INFO20003 Tutorial – Week 3 Solutions

Objectives:

This tutorial will cover:

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

Exercises:

1. ER Review – fundamental 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

  An **entity** is defined as a real-world object or concept distinguishable from other objects. In relation to databases, an entity is a single person, place or thing about which data can be stored. An entity can be concrete, such as "person" or "book", or abstract, such as "currency" or "sporting match".

  Some entities require other entities in order to exist. If an entity cannot be uniquely identified without referring to another entity, it is called a **weak entity**. 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 describing an entity are called **attributes**. For example, for an "employee" entity, every employee will have their name, date of birth and address stored. Each attribute has a domain – a set of permitted values.

- Business rules to relationships – key constraints and participation constraints

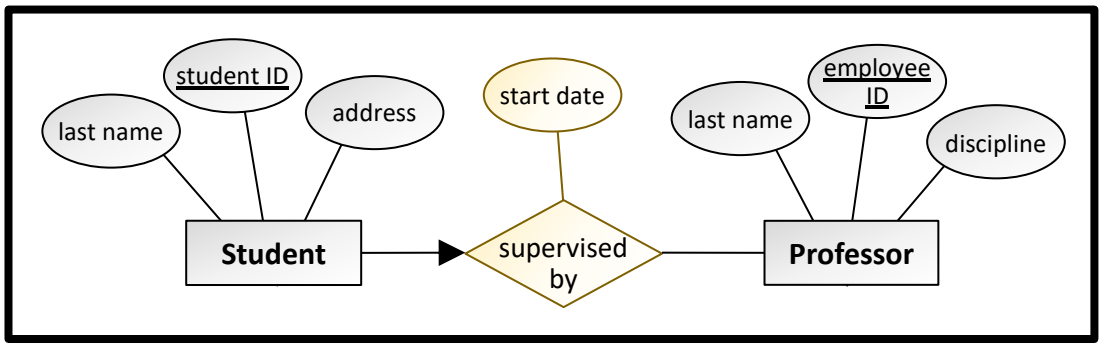
  Business rules are used to define the entities, attributes, their relationships and constraints. Identification and documentation of the business rules is an important part of designing the database. Mostly the source of such business rules are managers, policy makers, standards, written documents and 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 two or more entities. There are two important types of

constraints that define the properties of a relationship – key constraints and participating constraints.

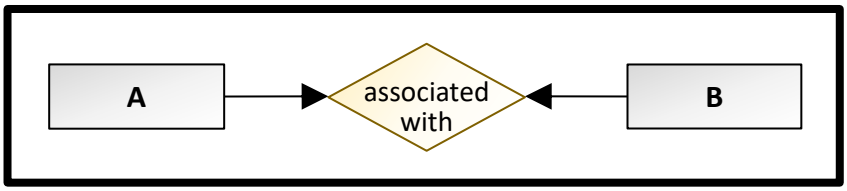
Key constraints: Consider an example from a university: "A student is allowed to be supervised by at most one professor, but the professor on the other hand can supervise more than one student".

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 a "many-to-one" key constraint and can be represented by an arrowhead:

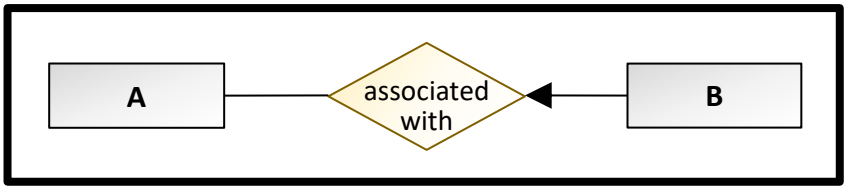


The three types of key constraints, "one-to-one", "one-to-many" and "many-to-many", are represented as follows. Consider a relationship between entity A and entity B:

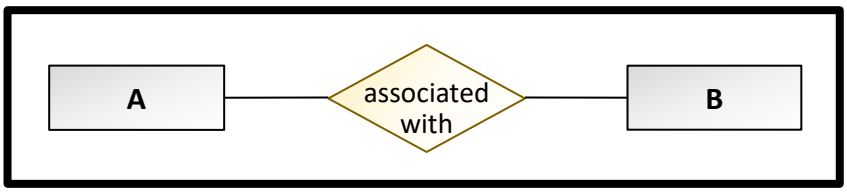
- One-to-one: An entity in A can have at most one association with an entity in B and vice versa.



