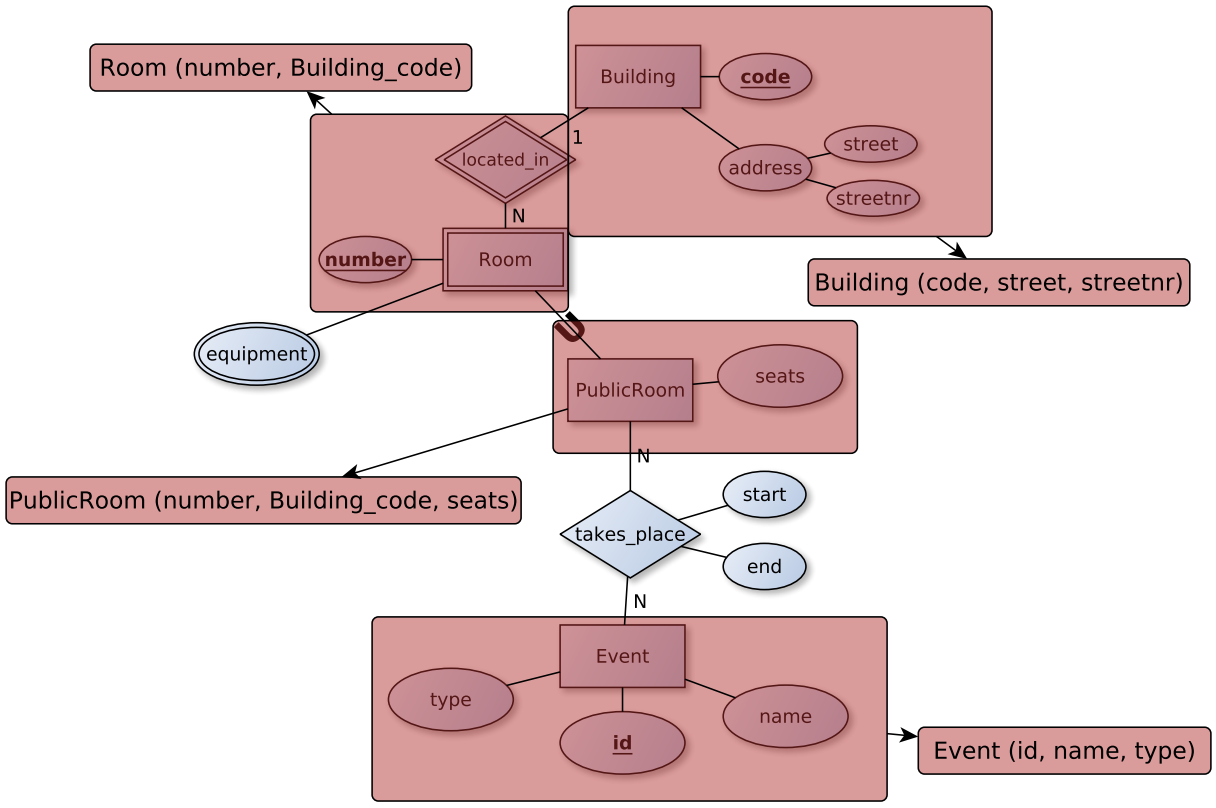
2: Entity types to relations



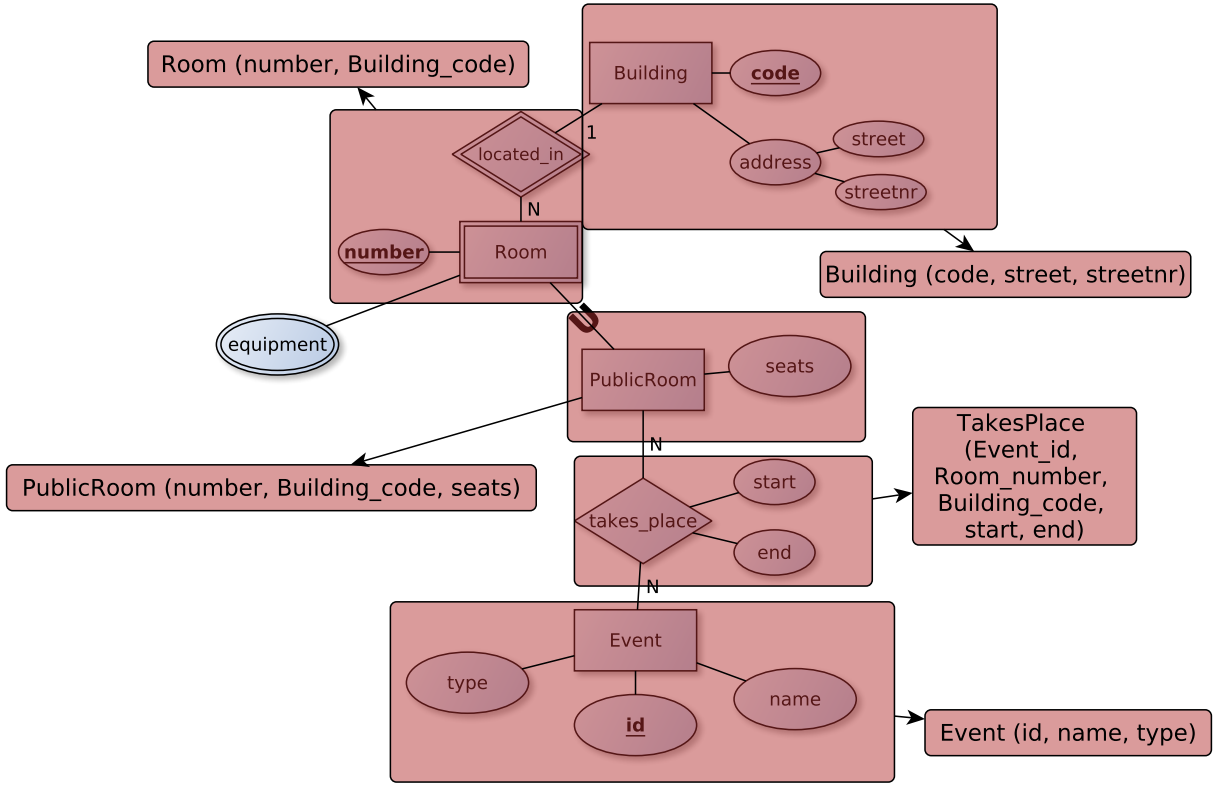
3: Weak entity types to relations



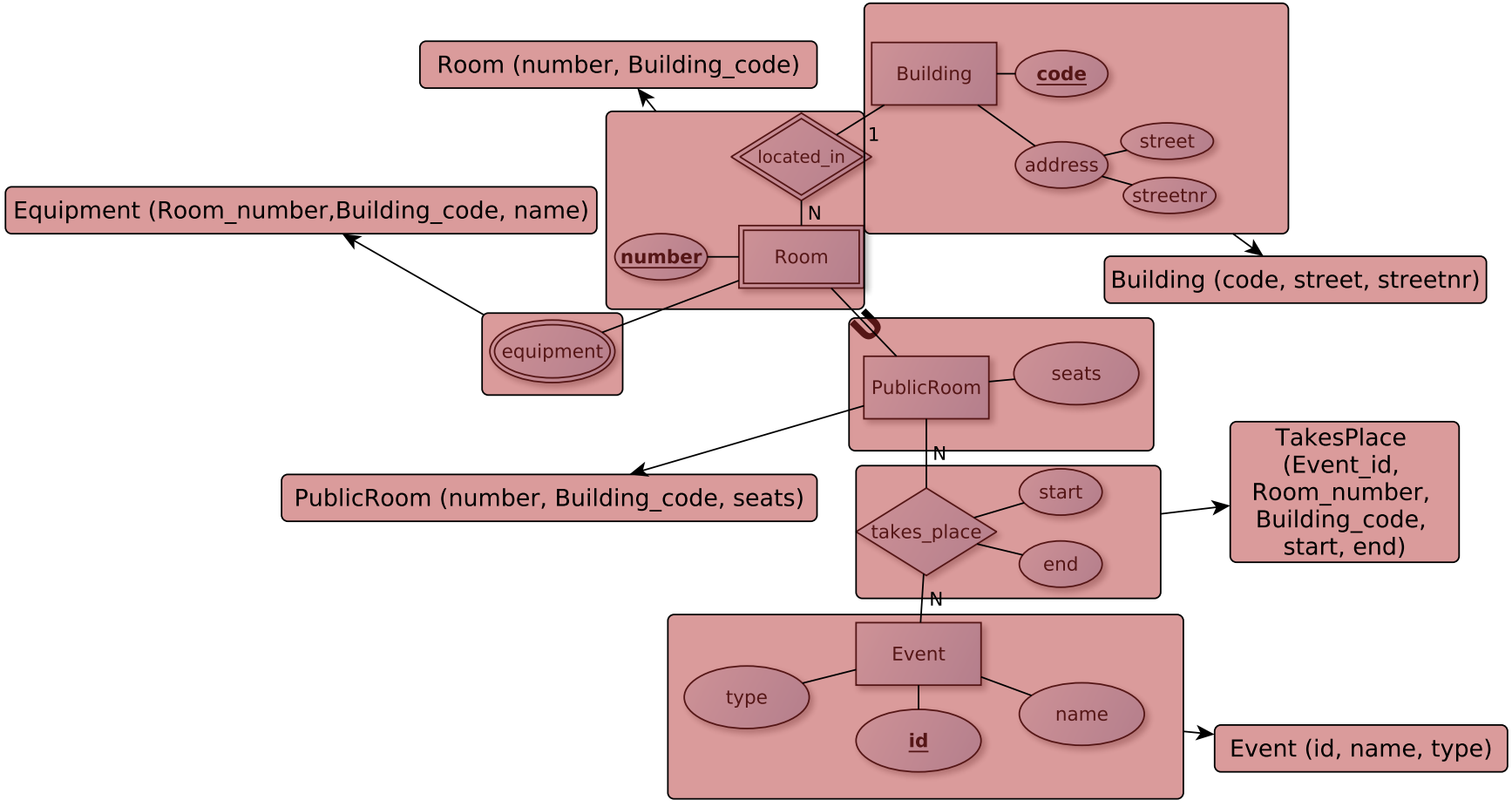
4: (EER) Translate is-a relationships



5: Translate relationship sets



6: Translate multivalued attributes



Modeling: Conclusion

- The ER Model helps us to translate real-world concepts to relations representing it.
- The real world is complex, always represent only what is needed for your projecttypes
- try not to think about relations when designing it
- one exception: try to model without using non-binary relationship types

  A good design is faithful to the constraints of the application, but not overzealous