Database Systems

Tutorial Week 3

Objectives

I. Entity-Relationship (ER) modelling review

II. Case study — use the previous week’s case study to design a conceptual model

III. Convert the conceptual model to a logical model

IV. Introduce the notion of physical model

Entity-Relationship (ER) Modelling Review

Entities:

- Real-life objects e.g. employees, employers, insurance policies, sporting matches

Weak entities:

- Entity that can’t be uniquely identified without referring to another entity
- E.g. company insurance policy insures an employee and any dependents
  - Dependent can’t exist in system without employee
  - I.e. dependent can’t get insurance as a dependent unless they’re a dependent of an employee
  - “Dependent” = weak entity
  - “Employee” = owner entity

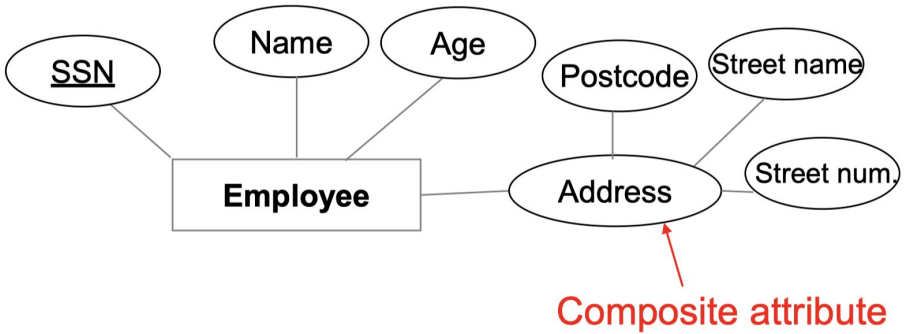
Entity-Relationship (ER) Modelling Review

Attributes:

- Information about an entity e.g. ID, date of birth

Composite attribute:

- An attribute composed of smaller ones e.g.



Entity-Relationship (ER) Modelling Review

Attributes:

- Information about an entity e.g. ID, date of birth

Multi-valued attribute:

- Can take multiple values of the *same data type* e.g. phone number

Entity-Relationship (ER) Modelling Review

Business rules:

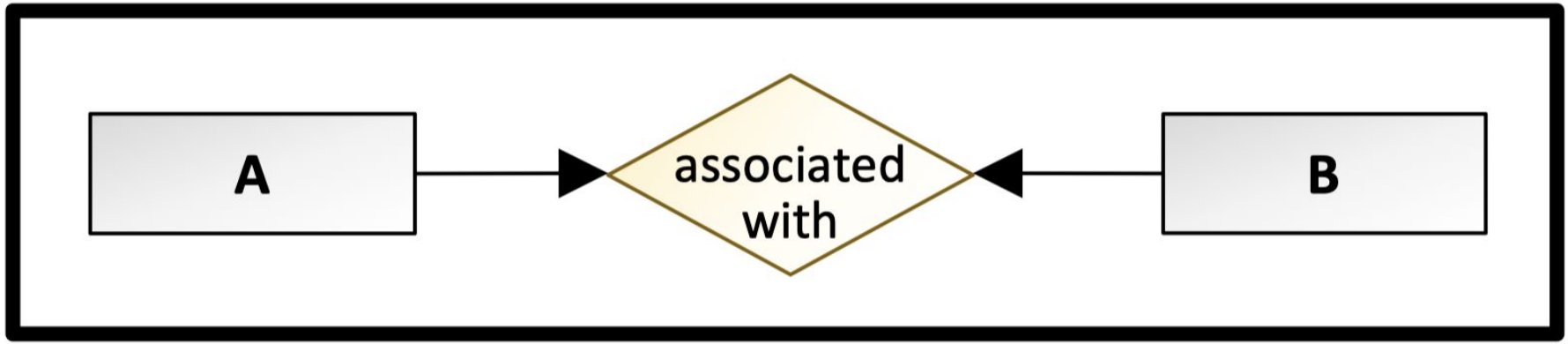
- Define entities and attributes
- Define relationships and constraints between entities
- A relationship can be between two or more entities
- Two important constraints that define properties of a relationship:
  - Key constraints
  - Participation constraints

Key Constraints

- Determine the number of objects taking part in the relationship set
- 3 types of key constraints
  - One-to-one
  - One-to-many
  - Many-to-many
- “Many” represented with a line
- “One” represented with an arrow

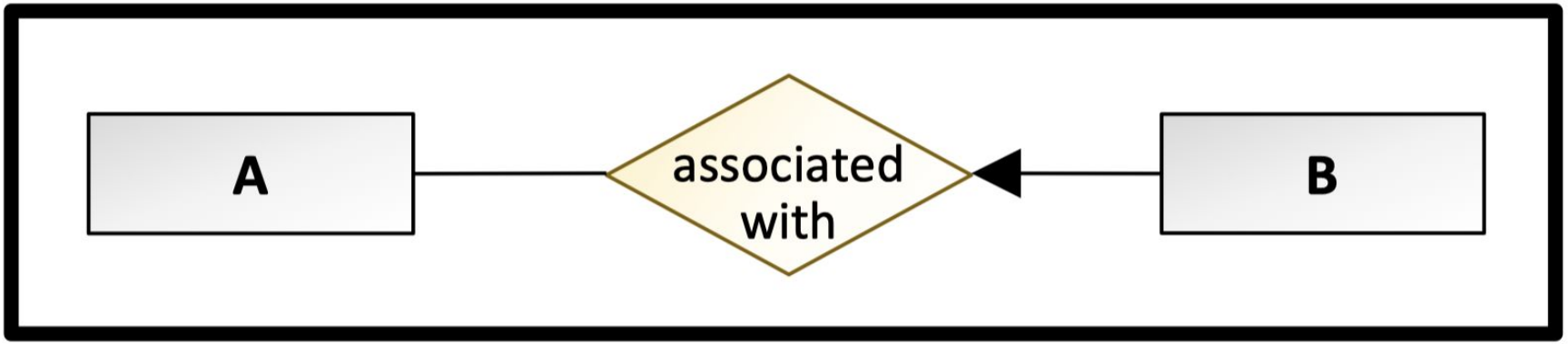
Key Constraints

- One-to-one
  - An entity in A can have at most one association with an entity in B and vice versa.



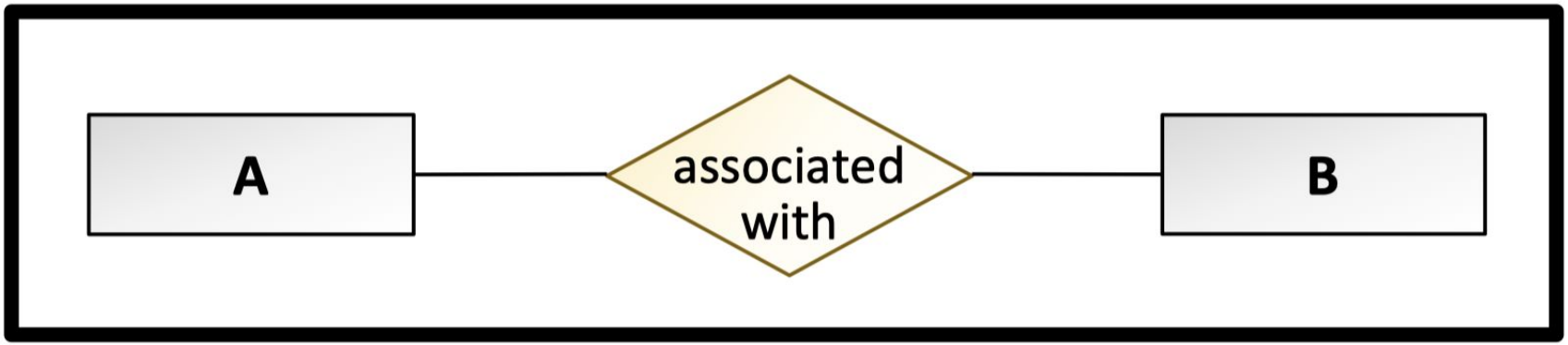
Key Constraints

- One-to-many
  - An entity in A can have association with many entities in B. However, an entity in B is only associated with at most one entity in A.



Key Constraints

- Many-to-many
  - An entity in A can have association with many entities in B and vice versa.



Participation Constraints

Participation of an entity in a relationship can either be “total” or “partial”.

Total participation = mandatory for an entity to take part in a relationship

- Represented by a bold line

Partial participation = optional for an entity to take part in a relationship

- Represented by a thin line

Cheat Sheet

Chen	Cardinality	Ex:	Crows
	0 : m	"may have one or more"	
	1 : m	"at least one or more"	
	0 : 1	"may have at most one"	
	1 : 1	"must have one and only one"	

bold = mandatory
line = optional

vertical line = mandatory
circle = optional

Case Study — Design a Conceptual Model

A cinema chain operates a number of cinemas. Each cinema has several screens, numbered starting from 1. The chain keeps track of the size (in feet) and seating capacity of every screen, as well as whether the screen offers the Gold Class experience.

The cinema chain owns hundreds of movie projectors – both film projectors (16 mm and 35 mm) and digital projectors (2D and 3D). The chain stores key information about each projector, namely its serial number, model number, resolution and hours of use. Each movie screen has space for a single projector; technicians must be able to identify which screen each projector is currently projecting onto.

A wide range of movies are shown at these cinemas. The system should keep track of the last time a movie was shown on a particular screen. The marketing department needs to know the movie's title and year of release, along with the movie's rating (G, PG, M, MA15+ or R18+).

Each cinema has a numeric ID, name and address. For cinemas that are not owned outright, the business also keeps track of yearly rent. The system needs to be able to generate weekly activity reports for the chain's chief operating officer.

Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.
c. Apply constraints (**key constraints** and **participation constraints**) to the relationships.
d. Add **attributes** which describe the entities and relationships.
e. Finalise your conceptual model by marking **weak entities**, **identifying relationships** and **key attributes**.

Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.

- Cinema
- Screen
- Projector
- Movie

Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.

- "Each cinema has several screens"
  → screen is **located in** cinema (or cinema **contains** screen)

Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.

- "Technicians must be able to identify which movie screen each projector is currently projecting onto"
  → projector **projects onto** movie screen (or movie screen is **projected onto by** projector)

Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.

- "The system should keep track of the last time a movie was shown on a particular screen"
  → screen **shows** movie (or movie is **shown on** screen)

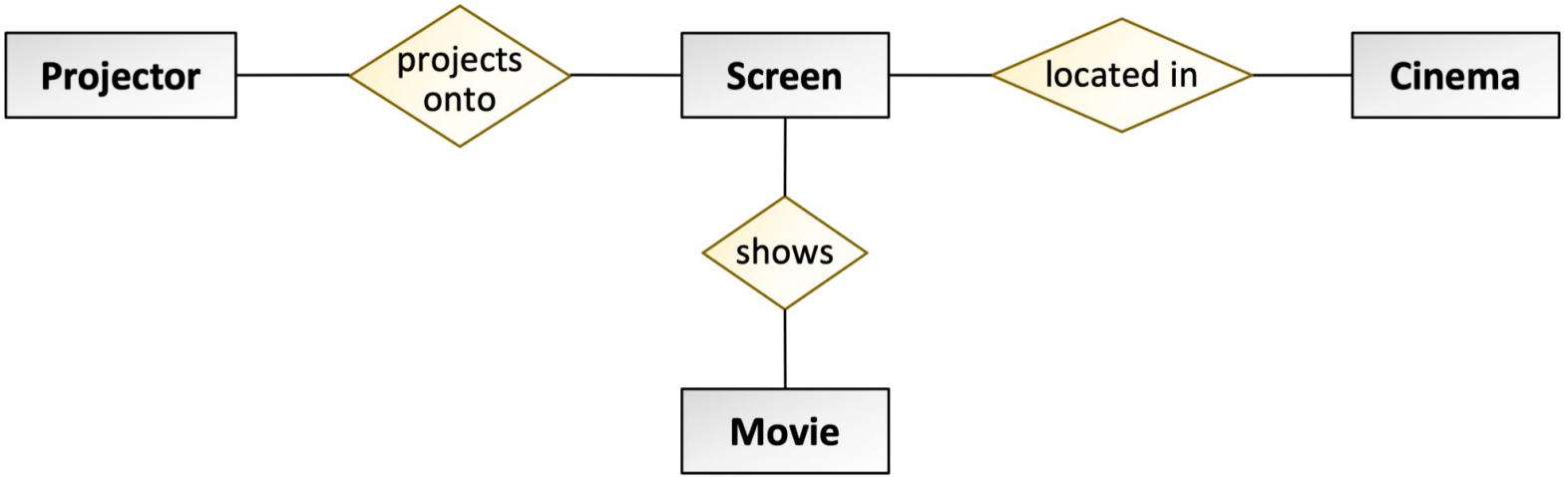
Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.

- Avoid using vague words like "has" to label your relationships!

Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.



Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.
c. Apply constraints (**key constraints** and **participation constraints**) to the relationships.

"Located in" relationship:

- A screen **must** be located in **exactly one** cinema
  → screen is "mandatory one", drawn as a bold arrow
- A cinema **must** contain **at least one** screen
  → cinema is "mandatory many", drawn as a bold line

Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.
c. Apply constraints (**key constraints** and **participation constraints**) to the relationships.

"Projects onto" relationship:

- A projector **may** project onto **exactly one** screen
  → projector is "optional one", drawn as a thin arrow
- A screen **may** be projected onto by **exactly one** projector
  → screen is "optional one", drawn as a thin arrow

Case Study — Design a Conceptual Model

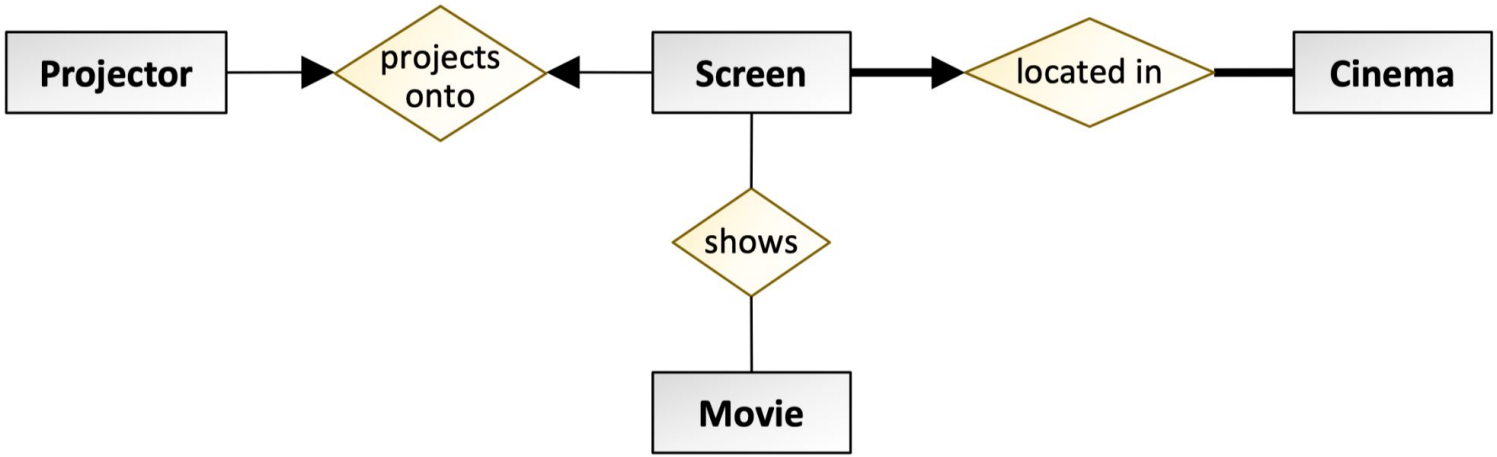
a. Revise last week's identified **entities**.
b. Form **relationships** between entities.
c. Apply constraints (**key constraints** and **participation constraints**) to the relationships.

"Shows" relationship:

- A screen **may** show **many** movies (over time)
  → screen is "optional many", drawn as a thin line
- A movie **may** be shown on **many** screens (over time)
  → movie is "optional many", drawn as a thin line

Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.
c. Apply constraints (**key constraints** and **participation constraints**) to the relationships.

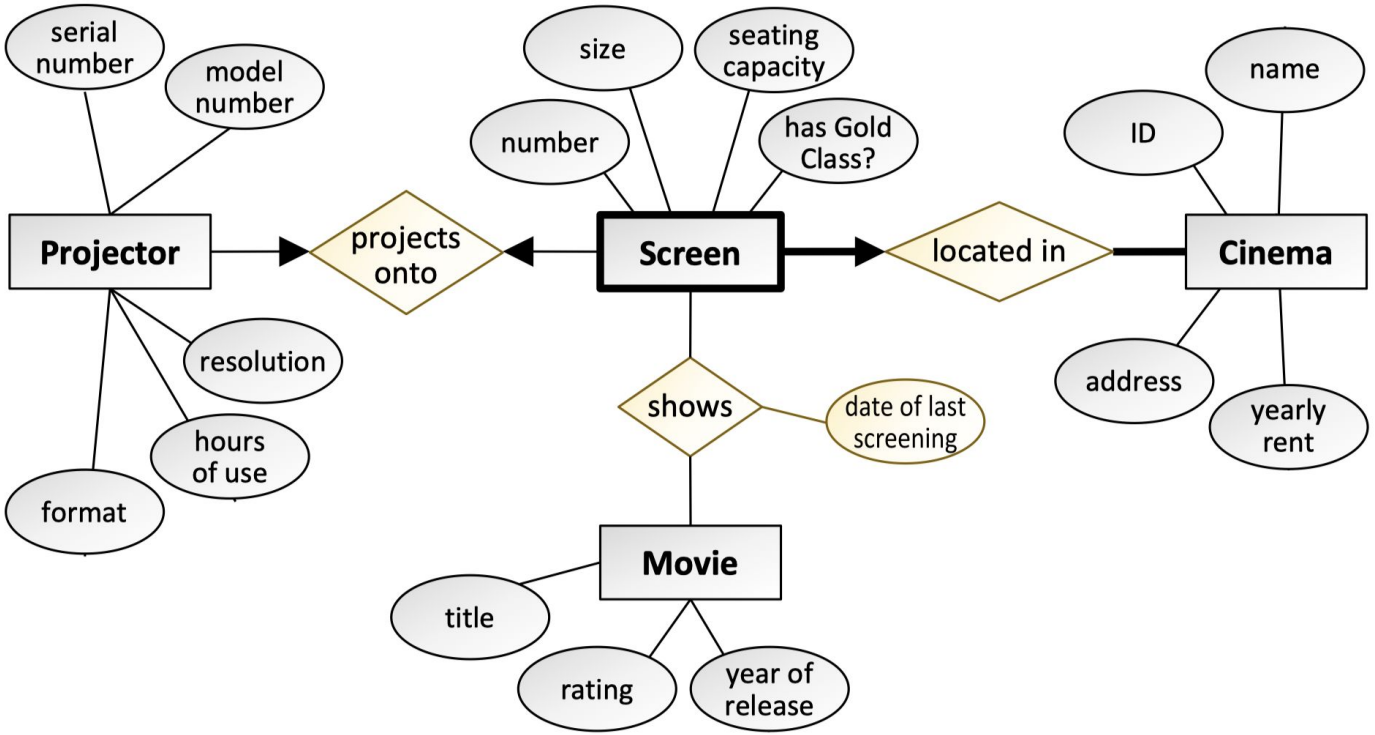


Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.
c. Apply constraints (**key constraints** and **participation constraints**) to the relationships.
d. Add **attributes** which describe the entities and relationships.

- Look for attributes for the relationships as well!!!
- "The system should keep track of the last **time** a movie was shown on a particular screen"
  → "Shows" relationship has "date of last screening" attribute

Case Study — Design a Conceptual Model



Case Study — Design a Conceptual Model

a. Revise last week's identified **entities**.
b. Form **relationships** between entities.
c. Apply constraints (**key constraints** and **participation constraints**) to the relationships.
d. Add **attributes** which describe the entities and relationships.
e. Finalise your conceptual model by marking **weak entities**, **identifying relationships** and **key attributes**.

Case Study — Design a Conceptual Model

Weak entities and **identifying relationships**:

- Each weak entity has one and only one strong entity to depend on, so consider entities with **bold arrows** coming out of them
  - These are potentially weak entities
- Consider the “screen” entity
  - If you remove a cinema from the system, you have to remove its screens
  - Screen lacks its own unique identification — it’s only identified in terms of the cinema it’s located in e.g. “Melbourne Central cinema, screen 1”

  → Screen is a weak entity, and “located in” is its identifying relationship
- Use bold outlines for weak entities and identifying relationships

Case Study — Design a Conceptual Model

Key attributes:

- Pick an attribute for each entity that uniquely identifies every record

- Cinema’s PK is ID

Case Study — Design a Conceptual Model

Key attributes:

- Pick an attribute for each entity that uniquely identifies every record

- Projector’s PK is serial number

Case Study — Design a Conceptual Model

Key attributes:

- Pick an attribute for each entity that uniquely identifies every record

- “Movie”? Add our own **surrogate key.**
- Unique, meaningless number used to identify each record e.g. fast food order numbers
- Movie’s PK is ID attribute that we artificially create

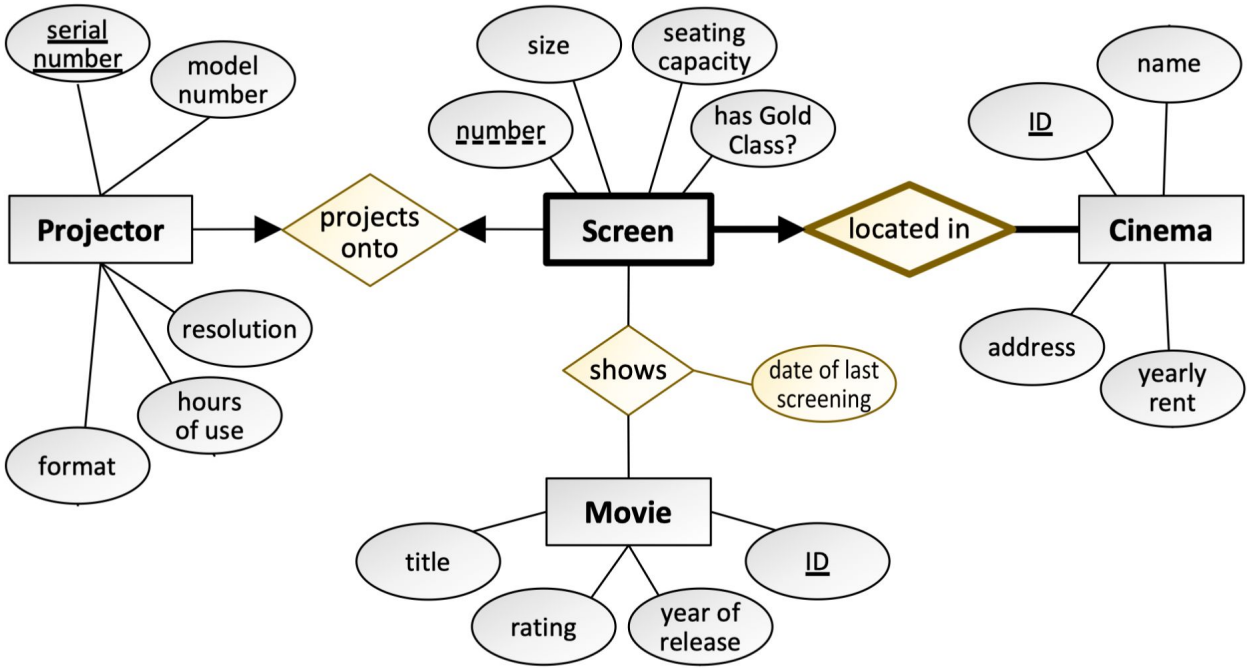
Case Study — Design a Conceptual Model

Key attributes:

- Pick an attribute for each entity that uniquely identifies every record

- Screen is identified by the cinema ID and the screen number together.
- Screen has a partial key number

Case Study — Design a Conceptual Model



Logical and Physical Modelling

What needs to be changed to convert a conceptual ER model to a logical design?

- Resolve multivalued attributes by splitting them into separate tables
- Resolve composite attributes by redrawing the component parts as separate attributes
- Resolve relationships by adding foreign keys and associative entities to the model, and placing relationship attributes in the correct location

- Also at this stage, names are conventionally changed into CamelCase
  e.g. “seating capacity” → “SeatingCapacity”

Logical and Physical Modelling

Develop a logical design for the case study.

- Recall: a foreign key (FK) is a column of one table which refers to the primary key of another table.

Logical and Physical Modelling

Develop a logical design for the case study.

- Resolve one-to-one relationships by adding a FK on either table
- **Mandatory** side of the relationship gets the FK in Chen's notation
- To resolve the Projector-Screen relationship, can add a FK on either the Projector table (that references the PK of the Screen table) or a FK on the Screen table (that references the PK of the Projector table)
- Let's go with the 2nd option:

FK

Screen (ScreenNumber, Size, SeatingCapacity, HasGoldClass, ProjectorSerialNumber)

Logical and Physical Modelling

Develop a logical design for the case study.

- Resolve one-to-many relationships by adding a FK on the **one** side of the relationship (Chen's notation)
- To resolve the Cinema-Screen relationship, add FK on the Screen table
- Because this is an identifying relationship, this FK "cinema ID" will also be a primary key ("primary foreign key")
- CinemaID and ScreenNumber together form the composite primary key

FK FK
Screen (CinemaID, ScreenNumber, Size, SeatingCapacity, HasGoldClass, ProjectorSerialNumber)

Logical and Physical Modelling

Develop a logical design for the case study.

- Resolve many-to-many relationships by creating a new entity (“**associative entity**”)
- The associative entity contains PFKs for each table in the relationship
- To resolve the Screen-Movie relationship, make an associative entity called “MovieScreening”
- Include composite PK of Screen and PK of Movie as primary foreign keys:

MovieScreening (CinemaID, ScreenNumber, MovieID)

FK FK FK

Logical and Physical Modelling

Develop a logical design for the case study.

- Relationship attribute, “date of last screening”?
- Relationship attributes always placed in same table as foreign key(s)
- In this case, foreign keys located in **associative entity**, so “date of last screening” goes here as an additional non-key column:

MovieScreening (CinemaID, ScreenNumber, MovieID, DateOfLastScreening)

FK FK FK

Logical and Physical Modelling

Develop a logical design for the case study.

Cinema (CinemaID, Name, Address, YearlyRent)

Screen (CinemaID (FK), ScreenNumber, Size, SeatingCapacity, HasGoldClass, ProjectorSerialNumber (FK))

MovieScreening (CinemaID (FK), ScreenNumber (FK), MovieID (FK), DateOfLastScreening)

Movie (MovieID, Title, YearOfRelease, Rating)

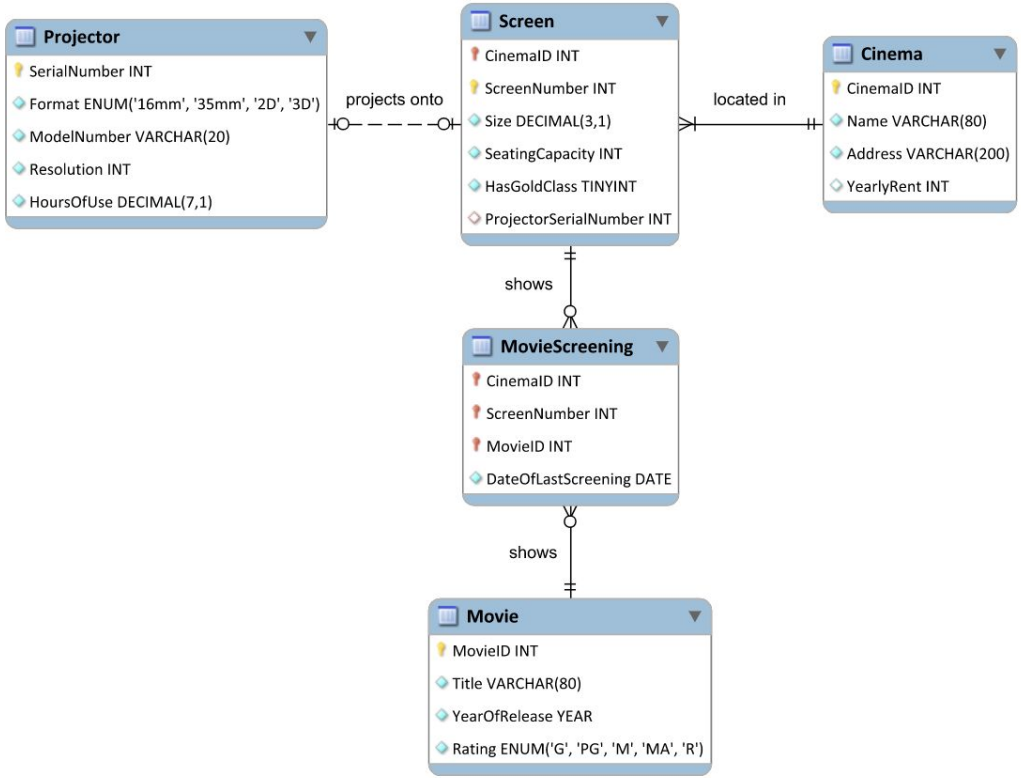
Projector (SerialNumber, Format, ModelNumber, Resolution, HoursOfUse)

Logical and Physical Modelling

What will you change in the logical model to generate a physical model?

- For each column, add **data type** and **NULL/NOT NULL constraint** (whether it’s optional or mandatory)
- FK columns must have same data type as PK column they refer to
- Primary key columns must always be NOT NULL
- For NULL/NOT NULL of FKs, consider participation constraints of the conceptual model — if table’s participation in relationship is mandatory, FK must be NOT NULL

Physical Model



Week 3 Lab

- Canvas → Modules → Week 3 → Lab → L03 Modelling 2 (PDF)
- Objectives:
  - Learn about MySQL data types
    - Choose proper data types for physical ER models
  - Create a physical data model with the MySQL Workbench modelling tool
- Breakout rooms, “ask for help” button if you need help