# Entity-relation models 1

Lecture 2

Monday - Aug 28, 2023



## Housekeeping

Another busy week, mostly with software set up.

- Start to work on homework 2, DUE FRIDAY!
- Review the videos:
  - Database design, ER diagramming, ER diagrams to relational tables.
- Kick around database ideas with your teammate(s) and begin to draw pictures!
- **Quiz 1** next week, topics: Database design, ER diagrams. More details on Wednesday.

| Module | Week | Date | Day | Lectures/Quizzes | Deliverables/Notes |
|---|---|---|---|---|---|
| ER Models | 2 | 8/28 | Mon | MTG2: L2 (Entity-relation models 1) | Last day of add/drop |
| 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) | |

# Homework 2 discussion



# Overview and Software Baseline

### Overview

Homeworks 2 and 3 both focus on ER diagramming and report writing.

Homework 2 is *mostly* about getting everything installed and running. The majority of the points are awarded for simply making the tools work. Why *do it for free* when you can earn points towards the final grade!

See the rubric on the main homework page for details on how the assignment will be scored.

Link to homework 2

### Software Baseline

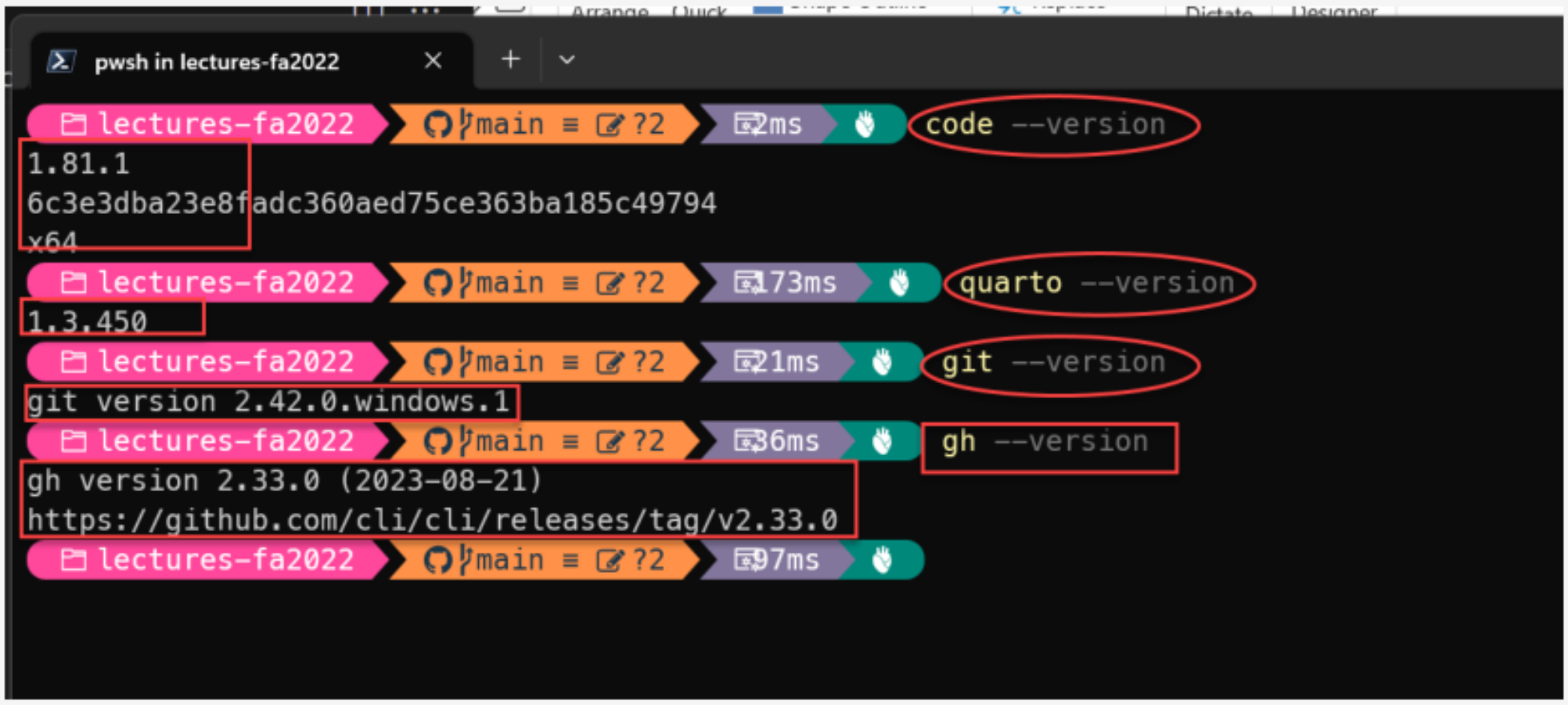
To successfully complete homework 2 - and the rest of the class - you should verify:

- visual studio code,
- quarto,
- git,
- gh (the github CLI, newly added to the list).

If you're feeling bold:

- pyenv (optional)
- python

# Overview and Software Baseline




*Notes: using the command line: all programs are running from command line*



# Quarto and rendering

This figure presents a high-level quarto workflow.


For homework 2, using quarto to render *qmd* files to markdown *md* files. We'll use GITHUB *present* the *md* files in a pretty way.

In later assignments, we'll use quarto to render further down the pipeline, from *qmd* to *html*, *pptx*, or *pdf*.

As an example, homework 1 provides a *qmd* file to be rendered to *html*. Give it a whirl!

# Demonstration of HW2

- Open command line
- Check software versions
- Download repo
- render a few items
- push back to github
- view results



# Database design process

# 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
3. Logical design
4. Physical design
5. Implementation



# Database design process - more detail

1. Understanding requirements
   - Problem domain,
   - Need,
   - Context, scope and perspective,
   - Uses cases, user roles
   - Security, privacy
   - Use cases, specific queries
2. Conceptual design
   - Entities, attributes, relationships
   - ER diagrams
3. Logical design
   - converting entities, attributes and relations, and other odd features to tables.
   - Normalization and functional analysis
4. Physical design
   - More fully defining tables
   - Adding integrity constraints, triggers
   - Reporting: adding views
   - Maintenance: adding procedures
   - Considering access: defining the API
5. Implementation
   - code it up,
   - load it up,
   - test it up,
   - see where you screwed it up,
   - iterate it up.

*This maps nicely with the Semester Project deliverables!*

# Database design process - normalization

*Normalization* is the process of reorganizing the columns and tables of a relational database to minimize data redundancy and avoid inconsistencies

*Normalization* decomposes a table into smaller tables without losing information

*Normalization* is iterative, don't *expect* perfection the first time through!

*Normalization* is iterative, don't *try* to achieve perfection the first time through. Relax!

*Normalization* Theory provides simple tests to determine a "good design"



# Database design process - ER diagrams

It's all about sharing your database design. How do you share your design with others?

An entity-relationship (ER) model was first proposed by Peter Chen in 1976

- Highlights the conceptual features (schema) of a database.
- Maps the meanings and interactions of real-world entities to a conceptual schema
- Focuses on portraying relations between entities, attributes, and relationships in a graphical fashion.
- Different notions to support different needs.
- Quarto supports at least two notations. We'll use both!
- Tons of other tools to draw ER diagrams using different notations.
- THERE IS NO STANDARD!

# Entities, attributes and relationships and the ER Diagram



# 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.

# Example

Imagine a city full of buildings.

Each building can can be located with an address, city, and state. A building consists of one or more rooms.

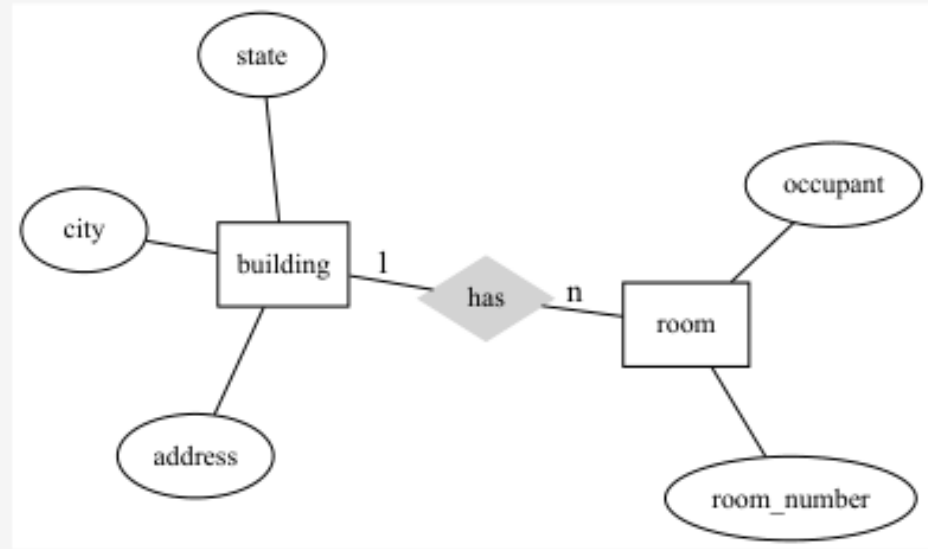
Rooms have room_numbers and occupants.

**Identify the entities, attributes and relationships**



# Chen notation

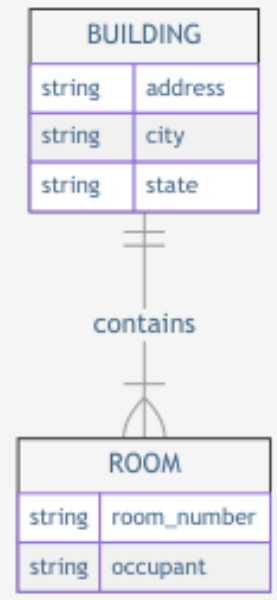
```
graph ER {
    layout=neato;
  scale=1.1;
    node [shape=box]; building; room;
    node [shape=ellipse]; address, city, s
    node [shape=diamond,style=filled,color
    building -- has [label="1",len=1.00]
    has -- room [label="n",len=1.00]
    building -- address
    building -- city
    building -- state
  room -- room_number
    room -- occupant
}
```




Link to graphviz on-line editor

# Crow's Foot notation

```
1  erDiagram
2  BUILDING ||--|{ ROOM : contains
3
4  BUILDING{
5    string address
6    string city
7    string state
8  }
9
10 ROOM {
11   string room_number
12   string occupants
13 }
```




Link to mermaid live



# 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.



# Example

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.**

Link to graphviz on-line editor

Link to mermaid live

# Housekeeping

Another busy week, mostly with software set up.

- Start to work on homework 2, DUE FRIDAY!
- Review the videos:
  - Database design, ER diagramming, ER diagrams to relational tables.
- Kick around database ideas with your teammate(s) and begin to draw pictures!
- **Quiz 1** next week, topics: Database design, ER diagrams. More details on Wednesday.

| Module | Week | Date | Day | Lectures/Quizzes | Deliverables/Notes |
|---|---|---|---|---|---|
| ER Models | 2 | 8/28 | Mon | MTG2: L2 (Entity-relation models 1) | Last day of add/drop |
| 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) | |