# Entity-relation models 2

Lecture 3

Wednesday - Aug 30, 2023

# Housekeeping

- Project deliverable 3 - this is approval for your project idea. You're making a brief video and submitting it for approval.
- Homework 2 due Friday. No class on Monday.
- Practice quiz available on Canvas.
- Quiz on Wednesday. Lockdown Browser required. Window of time, probably 5pm-7:30pm on Wednesday.

| Module | Week | Date | Day | Lectures/Quizzes | Deliverables/Notes |
|---|---|---|---|---|---|
| ER Models | 2 | 8/30 | Wed | MTG3: L3 (Entity-relation models 2) | |
| ER Models | 2 | 9/1 | Fri | | HW2 due (ER Models 1) |
| ER Models | 3 | 9/4 | Mon | | University closed: Labor day |
| ER Models | 3 | 9/6 | Wed | MTG4: Quiz 1 today (Entity-relation models) | |
| Relational Alg. | 4 | 9/11 | Mon | MTG5: L4 (DDL / DML / SQLLite / MySQL) | |
| Relational Alg. | 4 | 9/12 | Tue | | HW3 due (ER Models 2 - extended) |
| Relational Alg. | 4 | 9/13 | Wed | MTG6: L5 (Relational models) | |
| Relational Alg. | 4 | 9/15 | Fri | | PrjDel 3 due (Topic proposal video) |

# A quick review of last time …

# Key components of an ER model

**Entities**

These represent real-world objects, concepts, or things that have data stored about them. Each entity is depicted as a rectangle in the ER diagram.

**Attributes**

Attributes are properties or characteristics that describe the entities. For example, if you're modeling a *Customer* entity, attributes could include *CustomerID, Name, Email,* etc.

**Relationships**

Relationships illustrate how entities are related to each other. Relationships can be one-to-one, one-to-many, or many-to-many, and they help define how data is interconnected.

# Cardinality

Cardinality describes the number of instances of one entity that can be associated with the number of instances of another entity through a relationship. It defines the multiplicity of the relationship.

There are three main types of cardinality:

**One-to-One (1:1)**

In a one-to-one relationship, each instance of one entity is related to only one instance of another entity, and vice versa.

**One-to-Many (1:N)**

In a one-to-many relationship, each instance of one entity can be related to multiple instances of another entity, but each instance of the other entity is related to only one instance of the first entity.

**Many-to-Many (N:M)**

In a many-to-many relationship, each instance of one entity can be related to multiple instances of another entity, and vice versa.

# Participation

Participation refers to whether every instance of an entity must participate in a relationship or if participation is optional. It indicates whether an instance of an entity is required to have a corresponding related instance in the related entity.

There are two main types of participation:

**Total Participation (Existence Dependency)**
: In total participation, every instance of the participating entity must be related to at least one instance of the related entity.

**Partial Participation**
: In partial participation, an instance of the participating entity may or may not be related to an instance of the related entity.

# and now back to our regularly scheduled program

# Let’s build an ER diagram!

Imagine a library management system that keeps track of books, authors, and publishers.

Each book in the system is identified by its International Standard Book Number (ISBN), which serves as its unique identifier. Books are associated with authors and publishers.

Authors are individuals who write books. Each author is identified by a unique AuthorID and has a name associated with them. Authors have written at least one book but can write as many as they are able.

Publishers are entities responsible for releasing books to the public. Each publisher has a unique PublisherID, a name, and a home city. Publishers have at least one book in their catalog and can publish as many as they are able.

**Prepare ER diagrams for this system using both Chen and Crow’s foot notation.**

# Let’s build an ER diagram! (starter kit)

## Using graphviz for Chen diagram

```
graph ER {
    layout=neato;
  scale=1.1;
    node [shape=box]; building; room;
    node [shape=ellipse]; address, city, state, occupar
    node [shape=diamond,style=filled,color=lightgrey];

    building -- has [label="1",len=1.00]
    has -- room [label="n",len=1.00]

    building -- address
    building -- city
    building -- state
  room -- room_number
    room -- occupant
}
```

Link to graphviz on-line editor

## Using mermaid for Crows foot notation

```
erDiagram
    ORDER }o--|| CUSTOMER : places
    ORDER }o--|| SALES_PERSON : places
    ORDER }o--|| PRODUCT : places
```

Link to mermaid live

# Closing the loop - dropping them into quarto!

(quick demo using HW2 as example)

# ER Diagram Practice

Try creating ER diagrams of both sorts with these problem descriptions.

## P1. Mural Tracking System

Imagine a city that wants to keep track of its murals, artists, and the buildings they are painted on. Each mural has a mural ID, title, and completion date. Artists have an artist ID, name, and artistic style. Buildings have a building ID, address, and owner name. An artist can create multiple murals, but each mural is created by a single artist. Murals can be painted on multiple buildings, and buildings can have multiple murals. Each mural is associated with a single building and is created by one artist.

# ER Diagram Practice, Practice

Try creating ER diagrams of both sorts with these problem descriptions.

## P2. Music Venue System

Consider a music venue system that wants to manage bands, venues, and concerts. Each band has a band ID, name, and genre. Venues have a venue ID, name, and capacity. Concerts have a concert ID, date, and ticket price. A band can play in multiple concerts, and concerts can host multiple bands. Concerts take place in venues, and each venue can host multiple concerts but each concert is held at a single venue. Bands perform at concerts, and each concert can host multiple bands.

# ER Diagram Practice, Practice, Perfect!

Try creating ER diagrams of both sorts with these problem descriptions.

## P3. Hospital System

Picture a hospital that wants to keep track of patients, doctors, and the treatments given. Each patient has a patient ID, name, and medical history. Doctors have a license number, name, and area of specialization. Treatments have a treatment code, name, and cost. Patients can receive multiple treatments, and each treatment can be administered to multiple patients. Doctors are responsible for administering treatments to patients, and each doctor can administer multiple treatments but each treatment is administered by a single doctor.

# Tricky situations

# Tricky situations

These represent design choices in how you choose to represent features of the real world in your conceptual design.

You should apply an 80/20 or 90/10 rule. Don’t worry about trying to capture every feature, just the big ones.

The rubber meets the road when you create the physical design and write the actual SQL code.

# Tricky situations - Entities

**Weak vs. Strong Entities**

> Strong entities exist independently. Weak entities depend on a strong entity for their existence. If the strong entity is deleted, then weak entity is deleted, too.

For instance, a “Employee” could be a strong entity, while “Dependents” could be a weak entity depending on it.

They’re both *entities*, however, your design decision is what to do with the weak entity when the strong entity gets deleted. Do you delete it or keep it?

*ask chatgpt: why do strong vs. weak entities matter in er diagrams?*

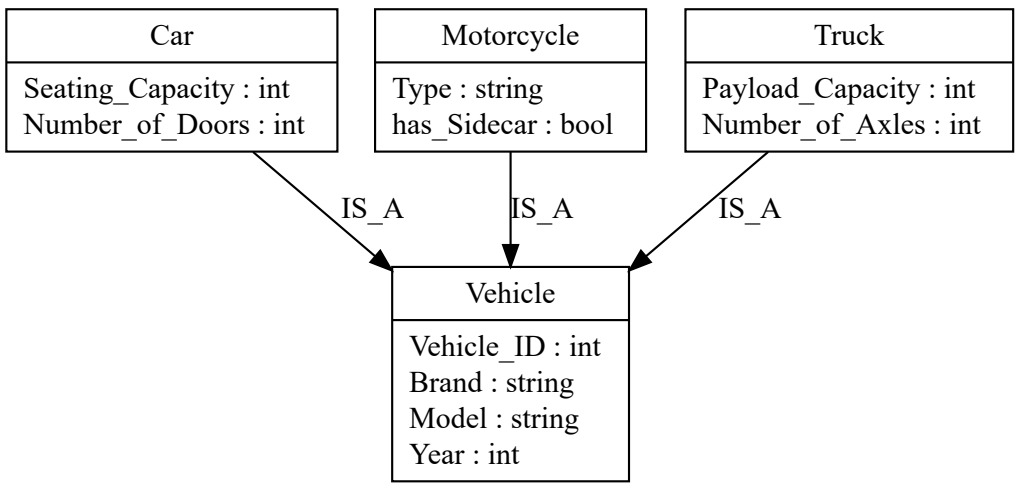
**Subtypes and Supertypes**

> Entities can be generalized or specialized. This can help improve data integrity and minimize the need to repeat fields.