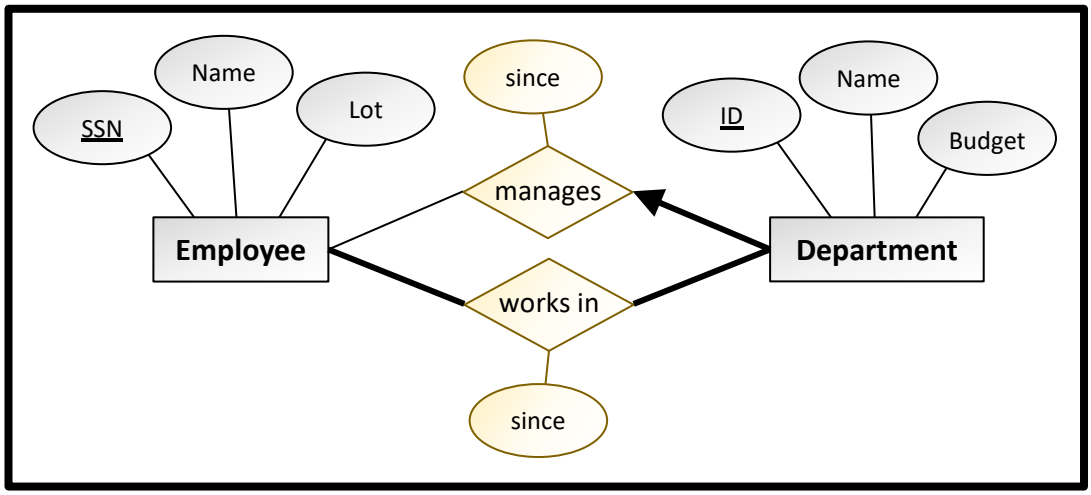
- One-to-many: An entity in A can have association with many entities in B. However, an entity in B is only associated with at most one entity in A.



- 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 in a relationship can either be "total" or "partial". Using the same example as lecture 3 slide 13, each department must have one manager, hence the participation of Department in "manages" is said to be total. On the other hand, not every employee is the manager of a department, so Employees entity's participation is partial in "manages", as shown below:



2. Consider the following case study:

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.

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

Continued over page

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

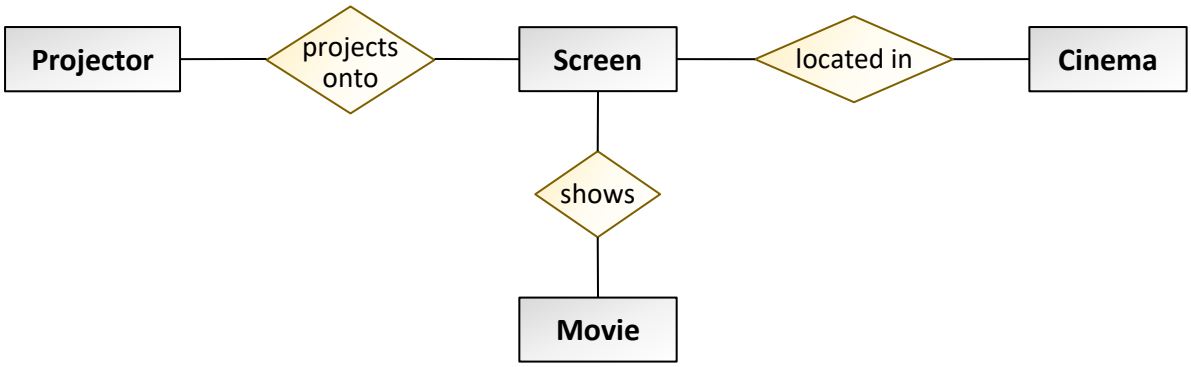
> In your assessment we don't expect you to "show your working" like this. It is sufficient to show your final model(s) and, if requested, add a brief description of any assumptions you made.

