INFO90002 Tutorial Week 3

Objectives:

This tutorial will cover:

I. Entity-Relationship (ER) modelling review
II. Case study – Use a case study to design a conceptual model
III. Convert the conceptual model to a logical model
IV. Introduce the notion of physical model – **complete as homework**

Exercises:

1. ER Review – selected concepts

NOTE for students: *This is a brief summary of selected concepts taught in lectures 3 and 4. The lectures contain detailed content related to these and many more key concepts. These notes should be considered quick revision instead of the sole resource for the course material.*

- Entity, weak entity

Any real-world object distinguishable from other objects is defined as an Entity. In relation to the database, entity is a single person, place or thing about which data can be stored. An entity can be concrete such as a person or a book or abstract such as concept, holiday etc. Together these entities will constitute 'entity set' such as Customers, Cities, and Books etc. However, there are some entities that require other entities to exist. Such entities are called weak entities. An example would be a company insurance policy that insures an employee and any dependents. The dependent cannot exist in the system without the employee; that is, a person cannot get insurance coverage as a dependent unless the person is a dependent of an employee. Therefore "Dependent" will be a weak entity.

- Attribute

The characteristics representing an entity are called attributes. All entities in the entity set will have the same set of attributes. For example, for entity Employee, every employee will have name, ssn, DOB and address saved. Each attribute has a domain i.e. set of permitted values

- Business rules to relationships – key constraints and participating constraints

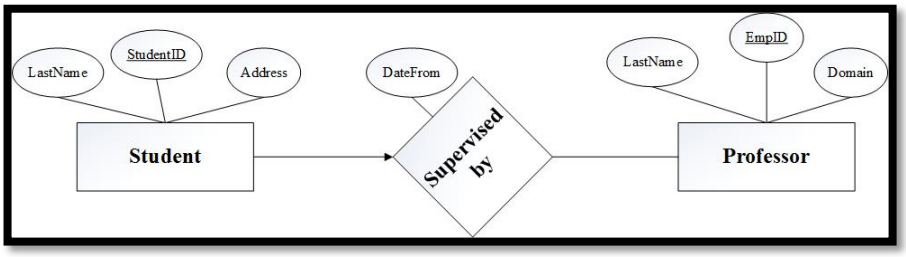
Business rules are used to define the entities, attributes their relationships and constraints. Identification and documentation of the business rules can help design the database. Mostly the source of such business rules are managers, policy makers, standards, written documents and interviews of the users. However in this subject you will be given a case study and you will be expected to extract such business rules from that written document to design a database (such as the case study related to the cinema chain you saw last week and will see again this week).

In terms of database design, the business rules are not only used to define the entities and their attributes but also help define the relationships and their constraints between entities. A relationship can be between 2 or more entities. There are two important types of constraints that define any relationship properties, key constraints and participating constraints.

- *Key Constraints:*

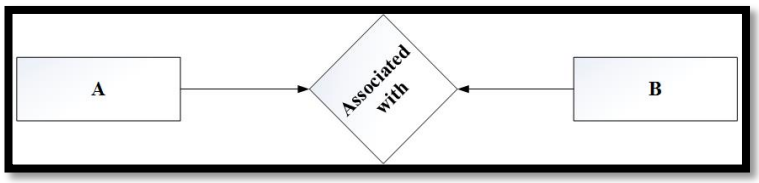
  Consider an example from a University "A student is allowed to be supervised by at most one professor but the professor on the other hand can supervise more than one student".

  In this example many students can be supervised by the same professor but each student can only have at most one supervisor. This is an example of 'many-to-one constraint" and can be represented by arrow head to indicate the key constraint:

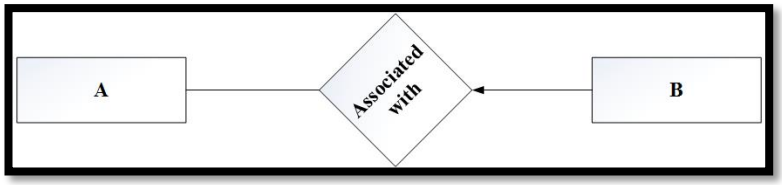


  The other key constraints such as 'one-to-one', 'one-to-many' and 'many-to-many' are represented as following (Consider the relationship between entity A and entity B):

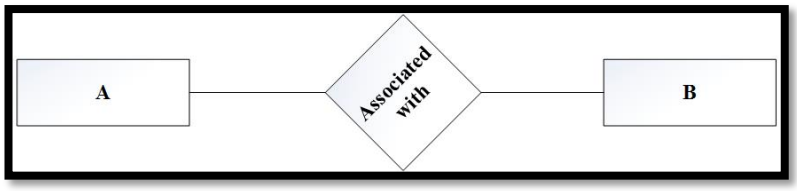
  - *One-to-one: An entity in A can have at most one association with an entity in B and vice versa.*



  - *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.*

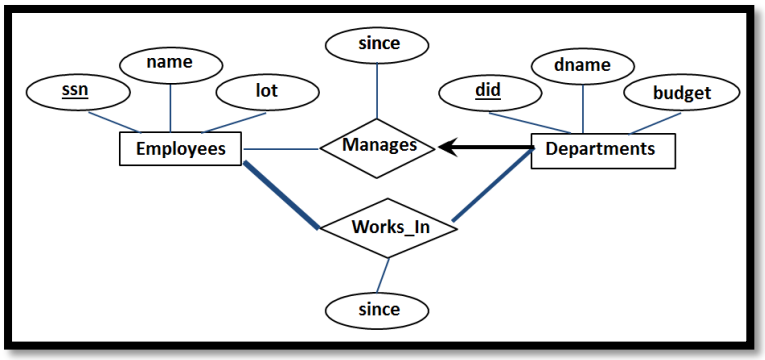


  - *Many-to-many: An entity in A can have association with many entities in B and vice versa.*



- *Participation Constraints*

  The participation of an entity set A in a relationship can either be 'total' or 'partial'. Using the same example as lecture 3 slide 17, each department must have one manager hence the participation of Department in 'Manages' is said to be total. Whereas not every employee is manager of some department therefore Employees entity's participation is partial in 'Manages' as shown below (from lecture):



2. Consider the following case study:

A cinema chain operates a number of cinemas. Each cinema has several theatres, numbered starting from 1. The chain keeps track of the size (in meters) and seating capacity of every theatre, as well as whether the theatre 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 cinema chain stores key information about each projector, namely its serial number, model number, resolution and hours of use. Each movie theatre has space for a single projector; technicians must be able to identify which theatre each projector is currently in use.

A wide range of movies are shown at these cinemas. The system should keep track of the last time a movie was shown in a particular theatre belonging to a particular cinema. 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.

3. TASK. Follow the steps to create a conceptual model in Chen's notation:

Let's move through the conceptual design process step-by-step.

In your assessment we don't expect you to 'show your working' like this. It is sufficient to show your final model(s) and, if requested, add a brief description of any assumptions you made.

I. Identify the entities.

- Cinema
- Theatre
- Projector
- Movie

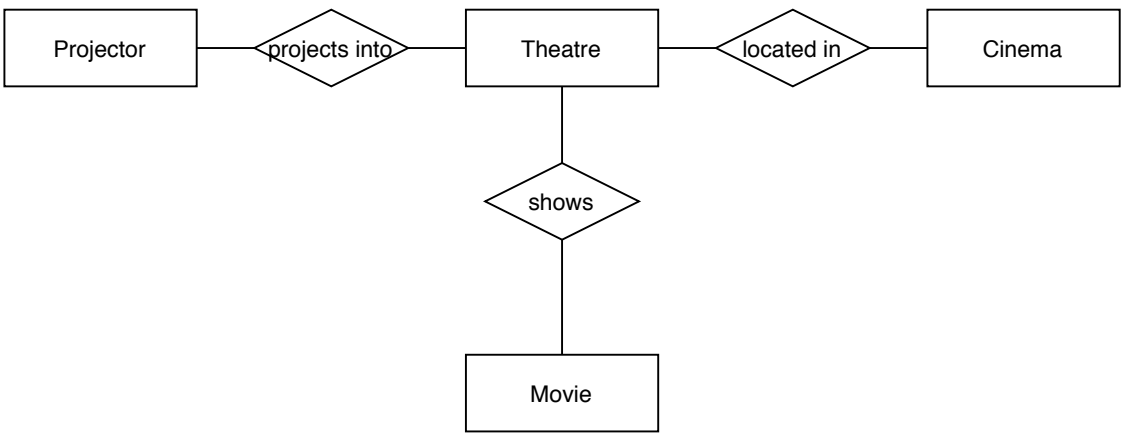
'Cinema chain' is not an entity in this model. You do not normally include the actual business or company whose business processes you are modelling. This is because there is only one instance of this company, and there is no data to store about it in any case.

II. Form relationships between entities.

Each cinema has several theatres"

- Theatre is **located in** cinema (or cinema **contains** theatre)
- "Technicians must be able to identify which movie theatre each projector is currently projecting onto"
- Projector **projects onto** movie theatre (or movie theatre is **projected onto by** projector)
- "The system should keep track of the last time a movie was shown on a particular theatre"
- Theatre **shows** movie (or movie is **shown on** theatre)

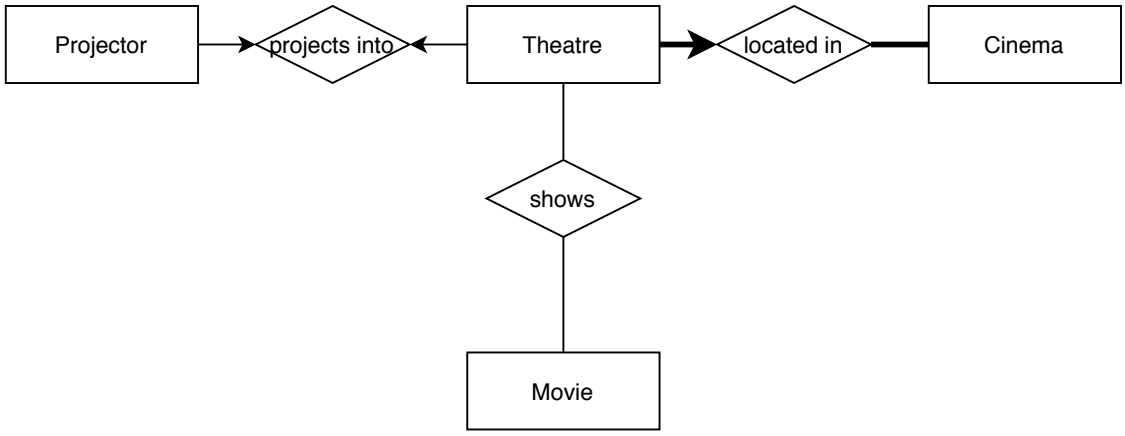
Remember, do not use vague words like "has" to label your relationships.



III. Apply constraints (key constraints and participation constraints) to the relationships.

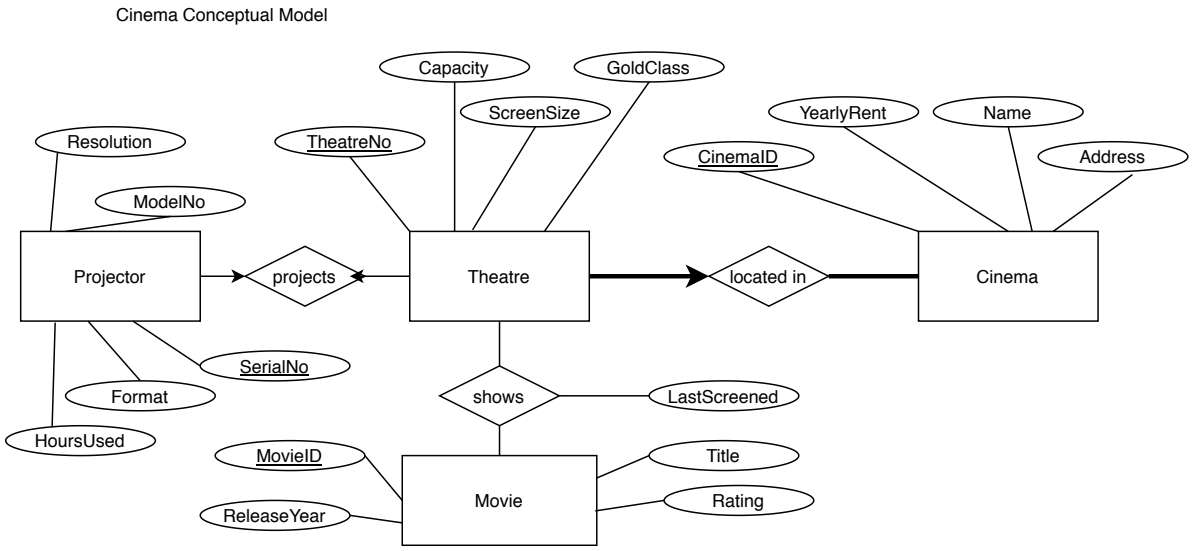
- "Located in" relationship:
  A theatre **must** be located in **exactly one** cinema.
- theatre is "mandatory one", drawn as a bold arrow (mandatory = bold, one = arrow)
- A cinema **must** contain **at least one** theatre.
- cinema is "mandatory many", drawn as a bold line (mandatory = bold, many = line with no arrow)
- "Projects onto" relationship:
  A projector **may** project onto **exactly one** theatre.
- projector is "optional one", drawn as a thin arrow (optional = thin, one = arrow)
  (Why optional? If it is a spare projector, under repair, etc, then it is not projecting onto a theatre.)
- A theatre **may** be projected onto by **exactly one** projector.
- theatre is "optional one", drawn as a thin arrow (optional = thin, one = arrow)
  (Again, some theatres might briefly have no projector, due to cinema maintenance or the swapping-over of projectors.)
- "Shows" relationship:
  A theatre **may** show **many** movies (over time).
- theatre is "optional many", drawn as a thin line (optional = thin, many = line with no arrow)
- A movie **may** be shown on **many** theatres (over time).
- movie is "optional many", drawn as a thin line (optional = thin, many = line with no arrow)

Not all constraints are spelt out in the requirements analysis. You need to use common sense to fill in the gaps.

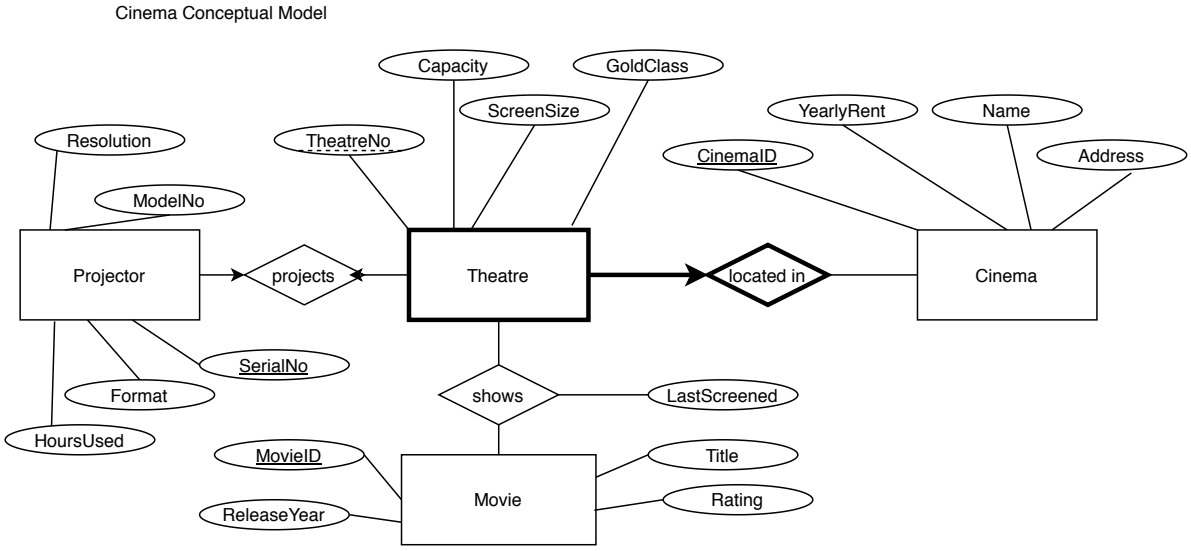


IV. Add attributes which describe the entities and relationships.

- "Each cinema has a **numeric ID**, **name** and **address**. For cinemas that are not owned outright, the business also keeps track of **yearly rent**."
- Cinema (ID, name, address, yearly rent)
- Theatres are "… **number**ed starting from 1. The chain keeps track of the **size** (in feet) and **seating capacity** of every theatre, as well as **whether the theatre offers the Gold Class experience**."
- Theatre (number, size, seating capacity, has Gold Class?)
- There are "… **film projectors (16 mm and 35 mm) and digital projectors (2D and 3D)**. Alongside its **serial number**, the chain stores key information about each projector, namely its **model number**, **resolution** and **hours of use**."
- Projector (format [i.e. 16 mm film/35 mm film/2D digital/3D digital], serial number, model number, resolution, hours of use)
- "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+)."
- Movie (title, year of release, rating)
- Don't forget to look for attributes for the relationships in your model, not just the entities!
- "Each cinema has several theatres"
- No relationship attributes for the "located in" relationship
- "Technicians must be able to identify which movie theatre each projector is currently projecting onto"
- No relationship attributes for the "projects onto" relationship
- "The system should keep track of the last **time** a movie was shown on a particular theatre"
- "Shows" relationship (date of last screening)



V. Finalise your conceptual model by marking weak entities, identifying relationships and key attributes.



Logical and physical modelling

4. TASK. What needs to be changed to convert a conceptual design to a logical design?

To translate a conceptual ER model to a logical design, we need to perform three tasks:

- 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.

Tip: Conventionally, names are changed into CamelCase at the logical stage, so "seating capacity" becomes "SeatingCapacity".

5. TASK. Develop a logical design for the above case study.

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

We will develop a logical design for the cinema case study.

There are no multivalued or composite attributes in this simple model. These will be discussed in Tutorial 4.

One-to-one relationships are resolved by adding a foreign key on either table, giving preference to the table that has mandatory participation in the relationship if there is only one.

> A foreign key is a column (or set of columns) of one table which refers to the primary key of another table.

To resolve the Projector-Theatre relationship, we could add a foreign key on the Projector table that references the primary key of the Theatre table, or we could add a foreign key on the Theatre table that references the primary key of the Projector table. Let's add a foreign key on the Theatre table:

Theatre (TheatreNumber, Size, SeatingCapacity, HasGoldClass, ProjectorSerialNumber^{FK})

One-to-many relationships are resolved by adding a foreign key on the **one** side of the relationship.

To resolve the Cinema-Theatre relationship, we add a foreign key on the Theatre table. Because this is an **identifying relationship**, this foreign key, "cinema ID", will also be a primary key (known as a "primary foreign key"):

Theatre (^{FK}CinemaID, TheatreNumber, Size, SeatingCapacity, HasGoldClass, Projector^{FK}SerialNumber)

Many-to-many relationships are resolved by creating a new entity called an "associative entity". This entity contains primary foreign keys for each table in the relationship.

To resolve the Theatre-Movie relationship, we need to make an associative entity. Let's call it "MovieTheatre". Remember that the primary key of Theatre is now made up of two columns, which must both be included in MovieTheatre as a foreign key:

MovieTheatre (^{FK}CinemaID, ^{FK}TheatreNumber, ^{FK}MovieID)

What about the relationship attribute, "date of last screening"? The rule of thumb for relationship attributes is that they always sit alongside the foreign key(s). In this case, the foreign keys are located in the associative entity, so we place "date of last screening" here as an additional non-key column:

MovieTheatre (^{FK}CinemaID, ^{FK}TheatreNumber, ^{FK}MovieID, DateOfLastScreening)

Here is the completed logical design. Notice that the Cinema, Movie and Projector tables have remained unchanged from the conceptual phase:

Cinema (CinemaID, Name, Address, YearlyRent)

Theatre (FK CinemaID, TheatreNumber, Size, SeatingCapacity, HasGoldClass, FK ProjectorSerialNumber)

MovieTheatre (FK CinemaID, FK TheatreNumber, FK MovieID, DateOfLastScreening)

Movie (MovieID, Title, YearOfRelease, Rating)

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

Technically, participation constraints are still part of the logical design, although we don't mention them in this textual notation. The participation constraints will be needed again during development of the physical model.

HOMEWORK

6. TASK. Using MySQL Workbench create a physical E.R. model of the case study.