For example, “Vehicle” could be a supertype of “Car” and “Motorcycle”. As another example, *Employee* can be the supertype and subtypes might include *engineer*, *salesperson*, *manager*.

*ask chatgpt: in er diagrams, why to supertype and subtypes matter?*

# Tricky situations - Entities




**In this example**

- Car, Motorcycle and Truck are subtype entities.
- They all share a common supertype (parent), *vehicle* that contains attributes common to all.
- Car, motorcycle and truck are strong entities. Vehicle is a weak entity. If a Car is deleted, the corresponding vehicle information must also be deleted!
- This is a good strategy to minimize duplication of common attributes across entities.
- However, you must ensure that care is taken when deleting to ensure that you don't get vehicles without subtypes!
- You can't directly query vehicle without a connection back to the subtype entity! Modeling the *is_a* relationship is key.

# Tricky situations - Attributes

**Simple vs. Composite**

Simple attributes cannot be divided any further, whereas composite attributes can.

For example “Full Name”, can be divided into “First Name” and “Last Name”.

**Single-valued vs. Multi-valued**

For example, *Address* or *Phone Number* can be multi-valued if multiple numbers are allowed for a single individual.

**Derived Attributes**

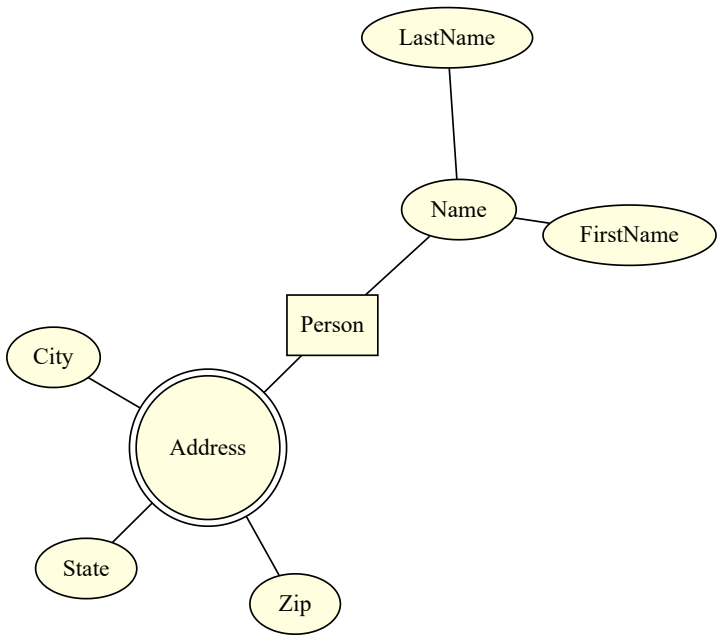
These are attributes that can be derived from other attributes in the database.

For instance, “Age” can be derived from “Date of Birth”.

**Key Attributes**

Primary keys, foreign keys serve as attributes that help uniquely entities.

# Tricky situations - Attributes




**In this example**

- *person* is the entity with two attributes.
- *name* is a composite attribute, with subattributes first name and last name.
- *address* is a multi-valued attribute, that is, each person can have multiple addresses. Each address is a composite attribute with city, state and zip.

# Tricky situations - Relationships

**Recursive Relationships**

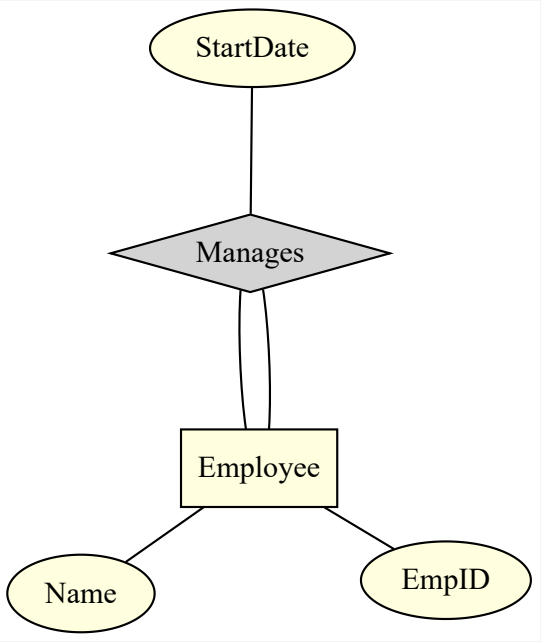
Sometimes an entity has a relationship with itself.

**Associative Entities**

When a many-to-many relationship exists, it often helps to introduce an associative entity that holds extra data about the relationship.

**In this example**

An *employee* manages other employees. This is a *recursive* relationship. We're also keeping track of the start date of the management relationshpi using an *associative entity*.



# Tricky situations - Relationships

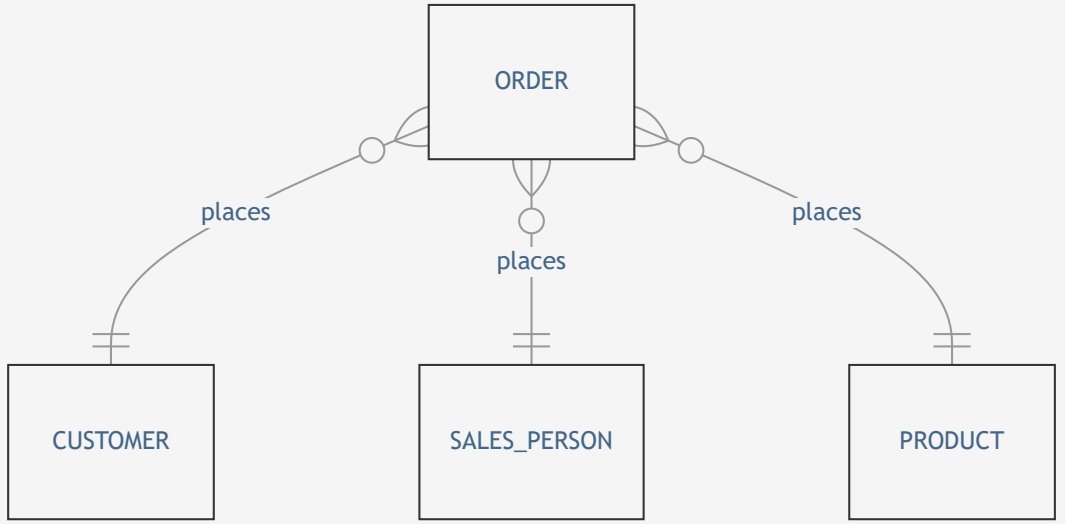
**N-ary Relationships**

More complex relationships that involve more than two entities.

A common relationship is a 3-way, or ternary relationship.

**In this example**

An order requires three entities to exist: a customer, a sales_person, and a product.



# … ok, now what? Let’s take a step back …

# Database design process - overview

## Overview

- Basic steps are on the right.
- There are many different approaches.
- Watch the videos, you'll see different approaches.
- Can be as hard or simple as you want to make it.
- Iterate!
- *Note: the Microsoft approach (on Canvas) jumps more quickly to logical design and implementation.*

## Database design process

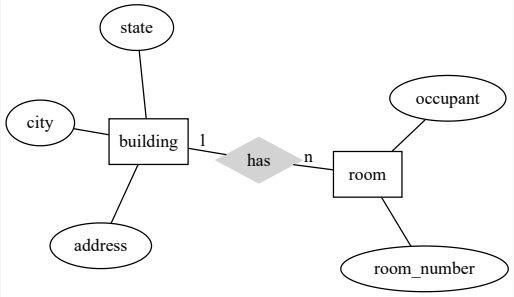
1. Understanding requirements
2. **Conceptual design** <- Chen diagrams/graphviz
3. **Logical design** <- Crows foot diagrams/mermaid
4. Physical design
5. Implementation

# Database design process - logical design

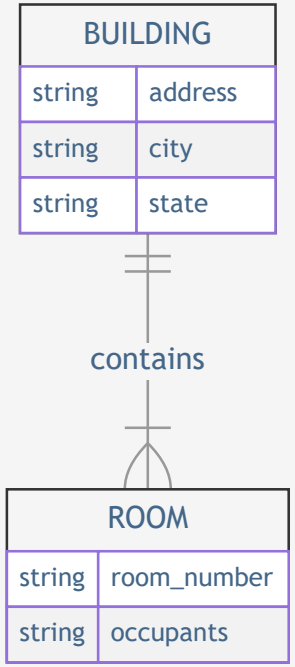
- Entity sets and relationship sets can be expressed uniformly as relation schemas that represent the contents of the database
- A database that implements an E-R diagram can be represented by a collection of relation schemas
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding set.
- Each schema has a number of columns corresponding to attributes which have unique names.

# Database design process - logical design

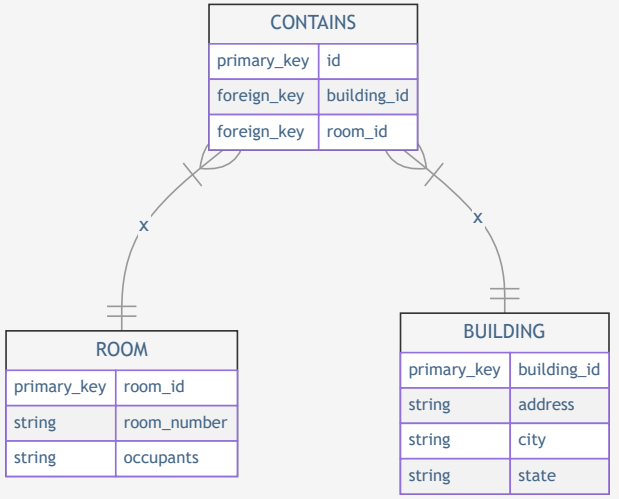
## Conceptual design




## Logical design




## Logical design refined




### *Relation set* notation:

- building( <u>building_id</u>, address, city, state)
- room( <u>room_id</u>, room_number, occupants)
- contains( <u>id</u>, building_id, room_id)

# Sample code from the earlier ER diagrams.

# Sample code

```
digraph G {
    graph [rankdir=TB];
    node [shape=record];

    // Define the supertype
    Vehicle [label="{Vehicle | Vehicle_ID : int \l Brand : string \l Model : string \l Year :

    // Define the subtypes
    Car [label="{Car | Seating_Capacity : int \l Number_of_Doors : int \l }"];
    Motorcycle [label="{Motorcycle | Type : string \l has_Sidecar : bool \l }"];
    Truck [label="{Truck | Payload_Capacity : int \l Number_of_Axles : int \l }"];

    // Indicate subtype-supertype relationships
    Car -> Vehicle [label="IS_A"];
    Motorcycle -> Vehicle [label="IS_A"];
    Truck -> Vehicle [label="IS_A"];
}
```

# Sample code

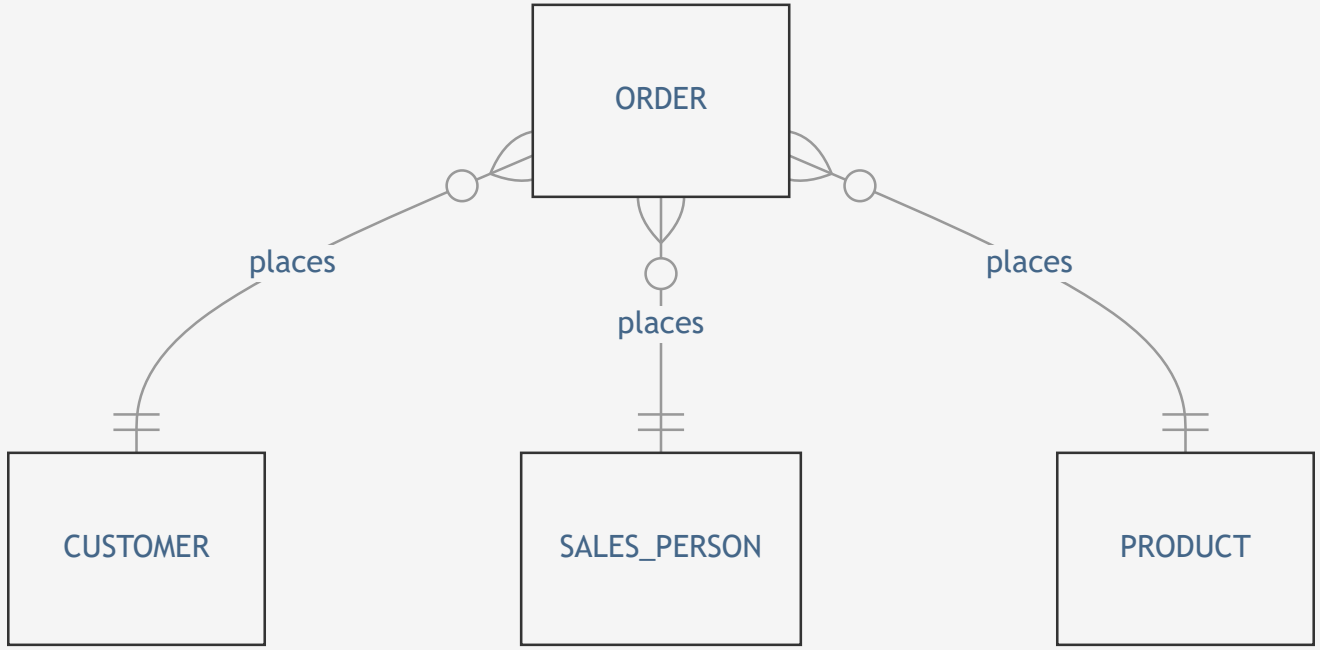
```
graph ERDiagram {
  layout=neato;
  scale=1.3;
  // Basic styling
  node [style=filled, fillcolor=lightyellow];

  // Entity
  Person [shape=box, label="Person"];

  // Composite attribute for Name
  Name [shape=ellipse, label="Name"];
  FirstName [shape=ellipse, label="FirstName"];
  LastName [shape=ellipse, label="LastName"];

  // Multivalued composite attribute for Address
  Address [shape=doublecircle, label="Address"];
  City [shape=ellipse, label="City"];
  State [shape=ellipse, label="State"];
  Zip [shape=ellipse, label="Zip"];

  // Connect entity to attributes
  Person -- Name;
```

# Sample code

```
graph ERDiagram {
  layout="neato";
  scale=1.3
  // Basic styling
  node [style=filled, fillcolor=lightyellow];

  // Entity
  Employee [shape=box, label="Employee"];

  // Attributes for Entity
  EmpID [shape=ellipse, label="EmpID"];
  Name [shape=ellipse, label="Name"];

  // Relationship
  Manages [shape=diamond, label="Manages", fillcolor=lightgray];

  // Attribute for Relationship
  StartDate [shape=ellipse, label="StartDate"];

  // Connect entity to attributes
  Employee -- EmpID;
```

# Sample code

```
erDiagram
    ORDER }o--|| CUSTOMER : places
    ORDER }o--|| SALES_PERSON : places
    ORDER }o--|| PRODUCT : places
```



# Housekeeping

- Project deliverable 3 - this is approval for your project idea. You're making a brief video and submitting it for approval.
- Homework 2 due Friday. No class on Monday.
- Practice quiz available on Canvas.
- Quiz on Wednesday. Lockdown Browser required. Window of time, probably 5pm-7:30pm on Wednesday.

| Module | Week | Date | Day | Lectures/Quizzes | Deliverables/Notes |
|---|---|---|---|---|---|
| ER Models | 2 | 8/30 | Wed | MTG3: L3 (Entity-relation models 2) | |
| ER Models | 2 | 9/1 | Fri | | HW2 due (ER Models 1) |
| ER Models | 3 | 9/4 | Mon | | University closed: Labor day |
| ER Models | 3 | 9/6 | Wed | MTG4: Quiz 1 today (Entity-relation models) | |
| Relational Alg. | 4 | 9/11 | Mon | MTG5: L4 (DDL / DML / SQLLite / MySQL) | |
| Relational Alg. | 4 | 9/12 | Tue | | HW3 due (ER Models 2 - extended) |
| Relational Alg. | 4 | 9/13 | Wed | MTG6: L5 (Relational models) | |
| Relational Alg. | 4 | 9/15 | Fri | | PrjDel 3 due (Topic proposal video) |