a. Revise last week's identified **entities**.
   - Cinema
   - Screen
   - 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.

b. Form **relationships** between entities.
   - "Each cinema has several screens"
     → screen is **located in** cinema (or cinema **contains** screen)
   - "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)
   - "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)

> Avoid using vague words like "has" to label your relationships.



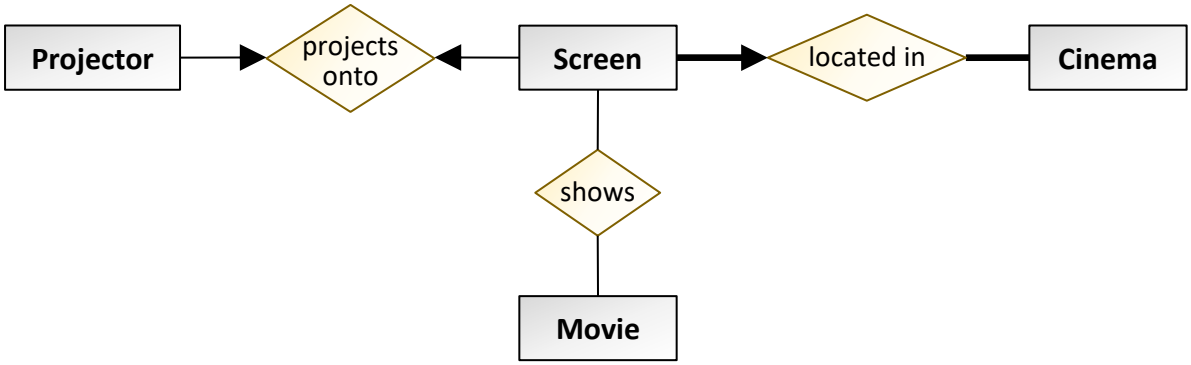
c. Apply **constraints** (key constraints and participation constraints) to the relationships.

- "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

- "Projects onto" relationship:
  A projector **may** project onto **exactly one** screen.
  → projector is "optional one", drawn as a thin arrow
  (Why optional? If it is a spare projector, under repair, etc, then it is not projecting onto a screen.)
  A screen **may** be projected onto by **exactly one** projector.
  → screen is "optional one", drawn as a thin arrow
  (Again, some screens might briefly have no projector, due to cinema maintenance or the swapping-over of projectors.)

- "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

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



d. 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)

- Screens are "… **number**ed 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**."
  → Screen (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)

