

CS354: Database

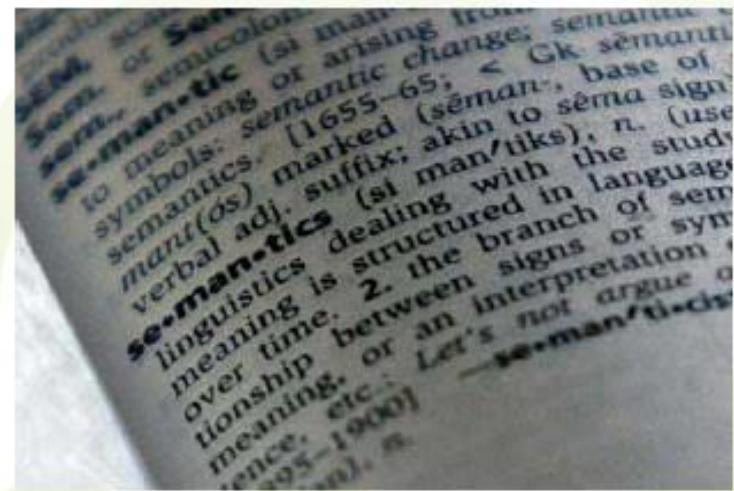
Data Modeling

- Introduction
- Data Models
- Phases of DB Design
- Basic ER Modeling
 - Chen notation
 - Alternative notations
- Example



Data Models

- In databases, the data's specific **semantics** are very important
 - What is described?
 - What values are reasonable/correct?
 - What data belongs together?
 - What data is often/rarely accessed?



Data Models

- Example: Describe the “age” of a person
 - Semantic definition:
The number of years elapsed since a person’s birthday
 - Integer data type
 - Always: $0 \leq \text{age} \leq 120$
 - Connected to the person’s name, passport id, etc.
 - May often be retrieved, but should be protected
 - ...



Data Models

- A **data model** is an abstract model that describes how data is represented and accessed
 - Examples: network model, relational model, object-oriented model, ...
 - **Warning:** The term “data model” is ambiguous
 - A data model **theory** is a formal description of how data may be structured and accessed
 - A data model **instance** or **schema** applies a data model theory to create an instance for some particular application

Data Models

- A **data model** needs three parts:
 - Structural part
 - **Data structures** which are used to create databases representing the objects modeled
 - Integrity part
 - Rules expressing the **constraints** placed on these data structures to ensure structural integrity
 - Manipulation part
 - Operators that can be applied to the data structures, to **update** and **query** the data contained in the database



Data Models

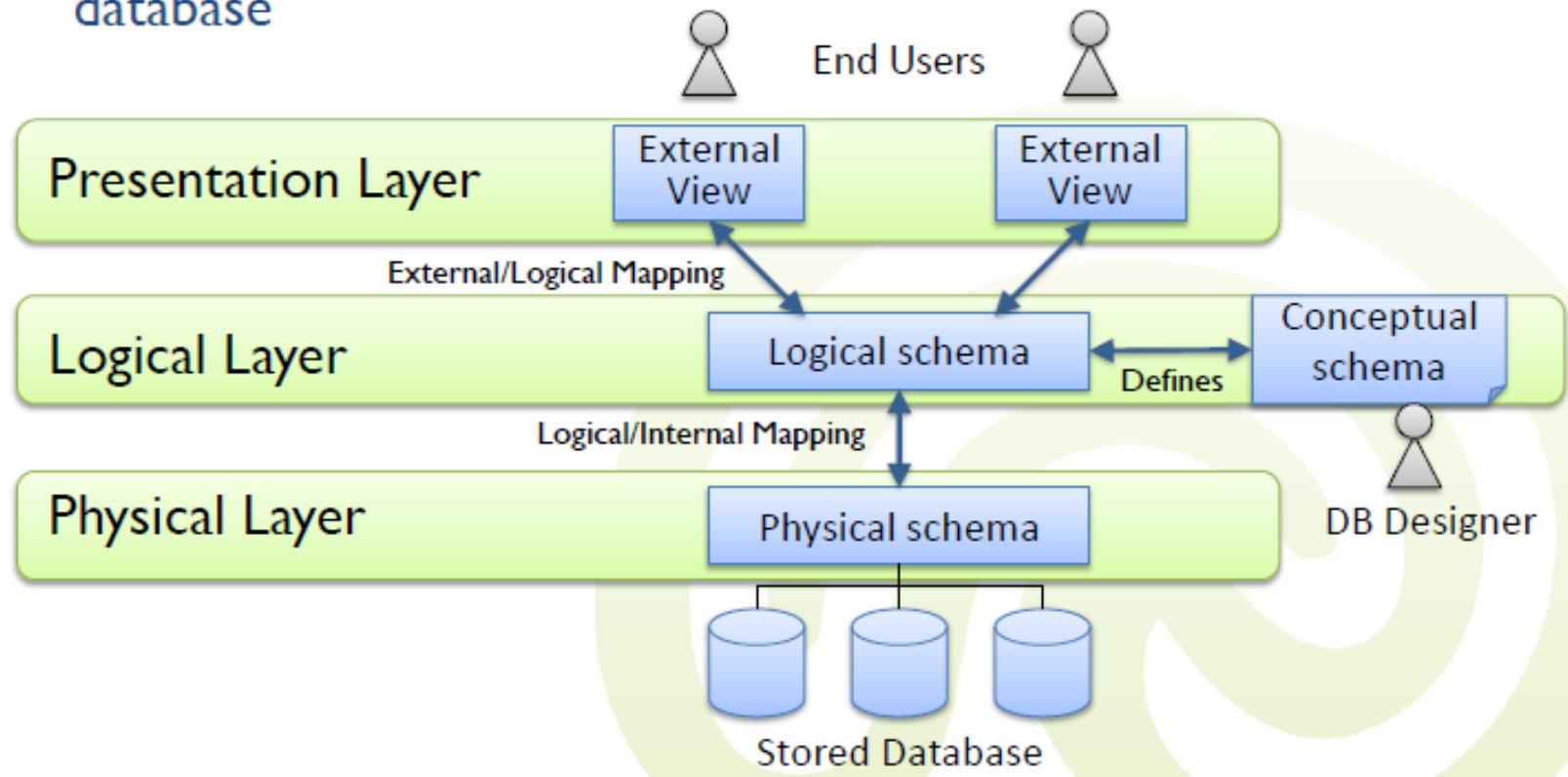
- Different categories of data models exist:
 - **High-level** or **conceptual** models
 - provide concepts that are close to the way many users perceive data
 - **Low-level** or **physical** data models
 - provide concepts that describe the details of how data is stored in the computer
 - **Representational** or **logical** data models
 - provide concepts that may be understood by end users but that are not too far removed from the way data is organized within the computer

Data Models

- Data models are instanced by **schemas**
 - A **conceptual schema** describes the semantics of a domain
 - What facts or propositions hold in this domain?
 - A **logical schema** describes the semantics, as represented by a particular data manipulation technology
 - Tables and columns, object-oriented classes, XML elements, ...
 - A **physical schema** describes the physical means by which the data is stored
 - Partitions, tablespaces, indexes, ...

Three-layer architecture

- ... also called ANSI-SPARC Architecture ...
 - separates the user applications and views from the physical database



Three-layer architecture

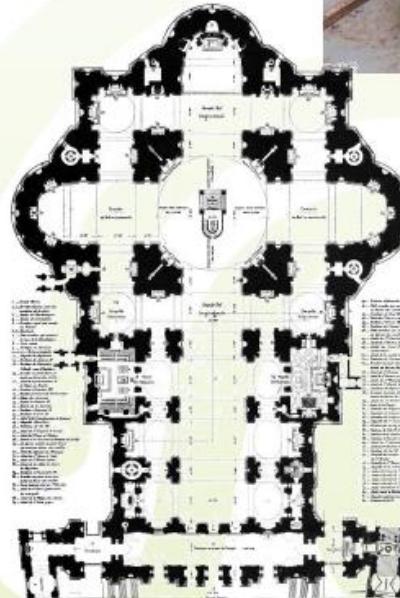
- Caution:
 - The **logical layer** in the ANSI-SPARC Architecture is often called **conceptual layer**
 - It is described using a **logical** or **representational** data model, but often based on a **conceptual schema design** in a high-level data model
 - The **external views**
 - Are also typically implemented using a **logical** data model and are possibly based on a **conceptual schema design** in a high-level data model

Data Models

- Why do we need three kinds of instances then?
- To maintain **independence!**
 - **Physical independence** means that the storage design can be altered without affecting logical or conceptual schemas
 - Logically, it does not matter where exactly the data about a person's age is stored, it is still the same data
 - **Logical independence** means that the logical design can be altered without affecting the data semantics
 - It does not matter whether a person's age is directly stored or computed from the person's birth date

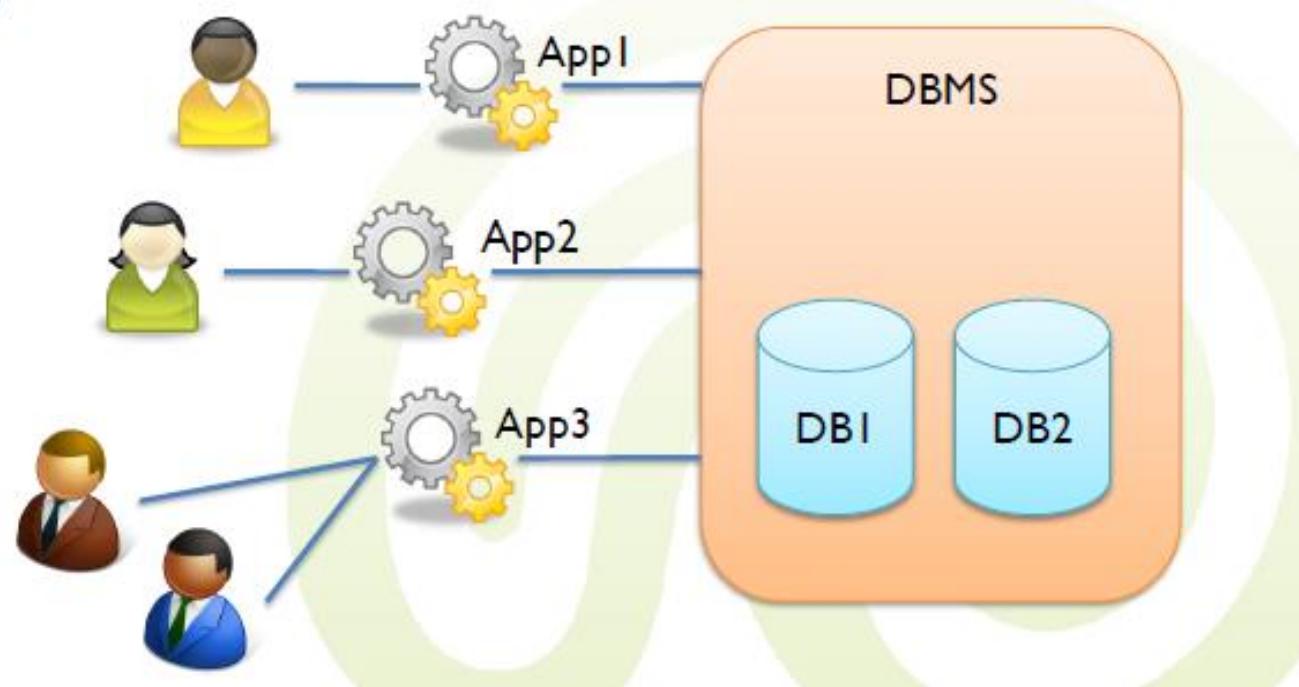
Data Modeling

- Introduction
- Data Models
- Phases of DB Design
- Basic ER Modeling
 - Chen notation
 - Alternative notations
- Example



Database Applications

- **Database applications** consist of
 - **Database instances** with their respective **DBMS**
 - associated **application programs** interfacing with the users



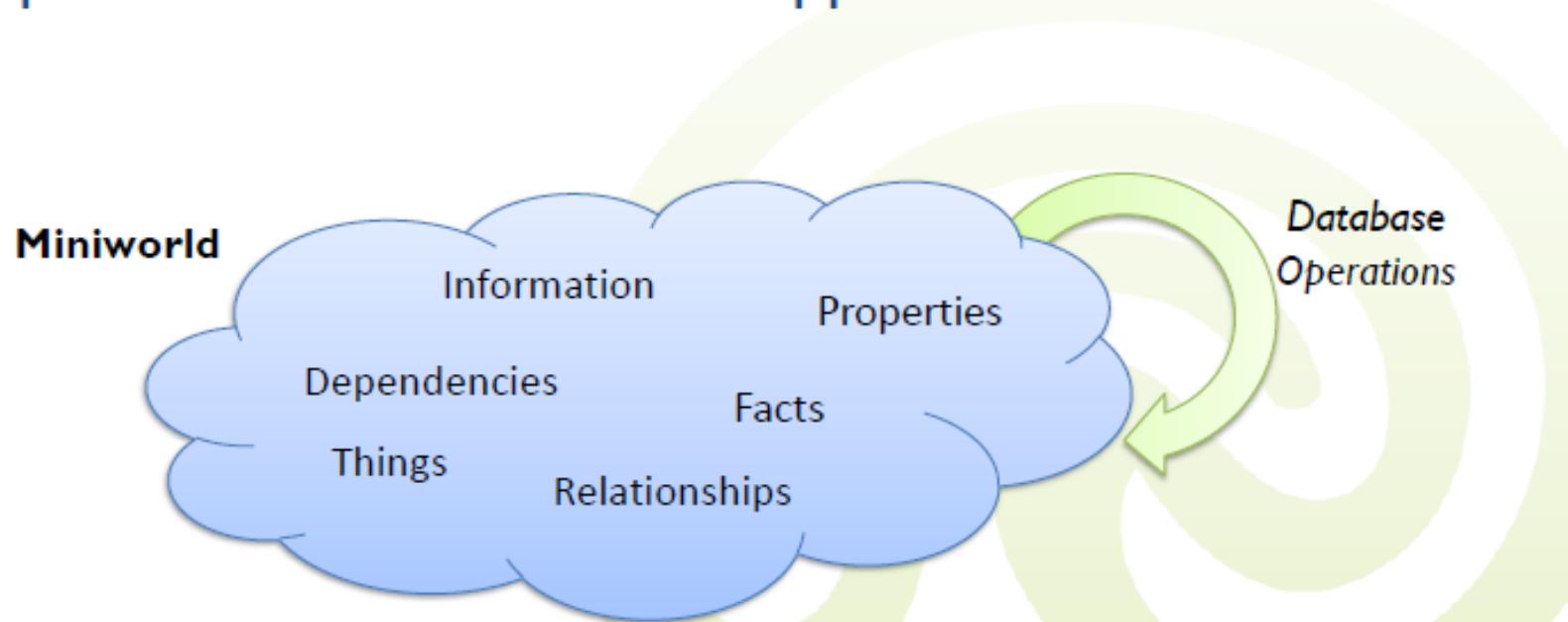
Database Applications

- Planning and developing application programs traditionally is a **software engineering** problem
 - Requirements Engineering
 - Conceptional Design
 - Application Design
 - ...
- Software engineers and data engineers cooperate tightly in planning the need, use and flow of data
 - **Data modeling**
 - **Database design**

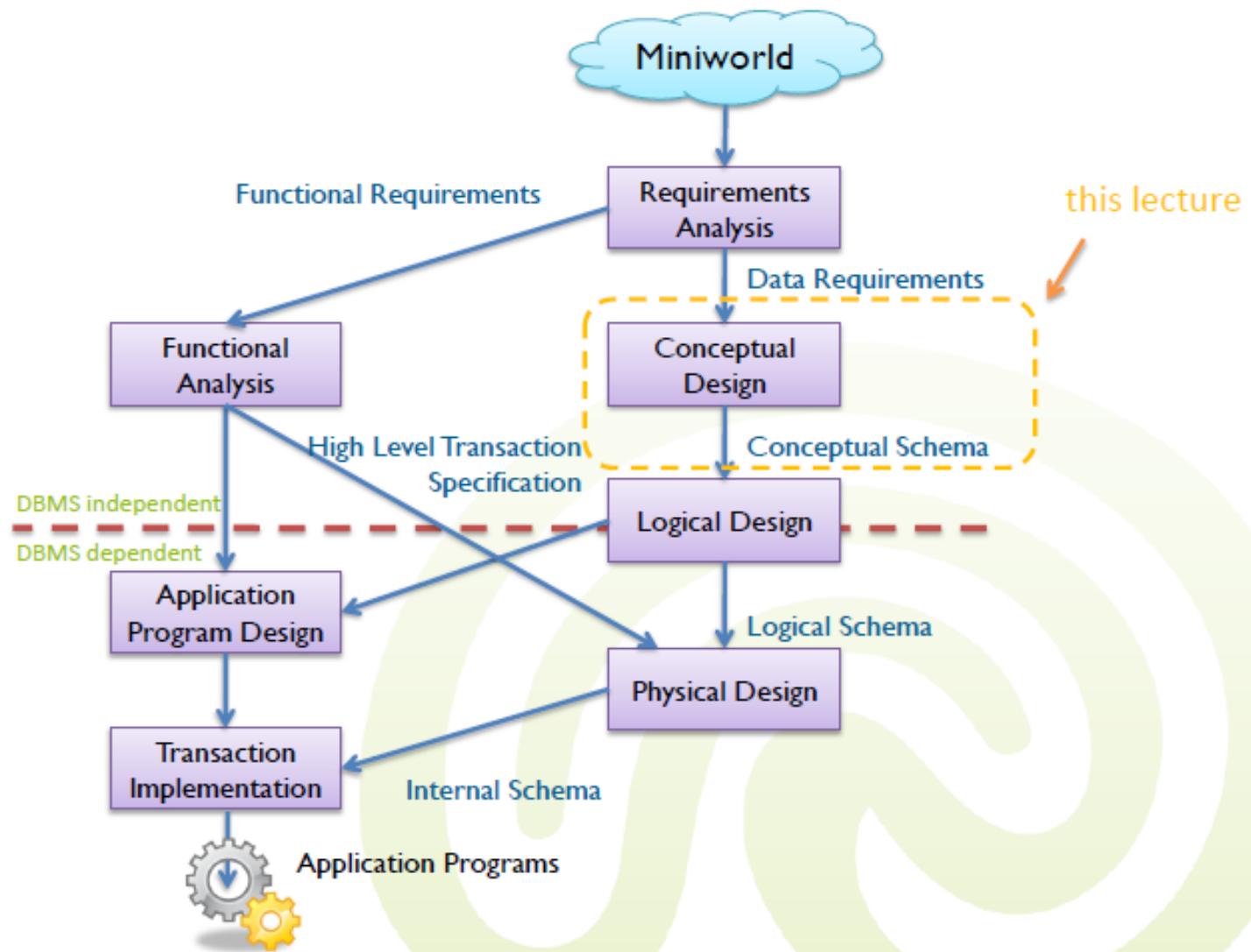


Universe of Discourse

- DB Design models a **miniworld** into a formal representation
 - Restricted view on the real world with respect to the problems that the current application should solve



Phases of DB Design



Phases of DB Design

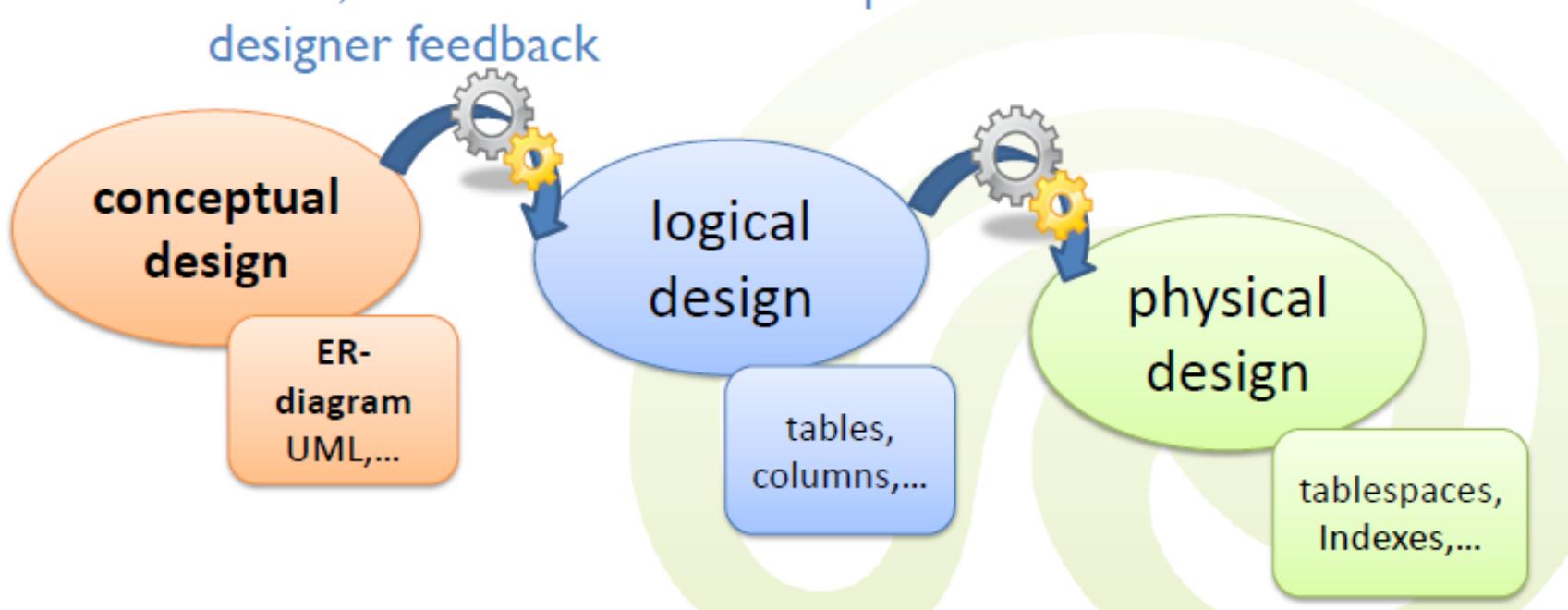
- **Requirements Analysis**
 - Database designers interview prospective **users** and **stakeholders**
 - **Data requirements** describe what kind of data is needed
 - **Functional requirements** describe the operations performed on the data
- **Functional Analysis**
 - Concentrates on describing **high-level** user operations and transactions
 - Does also not contain implementation details
 - Should be matched versus conceptual model

Phases of DB Design

- **Conceptual Design**
 - Transforms data requirements to **conceptual model**
 - Conceptual model describes data entities, relationships, constraints, etc. on **high-level**
 - Does not contain any implementation details
 - Independent of used software and hardware
- **Logical Design**
 - Maps the conceptual data model to the logical data model used by the DBMS
 - e.g. relational model, hierarchical model, ...
 - Technology independent conceptual model is adapted to the used DBMS software
- **Physical Design**
 - Creates internal structures needed to efficiently store/manage data
 - Table spaces, indexes, access paths, ...
 - Depends on used hardware and DBMS software

Phases of DB Design

- While modeling the data, 3 design phases have to be completed
 - The result of one phases serves as input for the next phase
 - Often, automatic transition is possible with some additional designer feedback



Data Modeling

- Introduction
- Data Models
- Phases of DB Design
- Basic ER Modeling
 - Chen notation
 - Alternative notations
- Example



ER Modeling

- Traditional approach to **Conceptual Modeling**
 - **Entity-Relationship Models (ER-Models)**
 - Also known as Entity-Relationship Diagrams (ERD)
 - Introduced in 1976 by **Peter Chen**
 - Graphical representation
- Top-Down-Approach for modeling
 - Entities and Attributes
 - Relationships
 - Constraints
- Some derivates became popular
 - ER Crow's Foot Notation (Bachman Notation)
 - ER Baker Notation
 - Later: Unified Modeling Language (UML)



ER - Entities

- **Entities**

- An entity represents a “**thing**” in the real world with an independent existence
 - An entity has an own identity and represents just one thing
- Example: a car, a savings account, my neighbor’s house, the cat “Snowflake”, a product, ...

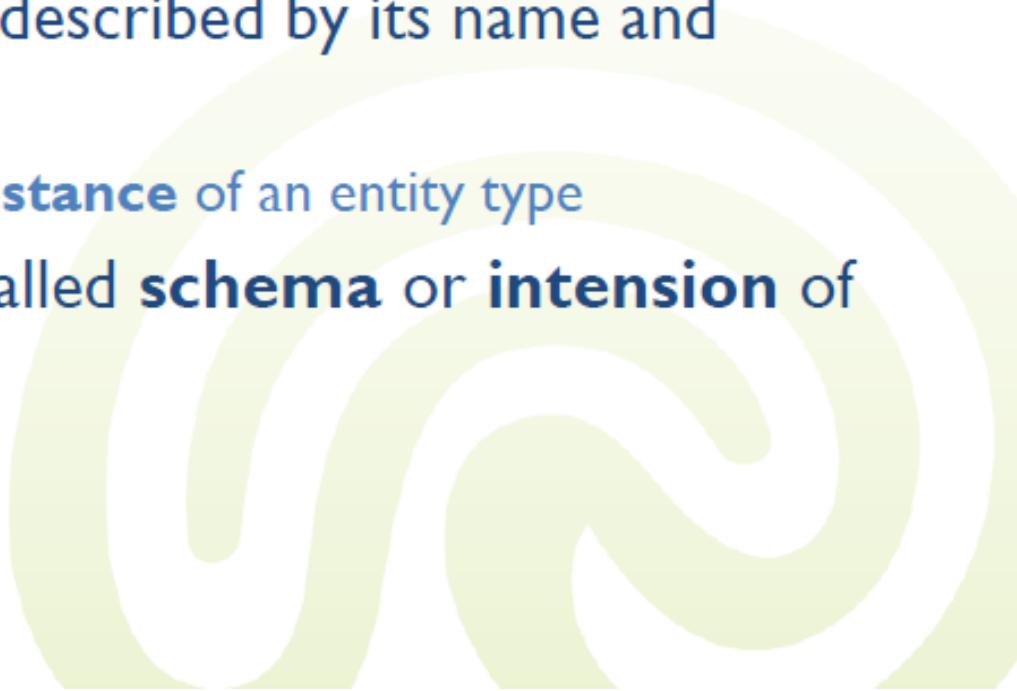


ER - Attributes

- **Attributes**
 - A **property** of an entity, entity type or a relationship type.
 - Example: name of an employee, color of a car, balance of an account, location of a house,...
 - Attributes can be classified as being:
 - **simple** or **composite**
 - **single-valued** or **multi-valued**
 - **stored** or **derived**
 - Example: name of a cat is simple, single-valued, and stored

ER – Entity Types

- **Entity Types** are sets of entities sharing the same characteristics or attributes
 - Each entity within the set has its own values
 - Each entity type is described by its name and attributes
 - Each entity is an **instance** of an entity type
 - Describes the so called **schema** or **intension** of similar entities



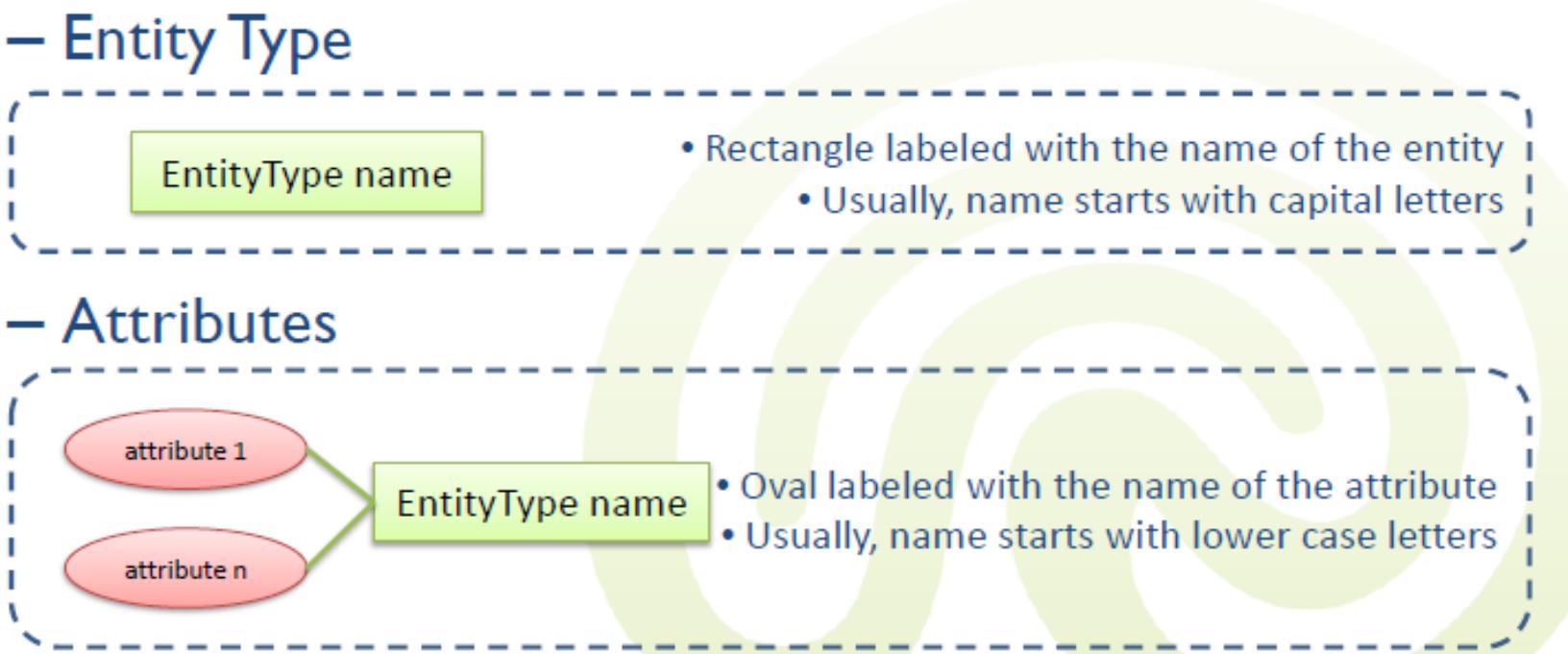
ER – Entity Sets

- An **Entity Set** is the collection of *all* entities of a given entity type
 - Entity sets often have the same name as the entity type
 - Cat may refer to the entity type as well as to the set of all Cat entities (sometimes also plural for the set: Cats)
 - Also called the **extension** of an entity type (or **instance**)



ER Diagrams

- ER diagrams represent **entity types** and **relationships** among them, not single entities
- Graphical Representation
 - Entity Type



ER Diagrams

- Textual Representation
 - Entity Types
 - Written as: EntityName (attribute1, attribute2, ...)
 - Entity
 - Written as: (value of attribute1, value of attribute2, ...)
- Example
 - Entity Type Cat
 - Cat (name, color)
 - Entity Set Cats
 - (Fluffy, black-white)
 - (Snowflake, white)
 - (Captain Hook, red)
 - ...



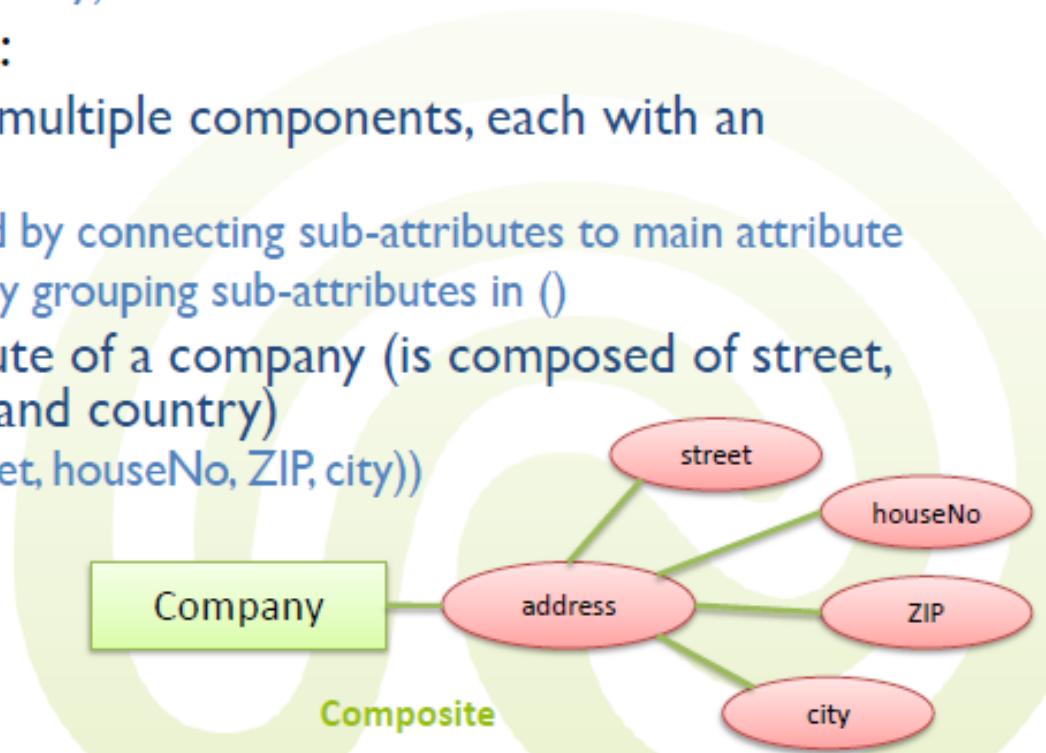
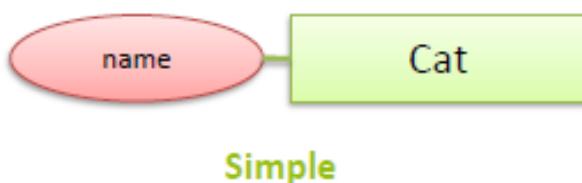
ER – Composite Attributes

- **Simple Attribute:**

- Attribute composed of a single component with an independent existence
- Example: name of a cat, salary of an employee,...
 - Cat (name), Employee(salary),...

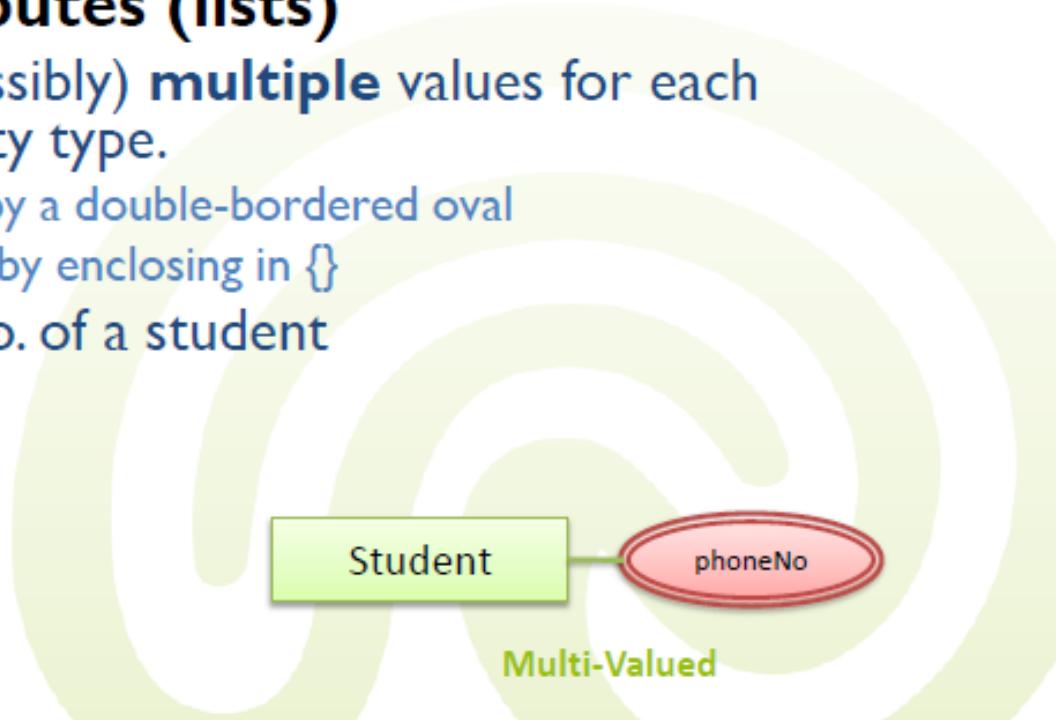
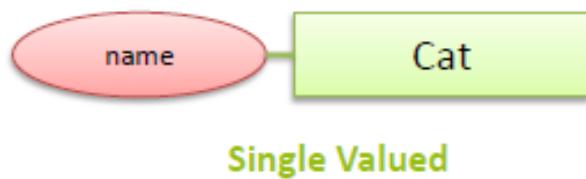
- **Composite Attribute:**

- Attribute composed of multiple components, each with an independent existence
 - Graphically represented by connecting sub-attributes to main attribute
 - Textually represented by grouping sub-attributes in ()
- Example: address attribute of a company (is composed of street, house number, ZIP, city, and country)
 - Company (address(street, houseNo, ZIP, city))



ER Multi-Valued Attributes

- **Single-Valued Attribute**
 - Attribute holding a **single** value for each occurrence of an entity type
 - Example: name of a cat, registrationNo. of a student
- **Multi-Valued Attributes (lists)**
 - Attribute holding (possibly) **multiple** values for each occurrence of an entity type.
 - Graphically indicated by a double-bordered oval
 - Textually represented by enclosing in {}
 - Example: telephoneNo. of a student
 - Student ({phoneNo})



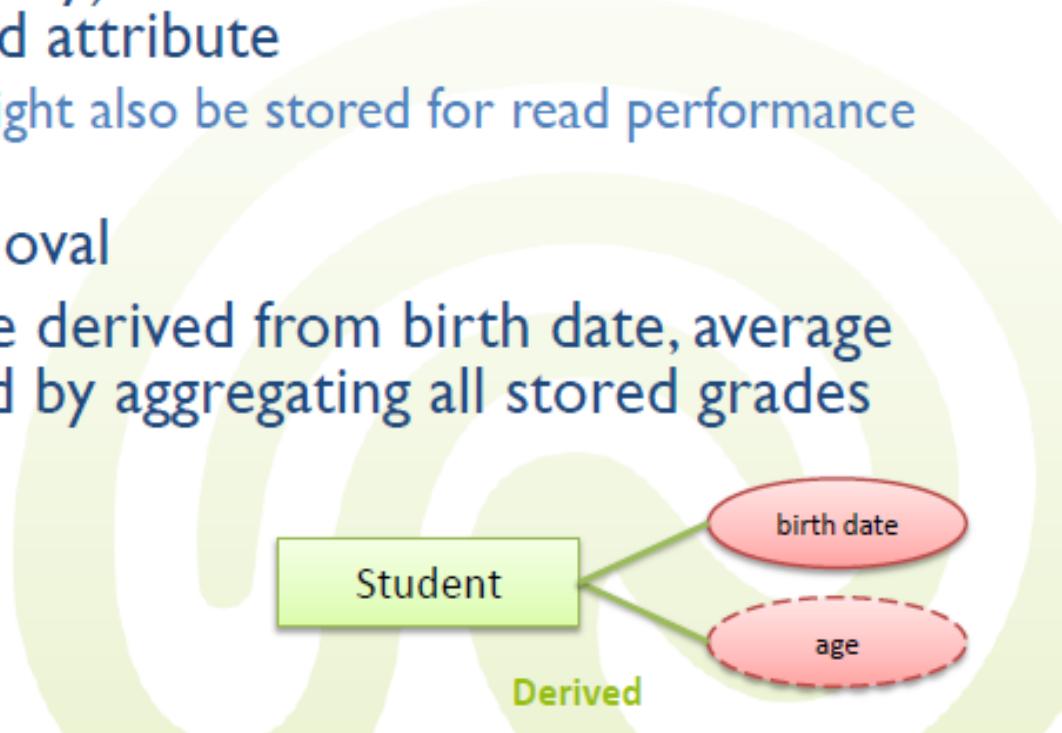
ER – Derived Attributes

- **Stored Attribute**

- The attribute is directly stored in the database

- **Derived Attribute**

- The attribute is (usually) not stored in the DB but derived from an other, stored attribute
 - In special cases, it might also be stored for read performance reasons
 - Indicated by dashed oval
 - Example: Age can be derived from birth date, average grade can be derived by aggregating all stored grades



ER - Keys

- Entities are **only** described by attribute values
 - Two entities with identical values cannot be distinguished (no OIDs, row IDs, etc.)
- Entities (usually) must be distinguishable
- Identification of entities with **key attributes**
 - Value combination of key attributes is **unique** within **all possible extensions** of the entity types
 - Key attributes are indicated by underlining the attribute name

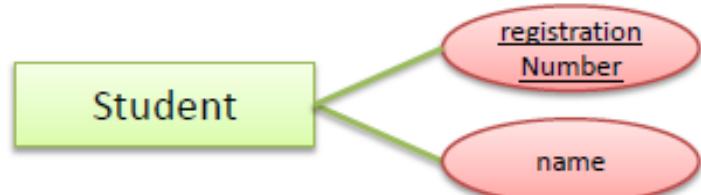


ER - Keys

- Key attribute examples

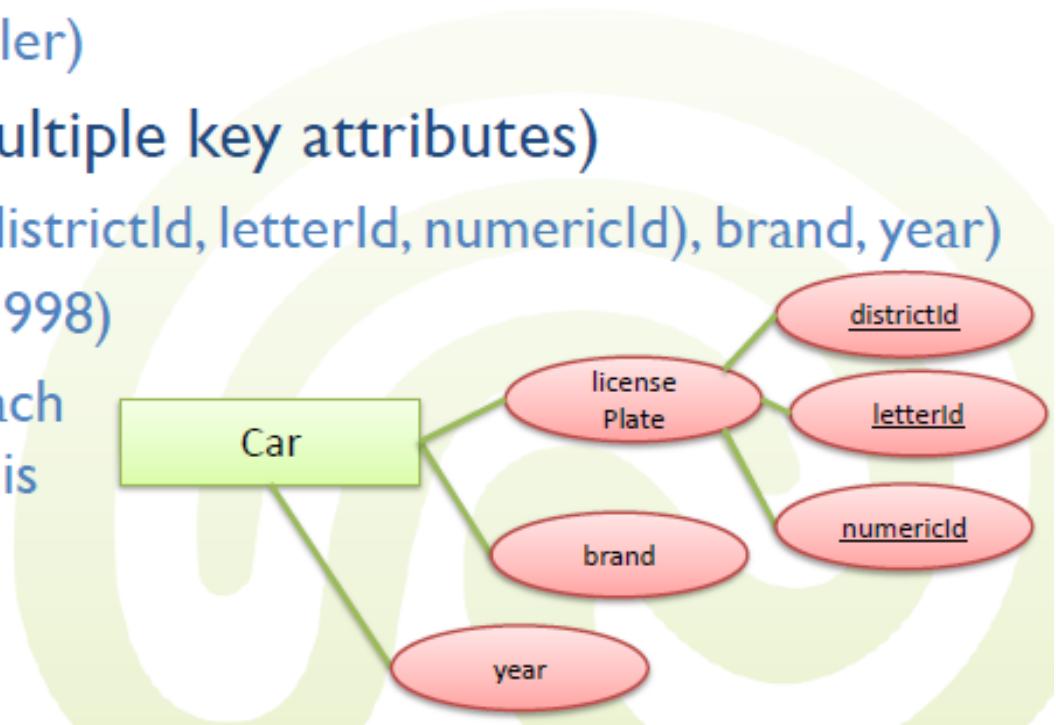
- Single key attribute

- Student (registrationNumber, name)
 - (432451, Hans Müller)



- Composite key (multiple key attributes)

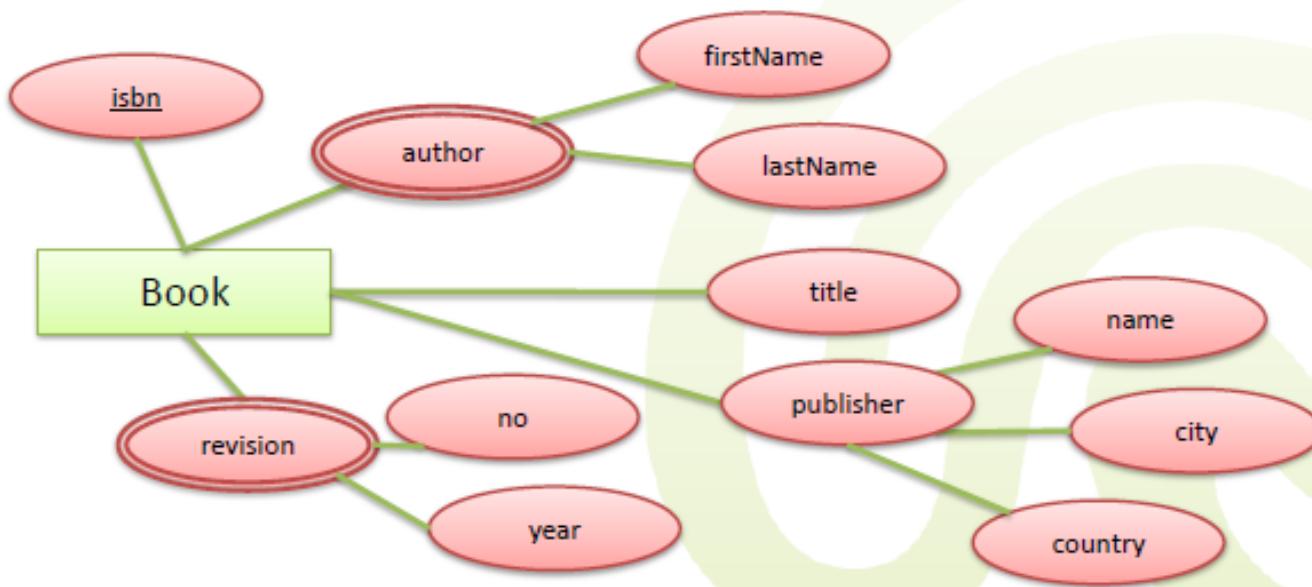
- Car (licensePlate(districtId, letterId, numericId), brand, year)
 - ((BS,CL,797),VW,1998)
 - Please note that each key attribute itself is not unique!



ER Modeling

- **Example Entity Type**

- Book (isbn, {author(firstName, lastName)}, title, subtitle, publisher(name,city, country), {revision(no, year)})
- (0321204484, {(Ramez, Elmasri), (Shamkant, Navathe)}, Fundamentals of Database Systems, (Pearson, Boston, US), {(4,2004),(2, 1994)})



ER - Domains

- Attributes cannot have arbitrary values: they are restricted by the attribute **value sets (domains)**
 - Zip Codes may be restricted to integer values between 0 and 99999
 - Names may be restricted to character strings with maximum length of 120
 - Domains are not displayed in ER diagrams
 - Usually, popular **data types** are used to describe domains in data modeling
 - e.g. integer, float, string,

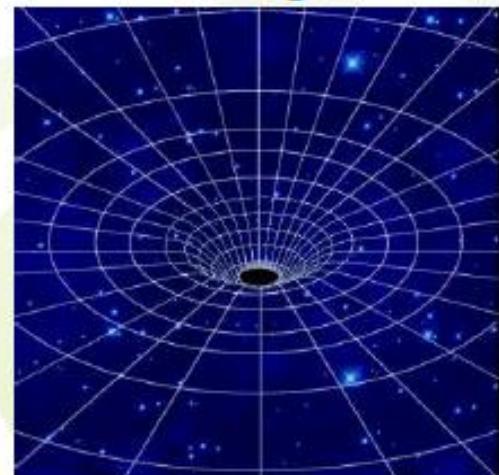
ER - Domains

- Commonly used data types

Name	Syntax	description
integer	integer	32-Bit signed integer values between -2^{31} and 2^{31}
double	double	64-Bit floating point values of approximate precision
numeric	numeric(p, s)	A number with p digit before the decimal and s digits after the decimal
character	char(x)	A textual string of the exact length x
varying character	varchar(x)	A textual string of the maximum length x
date	date	Stores year, month, and day
time	time	Stores hour, minute, and second values

ER – NULL Values

- Sometimes, an attribute value is **not known** or an attribute does **not apply** for an entity
 - This is denoted by the special value **NULL**
 - So called **NULL-value**
 - Example: Attribute “universityDegree” of entity Heinz Müller may be **NULL**, if he does not have a degree
 - **NULL** is usually always allowed for any domain or data type unless explicitly excluded



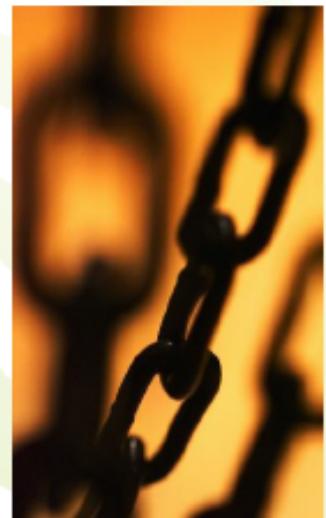
ER – NULL Values

- What does it mean when you encounter a NULL-value?
 - Attribute is not applicable
 - e.g. attribute "maiden name" when you don't have one
 - Value is not known
 - Value will be filled in later
 - Value is not important for the current entity
 - Value was just forgotten to set
- Actually there are more than 30 possible interpretations...



ER – Relationships

- Entities are not enough to model a miniworld
 - The power to model dependencies and relationships is needed
- In ER, there can be **relationships** between entities
 - Each relationship **instance** has a **degree**
 - i.e. the number of entities it relates to
 - A relationship instance may have attributes



ER – Relationships

- Similar to entities, ERDs do not model individual relationships, but **relationship types**
- **Relationship type**
 - Named set of all similar relationships with the same attributes and relating to the same entity types

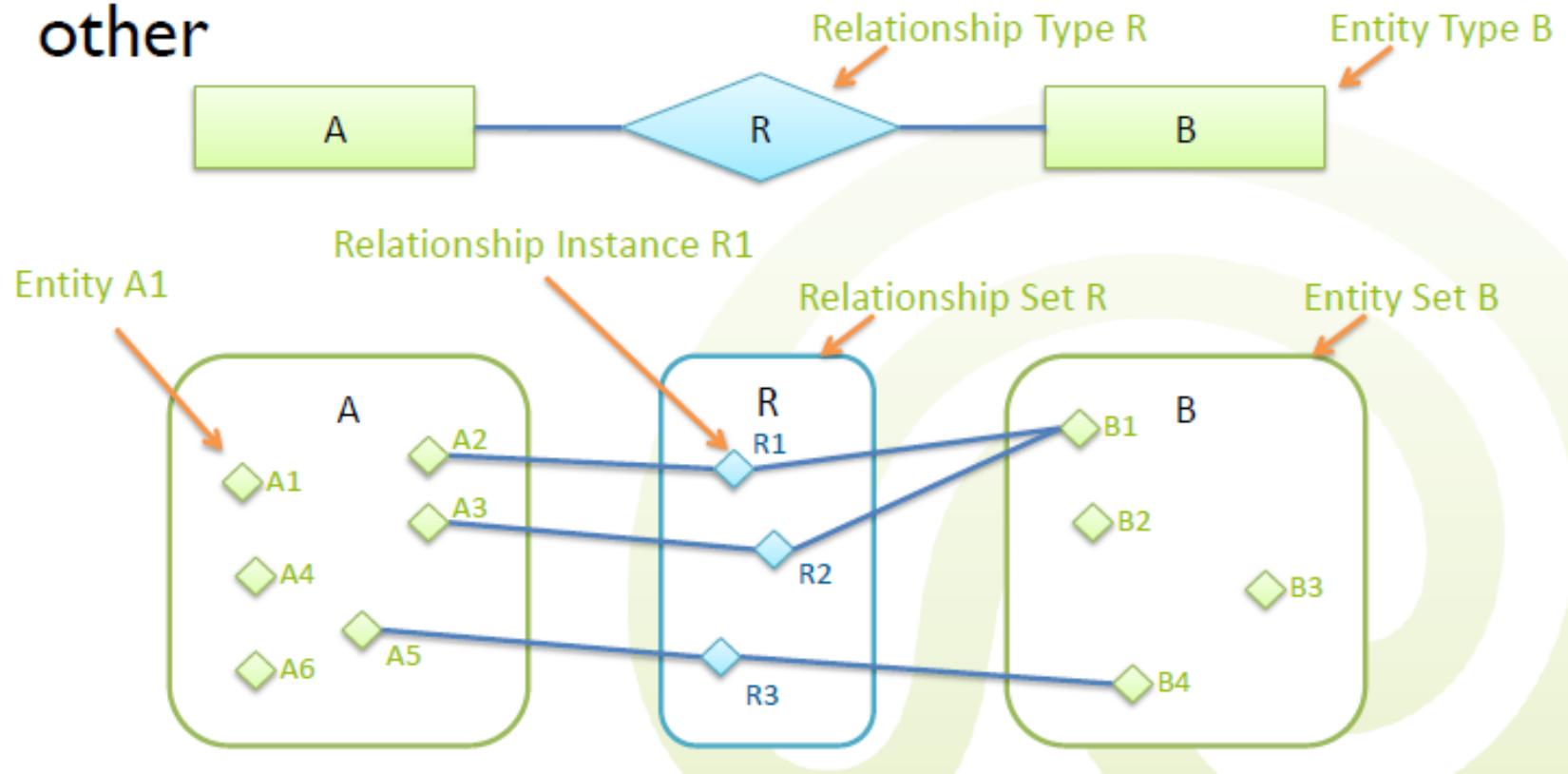


- Diamond labeled with the name of the relationship type
 - Usually, name starts with lower-case letters

- **Relationship set**
 - Set of all **relationship instances** of a certain relationship type

ER – Relationships

- **Relationships relate entities** within the **entity sets** involved in the **relationship type** to each other



ER – Relationships

- Example:

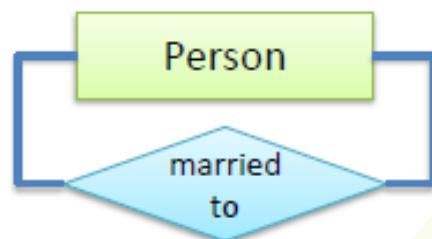
- There is an ‘ownership’ relation between cats and persons



- But more modeling detail is needed
 - Does every person own a cat? Does every cat have an owner?
 - Can a cat have multiple owners or a person own multiple cats?
 - Since when does a person own some cat?
 - Who owns whom?

ER – Relationship Cardinality

- Additionally, **restrictions** on the combinations of entities participating in an entity set are needed
 - Example: Relationship type “married to”



- Unless living in Utah, a restriction should be modeled that each person can only be married to a single person at a time
 - i.e. each person entity may only appear once in the “married to” relationship set
- **Cardinality** annotations are used for this
- Relationship types referring to just one entity type are called recursive

ER – Relationship Cardinality

- **Cardinality** annotations

- One cardinality annotation per entity type / relationship end
 - Minimum and maximum constraints possible
- Common Cardinality Expressions:
 - **(0, 1)** : Each entity may participate at most once in the relationship (i.e. relationship participation is optional)
 - **(1, 1)** : Each entity is bound exactly once
 - **(0, *)** : Each entity may participate arbitrary often in the relationship
 - **(1, *)** : Each entity may participate arbitrary often in the relationship, but at least once
 - No annotation is usually interpreted as **(0, *)**
 - If only one symbol / number s is used, this is interpreted as **(0, s)**
 - $*$ = **(0, *)**; 4 = **(0, 4)**
 - Sometimes, **N** or **M** are used instead of *****



ER – Relationship Cardinality

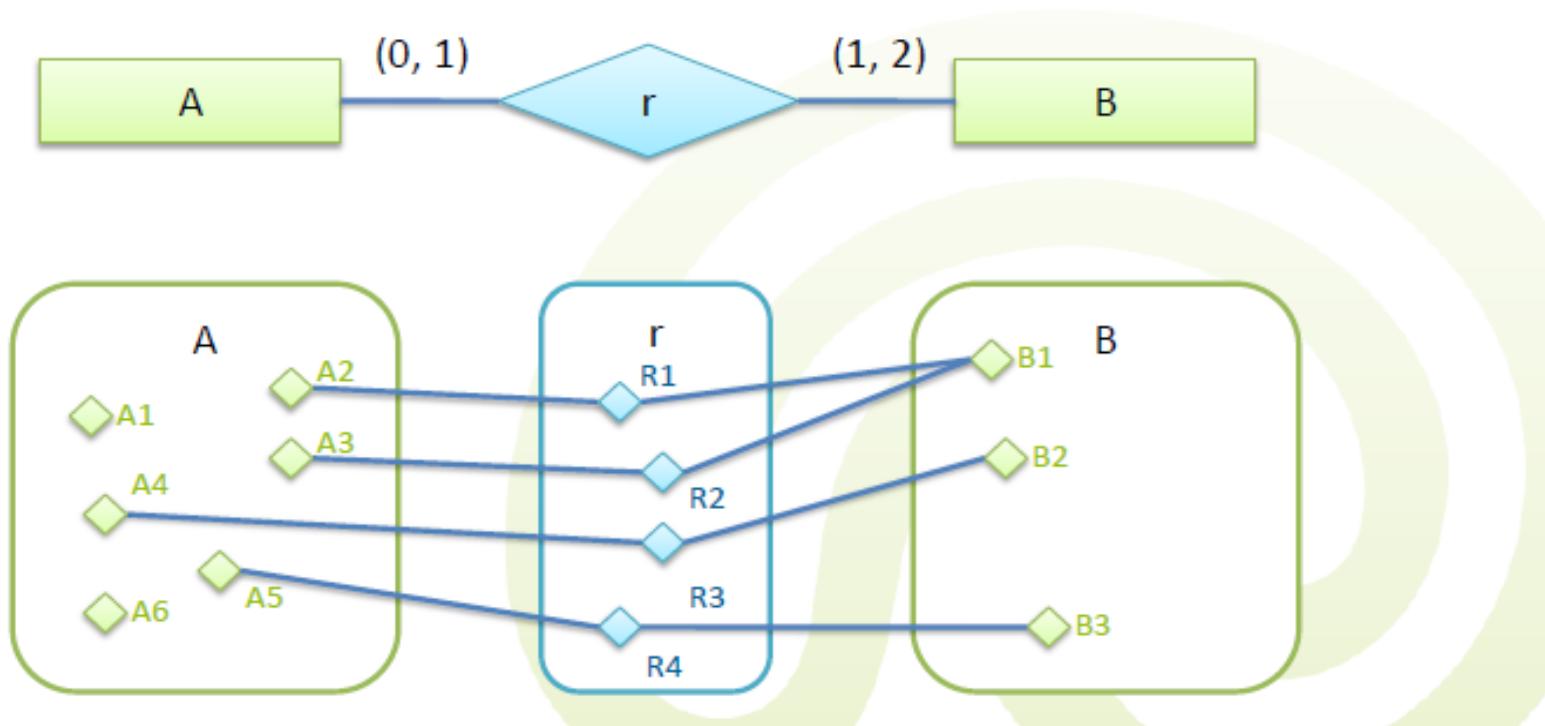
- Cardinalities express how often a specific entity may appear within a relationship set



- A specific entity of type A may appear up to once in the relationship set, an entity of type B appears at least once and at most twice
 - This means: Up to two entities of type A may relate to one entity of type B. Some entities in A are not related to any in B. All entities in B are related to at least one in A.

ER – Relationships

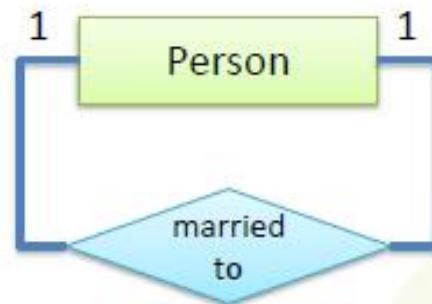
- “To each entity of type B, one or two entities of type A are related”



ER – Relationship Cardinality

- Example:

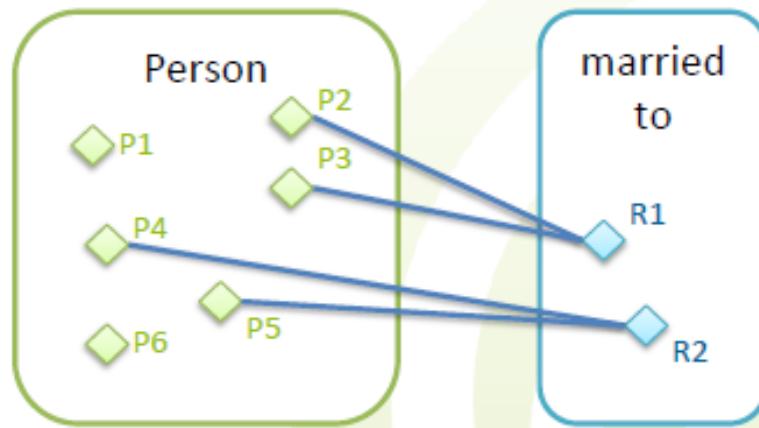
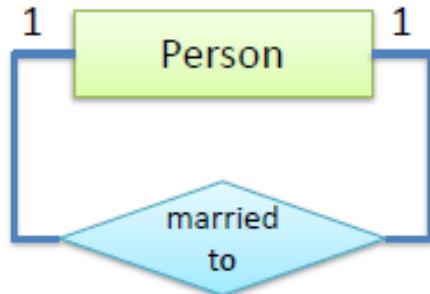
- “Each person can only be married to one other person”



- Each entity can only appear in one instance of the “married to” entity set



ER – Relationships



ER – Relationship Cardinality

- Example:
 - “A cat has up to 4 owners, but at least one. A person may own any number of cats.”

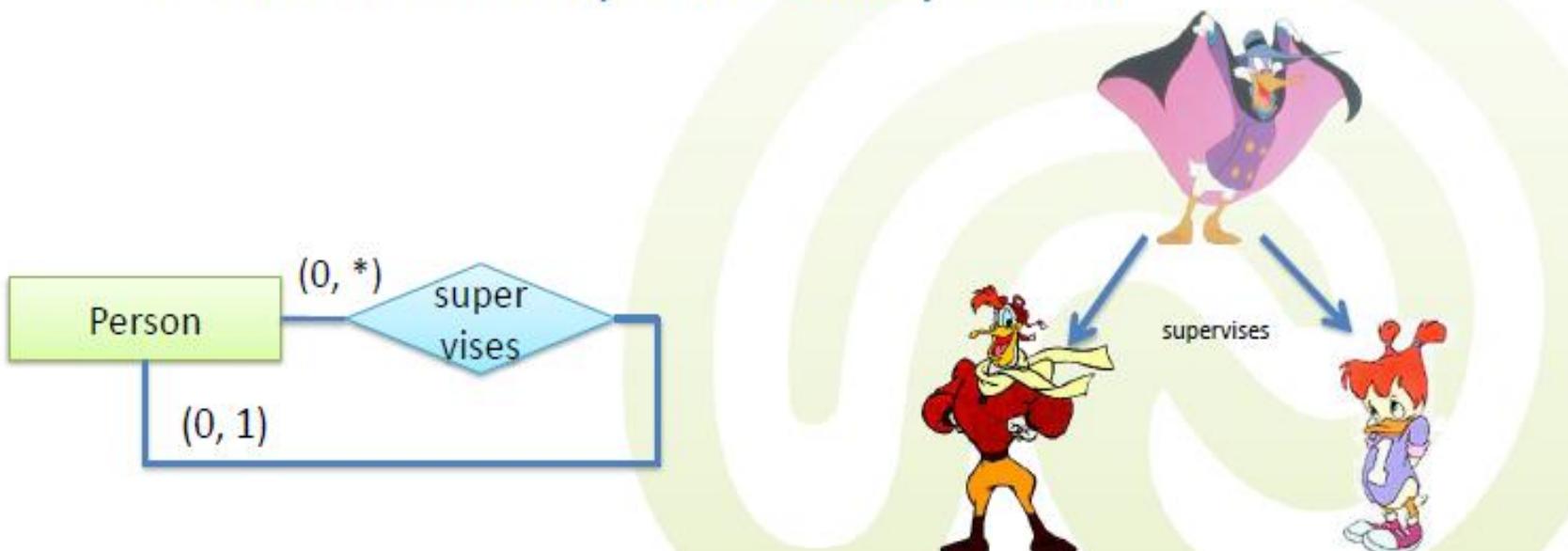


- “Lisa owns Snowball”, “Lisa owns Snowball II”



ER – Relationship Cardinality

- Example:
 - “A person may supervise any other number of persons”
 - “Drake Mallard supervises Launchpad McQuack”
 - “Drake Mallard supervises Gosaly Mallard”

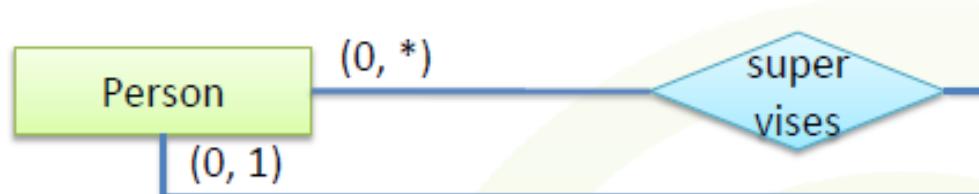


ER – Relationship Cardinality

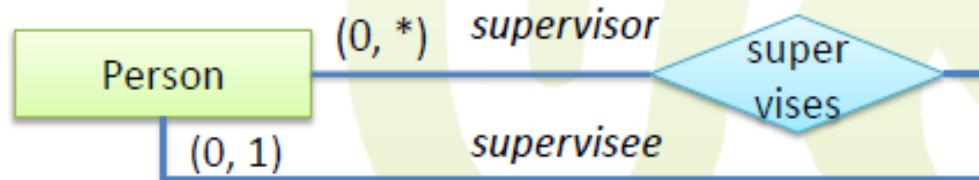
- Cardinalities for **binary** relationship types can be classified into common, more general **cardinality types**
 - These cardinality types are also often found in other modeling paradigms
 - **I:I (One-To-One)** – Each entity of the first type can only relate to exactly one entity of the other type
 - **I:N (One-To-Many)** – Each entity of the first type can relate to multiple entities of the other type
 - **N:I (Many-To-One)** – Multiple entities of the first type can relate to exactly one entity of the second type
 - **N:M (Many-To-Many)** – No restrictions. Any number of entities of first type may relate to any number of entities of second type.

ER – Relationship Roles

- Often, it is beneficial to clarify the **role** of an entity within a relationship
 - Example: Relationship “supervises”



- What is meant? Who is the supervisor? Who is the supervised person?
- Roles can be annotated on the relationship lines



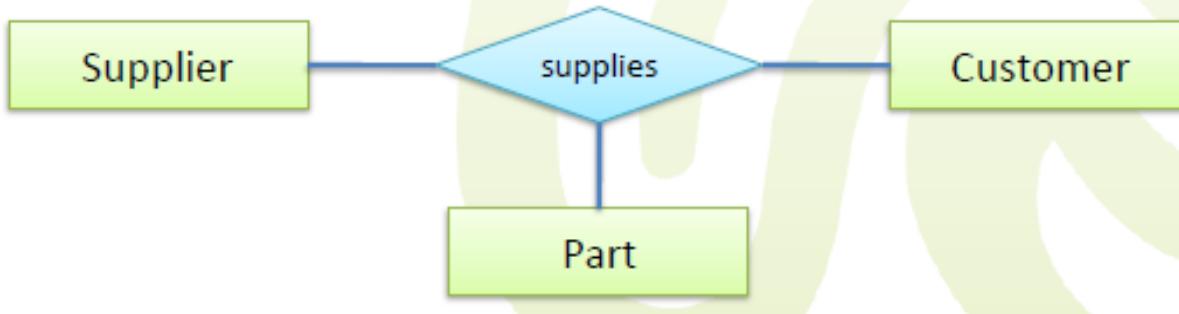
ER – Relationship Degree

- Relationship instances involve multiple entities
 - The number of entities in each relationship instance is called **relationship degree**

- Degree=2 : Binary Relation

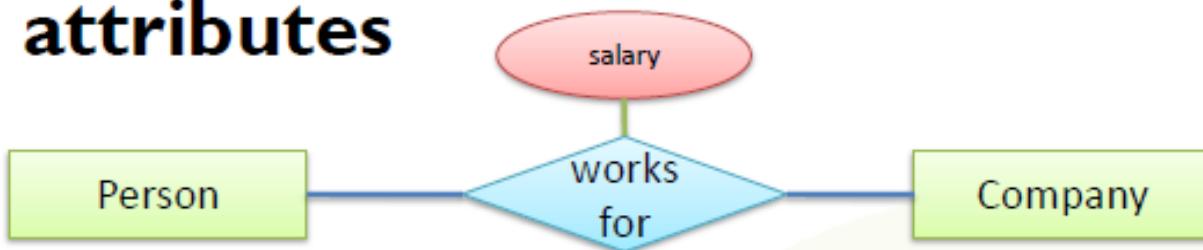


- Degree=3: Ternary Relation



ER – Relationship Attributes

- Similar to entities, relationship types may even have **attributes**



- For 1:1 relationships, the relationship attribute may be migrated to any of the participating attributes
- For 1:N relationships, the attribute may be only migrated to the entity type on the N-side
- For N:M relationships, no migration is possible

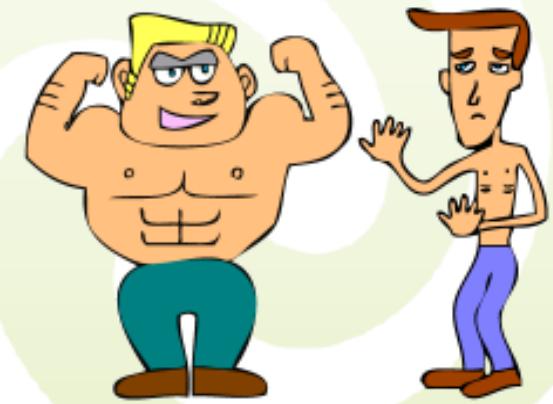
ER – Total Participation

- To express that all entities of an entity type appear in a certain relationship set, the concept of **total participation** can be used
 - The entity type which is totally participating is indicated by a double line
 - Example: “Each driver’s license must belong to a person”



ER – Weak Entities

- Each entity needs to be identifiable by a set of **key attributes**
- Entities existing **independently** of the context are called **strong entities**
 - A person exists whether it is married or not
- In contrast, there may be entities without an unique key called **weak entities**

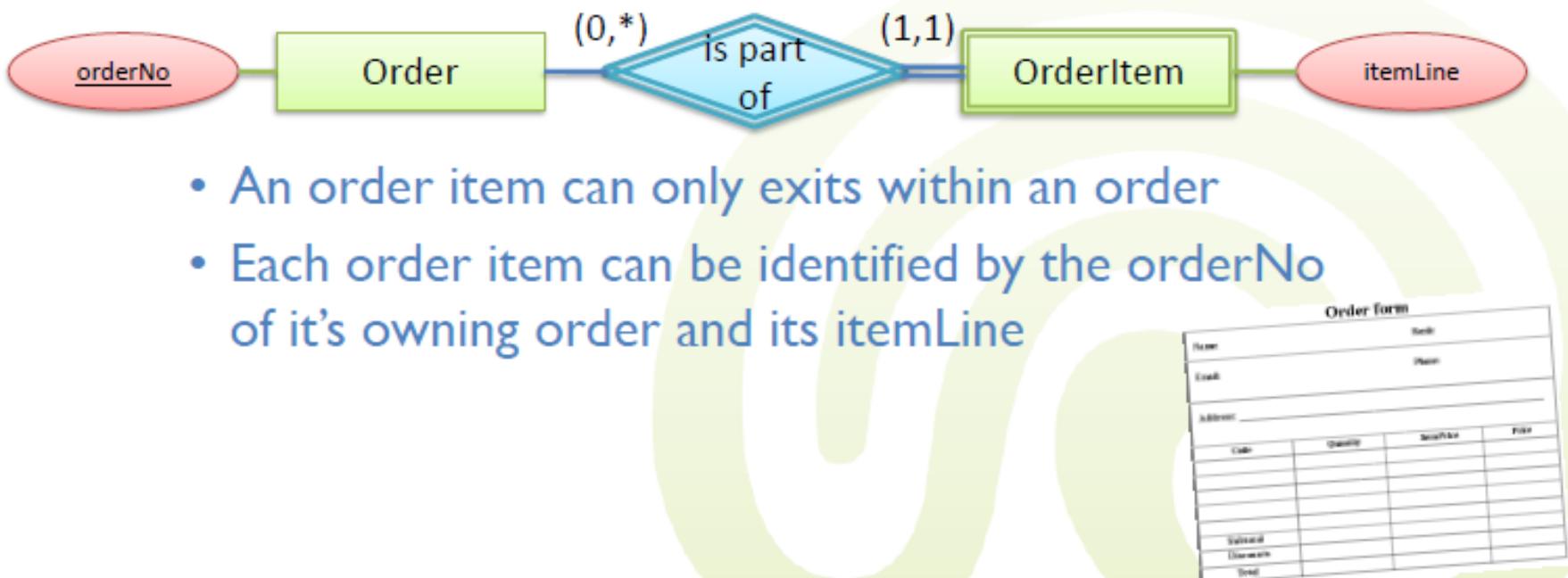


ER – Weak Entities

- **Weak entities** are identified by being related to strong entities
 - The strong entities “**own**” the weak entity
 - The weak one cannot exist without the strong ones
 - The relationships relating the strong to the weak are called **identifying relationships**
 - The weak entity is always totally participating in that relationship
 - Weak entities have **partial keys** which are unique within the identifying relationship sets of their strong entities
 - To be unique, the weak entity instance has to borrow the keys of the respective strong entity instances

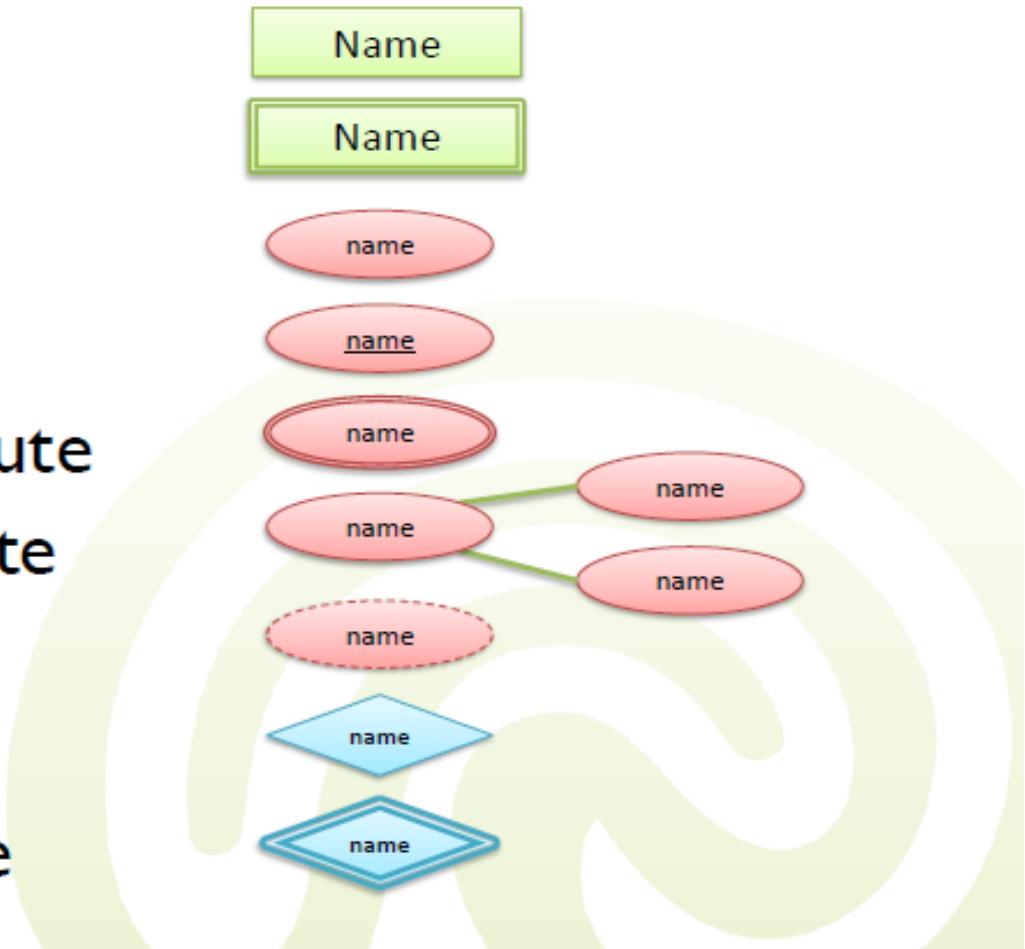
ER – Weak Entities

- Weak entity types and identifying relationship types are depicted by double-lined rectangles
 - Example:
 - “An online shopping order contains several order items”



ER – Overview

- Entity Type
- Weak Entity Type
- Attribute
- Key Attribute
- Multi-valued Attribute
- Composite Attribute
- Derived Attribute
- Relationship Type
- Identifying Rel. Type



ER – Overview

- Total participation of E2 in R



- Cardinality

- An instance of E1 may relate to multiple instances of E2



- Specific cardinality with min and max

- An instance of E1 may relate to multiple instances of E2



ER – Mathematical Model

- Mathematically, an **attribute** A of **entity type** E with **domain** V is a function from E to the power set $P(V)$
 - $A : E \rightarrow P(V)$
 - The power set $P(V)$ of V is the set of all subsets of V
 - The **value** of an attribute of the entity e is denoted as $A(e)$
 - This definition covers
 - null values (empty set)
 - single-valued attributes (restricted to singleton sets)
 - multi-valued attributes (no restrictions)
 - For a composite attribute $A(A_1, A_2, \dots, A_n), V$ is defined as
 - $V = P(V_1) \times P(V_2) \times \dots \times P(V_n)$

ER – Mathematical Model

- A **relationship type** R among n **entity types** E_1, E_2, \dots, E_n defines a **relationship set** among instances of these entity types
 - Each relationship instance r_i within the relationship set R associates n individual entities (e_1, e_2, \dots, e_n) , and each entity e_j in r_i is member of the entity type E_j , $1 \leq j \leq n$
 - Alternatively, the relationship type R can be seen as a subset of the Cartesian product of the entity types
 - $R \subseteq E_1 \times E_2 \times \dots \times E_n$

An example

- We want to model a simple university database
 - In our database, we have students. They have a name, a registration number, and a course of study.
 - The university offers lectures. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a title, provide a specific number of credits and have an unique ID
 - Each year, some of these lectures are offered by a professor at a certain day at a fixed time in a specific room. Students may register for that lecture.
 - Professors have a name and are member of a specific department.



An example

- How to start? What to do?
 - Find the basic **entity types**
 - Find the **attributes** of entities
 - Decide to which entity an attribute should be assigned
 - Which attributes are key attributes?
 - Some attributes are better modeled as own entities, which ones?
 - Define the **relationship types**
 - Which role do entities play?
 - Do relationships require additional entity types?
 - Are the relationships total? Identifying? Are weak entities involved?
 - What are the cardinalities of the relationship type?

An example

- Which are our entity types?
 - In our database, we have **students**. They have a name, a registration number and a course of study.
 - The university offers **lectures**. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a title, provide a specific number of credits and have unique ID
 - Each year, some of these lectures are offered by a **professor** at a certain day at a fixed time in a specific room. Students may attend that lecture.
 - Professors have a name and are member of a specific department.

An example

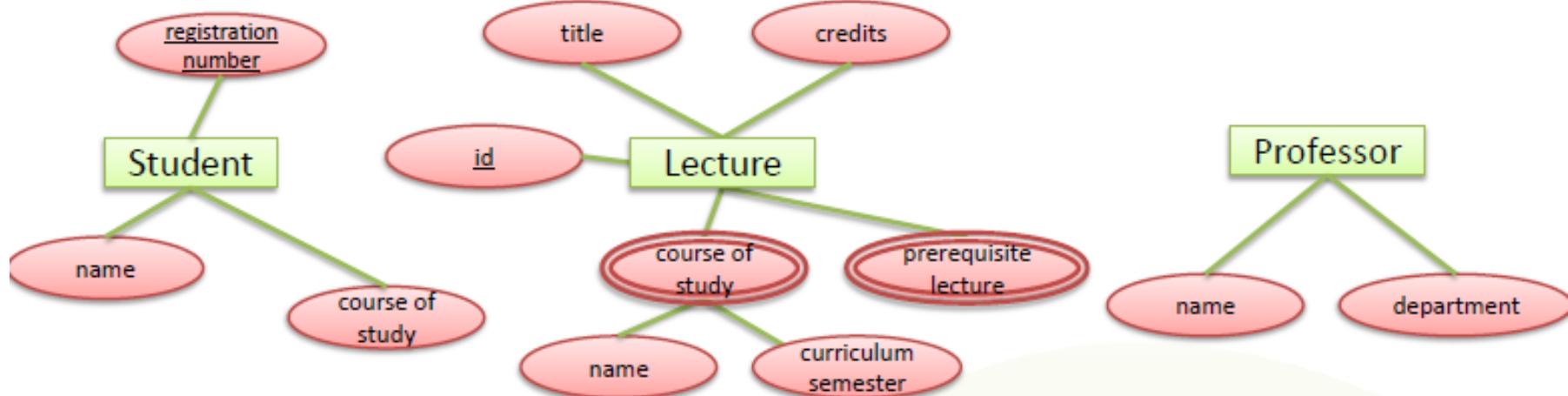
Student

Lecture

Professor

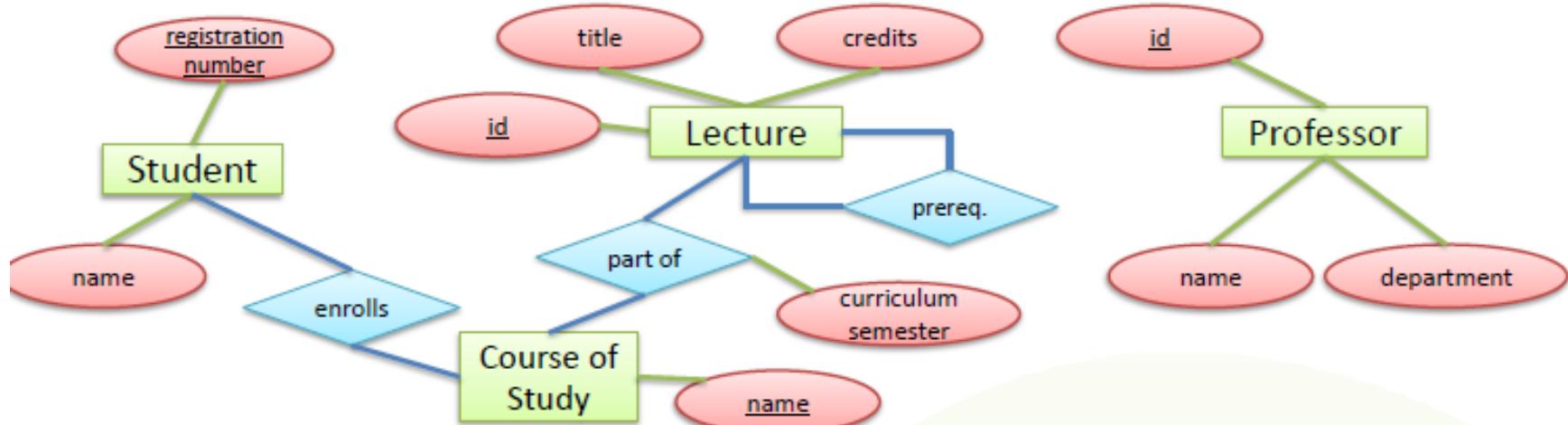
- What attributes are there?
 - In our database, we have students. They have a name, a registration number and a course of study.
 - The university offers lectures. Each lecture may be part of some course of study in a certain semester. Lectures may have other lectures as prerequisites. They have a title, provide a specific number of credits and have unique ID
 - Professors have a name and are member of a specific department.

An example



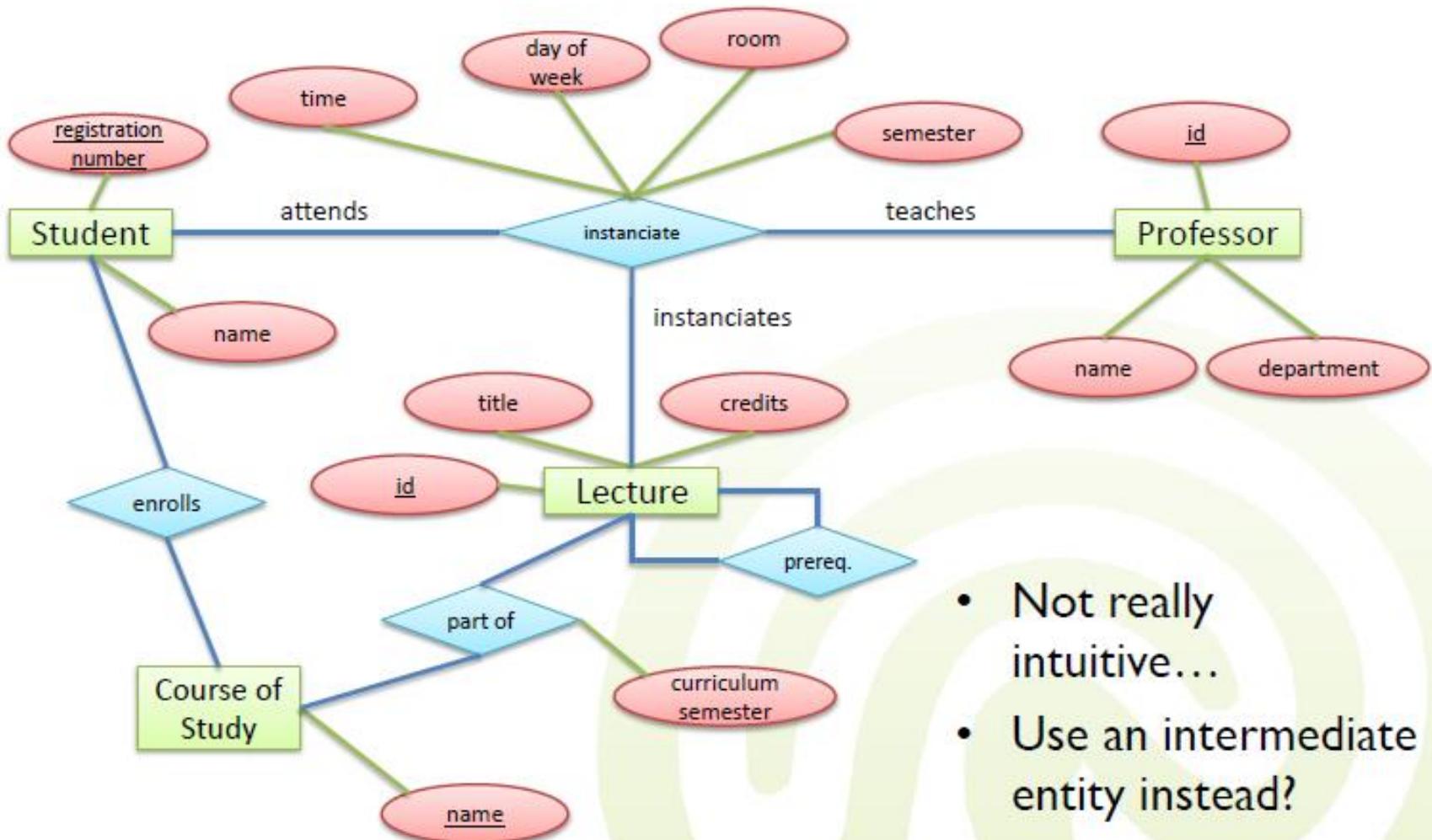
- First try...
 - This model is **really crappy!**
 - “Course of study” does not seem to be an attribute
 - Used by student and lecture. Even worse, lecture refers to a course of study in a specific curriculum semester.
 - Use additional entity type with relationships!
 - “Prerequisite lecture” also is not a good attribute
 - Prerequisite lectures are also lectures. Use a relationship instead!
 - “Professor” does not have key attributes

An example



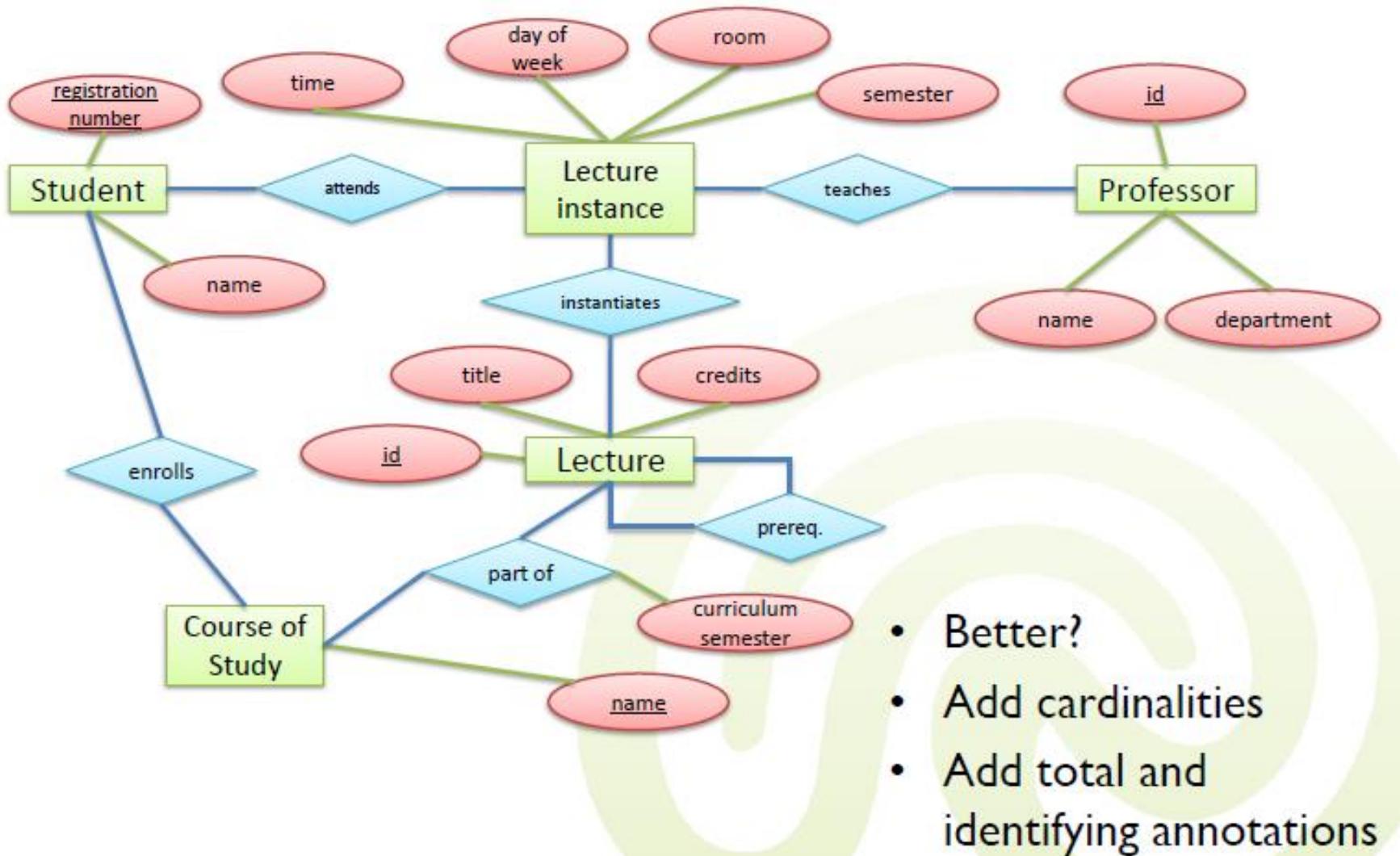
- Second try
 - Professors use a **surrogate key** now
 - Key is automatically generated and has no meaning beside unique identification
 - Course of study is an entity type now
- Which entity types are additionally related?
 - “Each year, some lectures of the pool of all lectures are offered by a professor at a certain day at a fixed time in a specific room. Students may attend that lecture.”

An example

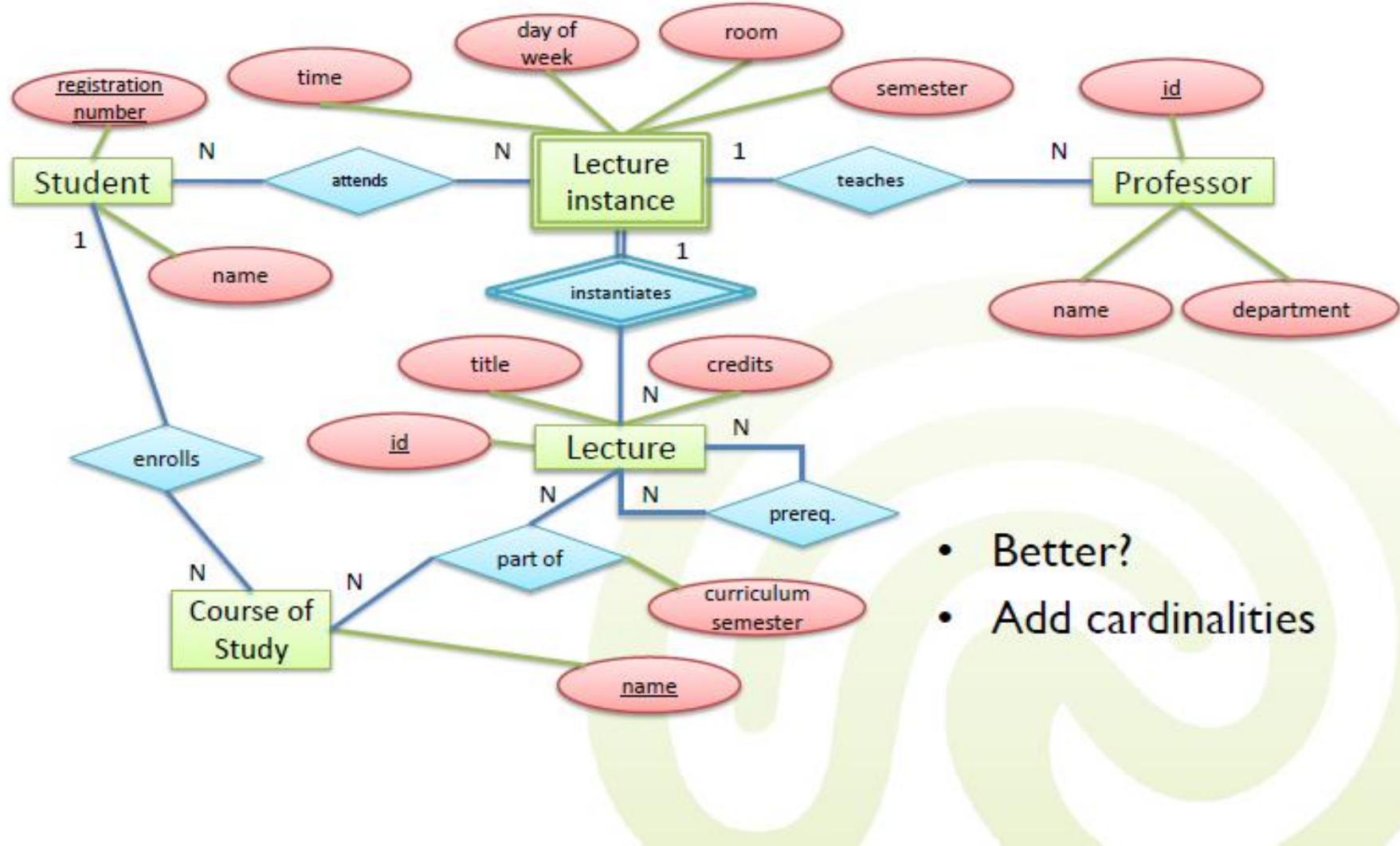


- Not really intuitive...
- Use an intermediate entity instead?

An example



An example



An example

- Modeling is not that simple
- Many possible (and also correct) ways of modeling the same miniworld
 - Some are more elegant, some are less elegant
- Models alone are not enough, they need to be documented
 - What are the meanings of the attributes? The meanings of the relationships?

Summary

- Data models
 - Definition, 3 parts, different categories
- Three-layer architecture (ANSI/SPARC)
 - Presentation, logical, physical level
- Data independence
- DB design phases
- **ER Modeling**
 - Chen notation