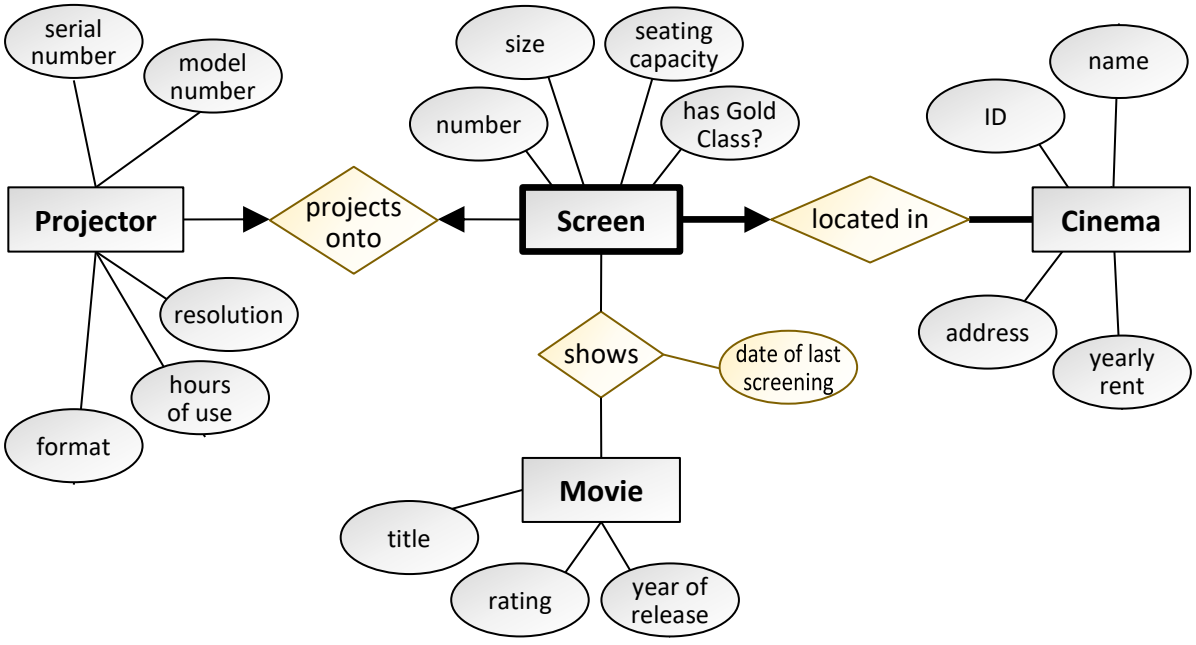
- "The marketing department needs to know the movie's **title** and **year of release**, along with the movie's **rating** (G, PG, M, MA15+ or R18+)."
  → Movie (title, year of release, rating)

Don't forget to look for attributes for the relationships in your model, not just the entities!

- "Each cinema has several screens"
  → No relationship attributes for the "located in" relationship

- "Technicians must be able to identify which movie screen 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 screen"
  → "Shows" relationship (date of last screening)



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

Weak entities and identifying relationships:

- We need to carefully consider all the entities with bold arrows coming out of them. Some of these could be weak entities. In this model, we only need to consider one entity – the screen entity.
- A screen only exists as long as the cinema it is located in continues to exist. If a cinema is removed from the system, so should all of the cinema's screens.
- Moreover, a screen is identified as something like "Melbourne Central cinema, screen 1". Notice how the screen is identified in terms of the cinema it is located in – it lacks its own unique identification.
  → Screen is a weak entity, and "located in" is its identifying relationship. These are both denoted using bold outlines.

Each weak entity must have at least one weak (identifying) relationship.

Key attributes:

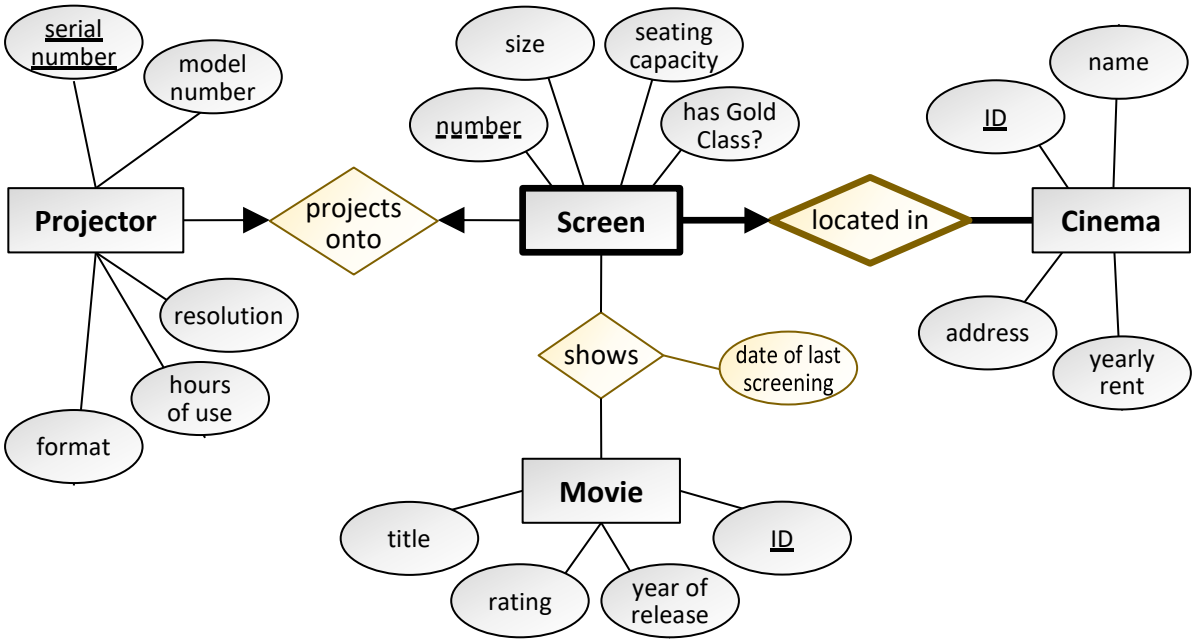
- We need to pick an attribute or set of attributes on each entity that uniquely identifies every instance of that entity.
- For example, every cinema has an ID, and all the IDs are different. This means we can uniquely identify a cinema by its ID.
  → Cinema's primary key is <u>ID</u>
- Likewise, we can uniquely identify a projector by its serial number.
  → Projector's primary key is <u>serial number</u>
- "Movie" does not have an obvious primary key. This means we need to add our own surrogate key.
  → Movie's primary key is a newly-added <u>ID</u> attribute

A surrogate key is a unique, meaningless number used to identify each instance of an entity. Bank account numbers and fast food order numbers are common examples of surrogate keys.

- In part e above, we pointed out that a screen is identified by the cinema ID and the screen ID together. This means that it is identified in part by the "located in" identifying relationship, and in part by the number attribute.
  → Screen has a partial key, <u>number</u>

Partial keys are only relevant for weak entities. Every weak entity must have a partial key.

The final model is over the page.



3. Logical and physical modelling

a. What needs to be changed to convert a conceptual design to a logical design? Develop a logical design for the above case study.

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".

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

Screen (ScreenNumber, Size, SeatingCapacity, HasGoldClass, Projector^{FK}SerialNumber)

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

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

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

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

MovieScreening (CinemaID[FK], ScreenNumber[FK], MovieID[FK])

What about the **relationship attribute**, "date of last screening"? Relationship attributes are always placed in the same table as 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:

MovieScreening (CinemaID[FK], ScreenNumber[FK], MovieID[FK], 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)

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)

> 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.

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

The main change is to go through every column and add a data type and NULL/NOT NULL constraint (i.e. deciding whether a column is optional or mandatory).

For data types:

- Go back to the business case and look for any ideas as to which data type should be used.
- Foreign key columns must have the same data type as the primary key column they refer to.
- If it is not clear which data type to use (for example, "model number"), pick any data type that seems reasonable. Different people might choose different data types for some columns.

For NULL/NOT NULL constraints:

- Primary key columns must always be NOT NULL.
- For foreign keys, look at the conceptual model, specifically the participation constraints. Is the table's participation in the relationship mandatory? If so, the foreign key must be NOT NULL. For example, every Screen must be located in a Cinema, so the CinemaID foreign key on the Screen table is NOT NULL. On the other hand, the ProjectorSerialNumber column on Screen is optional (may be NULL), because not every Screen is associated with a Projector.
- For other columns, think carefully about whether the data in that column is required or optional. For example, every movie has a title, and every screen has a seating capacity, even if that capacity is zero for some reason. But not every cinema has a weekly rent – see the case study: "**For cinemas that are not owned outright**, the business also keeps track of yearly rent."

In the Week 3 lab you used MySQL Workbench to draw a physical model for this case study. It might have looked something like this: