Explanation and Model Solution for "Assignment 1: The Entity-Relationship Model"

Jelle Hellings

3DB3: Databases – Fall 2021

Department of Computing and Software
McMaster University

Foreword

This document consists of three parts. First, we have annotated the original assignment *description* to highlight all information of interest. This part constitutes the main analysis step that should precede the design of the ER-diagram and the writing of the report. Second, we have included the *ER-Diagram* based on our analysis. Finally, we have given a *short report* describing the ER-Diagram, highlighting our choices, and describing all constraints that could not be included in the diagram.

Analysis

In the analysis, we use the following text annotations:

an entity	The highlighted piece of text resembled an entity-like object.
an attribute	The highlighted piece of text resembled an attribute of some entity-like object.
A RELATIONSHIP	The highlighted piece of text resembled a relationship between entity-like objects.
`a constraint`	The highlighted piece of text resembled a constraint on data.
~~not-relevant~~	The highlighted piece of text is not relevant (e.g., context, application details, ...)

~~A local chain of cinemas has decided that their current information system is not sufficient in a modern world in which online ticket sales are the norm. As the first step to renew their system, the owner of the cinema approached a consultant firm that drafted requirements for the new information system. Following is the high-level description of these requirements.~~

~~First, the system will store information about several cinemas.~~ Each **cinema** has a `unique` *name* and an *address*.[1] In the past, cinema names have changed (e.g., due to renovations and grant reopenings).[2] PER CINEMA, THE SYSTEM WILL ALSO MAINTAIN INFORMATION PER ROOM. Each **room** has a `unique` *number* `within the cinema` ~~(e.g., smaller cinemas have Room one to Room five)~~.[3] Per **room**, the system can keep track of the *characteristics of the cinema installation*.[4] These characteristics include

1. the screen type (e.g., normal, purpose-built IMAX, digital IMAX) and screen size;

[1] An entity cinema with attributes name and address. We can use name as the primary key.

[2] Names can change. Non-changing unique identifiers are often easier to work with. Hence, we can opt for a numeric automatically-generated identifier.

[3] The entity cinema owns rooms. Hence, room is a weak entity, there is an identifying relationship to cinema, and the room number can be a partial key.

[4] The simplest solution here is to have an attribute for each basic room characteristic.

2. the projector type (e.g., analog or digital, resolutions, supported 3D modes, 48FPS, and so on);
3. the sound system (e.g., the number of speakers and the supported sound options such Dolby Atmos, Dolby Digital, and DTS); and
4. the available accessibility options.[5]

~~This information is not only available for cinema visitors, but will also be communicated to private parties that are looking to hire a room (e.g., for a corporate event). Finally, per room the system also needs to know the exact seat arrangement, as the online system will allow customers to order tickets for specific seats~~. Hence, the system keeps track, PER **room**, of each **seat** and—per seat—its *location* in the room, the *row* it is in, and the *seat number* (within the row).[6,7] Furthermore, some seats are standard *reserved* for disabled people.[8]

~~Second, the system needs to store information for each screening.~~ Each **screening** is ASSIGNED A SINGLE ROOM and *timeslot* and the system distinguishes between `three types of screenings`[9,10]: **normal screenings** will SHOW A `single` FILM ~~and will be offered via normal ticket prices~~; **special screenings** can SHOW `one-or-more` FILMS ~~with a special ticket price, e.g., a marathon with a higher ticket price or a classic film at a reduced ticket price~~; and **private screenings** ~~(as part of hired rooms)~~.[11] ~~For each **public screening**, the system keeps track which films are shown.~~[12] ~~This film information is provided by the film distributors in a standard format: for now, the system represents this external information via an~~ entity **Film** with an *attribute* `fid`.[13] ~~If a screening will show multiple films (as part of special screening), then each of these films will be shown in the same room and the ticket of the customer assign the same seat during each film.~~

~~Finally, and most importantly, the system will support~~ sales and reservations. Via an (online) sale, customers can buy one or more TICKETS for a specific **screening** and that are assigned a **seat** on sale. ~~Customers that do not feel comfortable with paying online~~ can reserve their seats online ~~and buy a ticket for these reservations at the counter (these reservations will be cancelled 45 minutes before the start of the film)~~.[14] For each sale, we keep track of *how* they are made (online, ticket machine, at the counter) and whether the sale was related to a *reservation*.[15] ~~The cinema chain owners plan to analyze this data to see what factors influence the willingness of customers to buy and reserve tickets via their online system.~~ Furthermore, we keep track of the *paid price* ~~(which can depend on coupons, special actions, payment method, or room and~~

[5] This is under-specified. This could be a simple Boolean option (e.g., a flag that is true when the room is designed with accessibility in mind) or this could be a whole range of features (e.g., special seating, elevator access, ramp access, ...). Without further information, it is impossible to determine what the owner exactly wants. For now, we can choose for the simple Boolean option.

[6] The entity room owns seats and seats are uniquely identified within a room by its row and seat number. Hence, seat is a weak entity, there is an identifying relationship to room, and the row and seat number can be the partial key.

[7] It is unclear what location exactly is. In many cinemas, seat locations are purely determined by row and number. It is possible, however, that the cinema uses a per-room coordinate for each seat so that the seating arrangement can be visualized in any ticket ordering tools. This needs clarification from the cinema owner! For now, we keep location as an attribute.

[8] We can make this a Boolean attribute of each seat.

[9] We can represent a screening as an entity that has a timeslot (a begin-time followed by an end-time) and an exact-one-to-many relationship to its assigned room. We will use a unique identifier for each screening, as no clear identifier is provided in the text. As there are three types of screenings, we can consider using an ISA hierarchy if the three types behave differently.

[10] We note that practically speaking, the combination of rooms and timeslot should be unique (we do not want double booked rooms). Unfortunately, making combinations of rooms and timeslots unique does not eliminate double booked rooms: E.g., the timeslots (October 1, 5pm–7pm) and (October 1, 6pm–8pm) *are* unique values from a data perspective, but also overlap in practice.

[11] There are clear distinctions between the three types of screenings: normal screenings have exactly one film associated to them, special screenings have one-or-more films associated with them, and private screenings do not have films associated with them. Hence,**screening** is an ISA hierarchy. We note that the screening types do not overlap and each **screening** should be a specializations.

[12] This is redundant information: we already know how public screenings (normal and special screenings) keep track of films.

[13] Exact description of an entity Film with one primary key attribute fid.

[14] There are two types of ticket orders: one can buy or reserve a ticket for a screening. Either will allocate a specific seat for that screening. We represent both orders via a relationship between screenings and seats. To assure we can distinguish between sales and reservations, we maintain a Boolean attribute paid.

[15] How becomes a standard attribute. Furthermore, we keep track of whether this order was originally a reservation. Note that non-reservations must always be paid (as they are direct sales)!

~~film options)~~.[16,17,18]

Notes on the analysis The analysis already revealed all elements that are relevant for the ER-Diagram. Furthermore, the footnotes discuss the main choices we can make, that we did make, and what open questions are still around. These footnotes will serve as a direct basis for the report (and, in practice, will be starting points with further meetings with the cinema owners to figure out all requirements).

From the analysis, we have seen that the description is not very precise in the following areas:

- room characteristics (especially accessibility options);
- seat locations; and
- the sale infrastructure.

For all these, we have chosen *simple* solutions that can store the mentioned information. In practice, I expect these areas to be the main topics for future iterations on the requirements with the cinema owners.

Model Solution

We have derived an ER-Diagram that can represent all information present in the description. The resultant ER-Diagram can be found in Figure 1. Next, we provide a high-level overview of the ER-Diagram, discuss the choices made, and present all constraints.

Each cinema is represented by an entity **Cinema** with attributes *name* (of type TEXT) and *address* (of type TEXT). Names are unique and can change. As names can change, the *name* be tricky to use as primary keys. Hence, we assigned cinemas numeric automatically-generated identifiers *cid* (of type INTEGER).

Cinemas own rooms that are identified within their cinema by a room number. Hence, we model **Room** as a weak entity with an identifying relationship In_Cinema to **Cinema** and with a partial key *room_number* (of type INTEGER). Rooms have characteristics. From the description it is unclear to what level of details these characteristics need to be stored, but the examples provided for screen type, screen size, projector type, and sound system indicate that these characteristics can be stored in simple attributes of the weak entity **Room** Hence, we add attributes *screen_type* (of type TEXT), *screen_size* (of type INTEGER), *projector_type* (of type TEXT), and *sound_system* (of type TEXT). For the accessibility options we have chosen to add an attribute *access_flag* (of type BOOLEAN) that represents a flag that is set to true when the room is designed with accessibility in mind.

Rooms have seats and seats are uniquely identified within a room by its row and seat number. Hence, we model **Seat** as a weak entity with an identifying relationship In_Room to **Room** and with a partial key *row* (of type INTEGER) and *number* (of type INTEGER). It is unclear whether the row and number of a seat already specify its location. Hence, we have added a placeholder attribute *location* (of type TEXT) to **Seat** to keep track of the per-room coordinate of that seat. Finally, we added an attribute *reserved_flag* (of type BOOLEAN) to **Seat** that indicates whether this seat is reserved for disabled people.

We represent each screening by an entity **Screening**, we introduce an automatically-generated identifier *sid* as no clear identifier is provided in the text, and we represent the timeslot by two attributes *begin_time* (of type TIMESTAMP) and *end_time* (of type TIMESTAMP). We assign each screening to exactly one room via the relationship Scr_Room. There are three types of screenings and there are clear distinctions between them: normal screenings have exactly one film associated to them, special screenings have one-or-more films

[16] The paid price will become a final attribute.

[17] There is an important hidden constraint here: if a screening is in room X, a seat is in room Y, and there exists an order for this combination of screening and seat, then $X = Y$ (they must refer to the same room).

[18] The ticket sale description contains a lot of application and process logic (e.g., reservations are cancelled 45 minutes before the start of the film) and relatively little detail on what *exactly* needs to happen with all the data. E.g., it is unspecified whether the system needs to keep track of all reservations (including those cancelled) separately from the sales. We have chosen for a minimal method that can store all information requested, but not more.

associated with them, and private screenings do not have films associated with them. Hence, we introduce an ISA-hierarchy with root **Screening** and specializations **NormalS**, **PrivateS**, and **SpecialS**. We represent films by an entity **Film** with primary key *fid* (of type INTEGER), we associate exactly-one film to each normal screening via the relationship NS_Shows between **NormalS** and **Film**, and we associate one-or-more films to each special screening via the relationship SS_Shows between **SpecialS** and **Film**.

We have chosen for a minimal method that can store all the described information for sales and reservations: we represent *all* sales and reservations via the relationship Ticket_Order between entities **Screening** and **Seat**. To assure we can distinguish between sales and reservations, Ticket_Order has an attribute *paid_flag* (of type BOOLEAN) that is true when the ticket is already paid (a sale). Furthermore, we keep track of whether this order was originally a reservation via the attribute *reserved_flag* (of type BOOLEAN). Finally, we keep track of how the order was made via attribute *how* (of type TEXT) and what the price of the order is via attribute *price* (of type DECIMAL).

The following constraints are not expressed in the ER-Diagram:

- The attribute *name* of entity **Cinema** is unique.
- In entity **Screening**, values for the attribute *begin_time* must be after values for the attribute *end_time*.
- In relationship Ticket_Order, the attribute *paid_flag* must be true if the attribute *reserved_flag* is false (if a ticket is not reserved, it must be a sale order).
- If an **screening** S_1 is assigned to room X and a **seat** S_2 is in room Y, and there exists a Ticket_Order between S_1 and S_2, then the rooms X and Y must be the same room.
- Timeslots for entities **Screening** that are assigned to the same room should not overlap.
- The screening types do not overlap and each **Screening** must be a specializations.

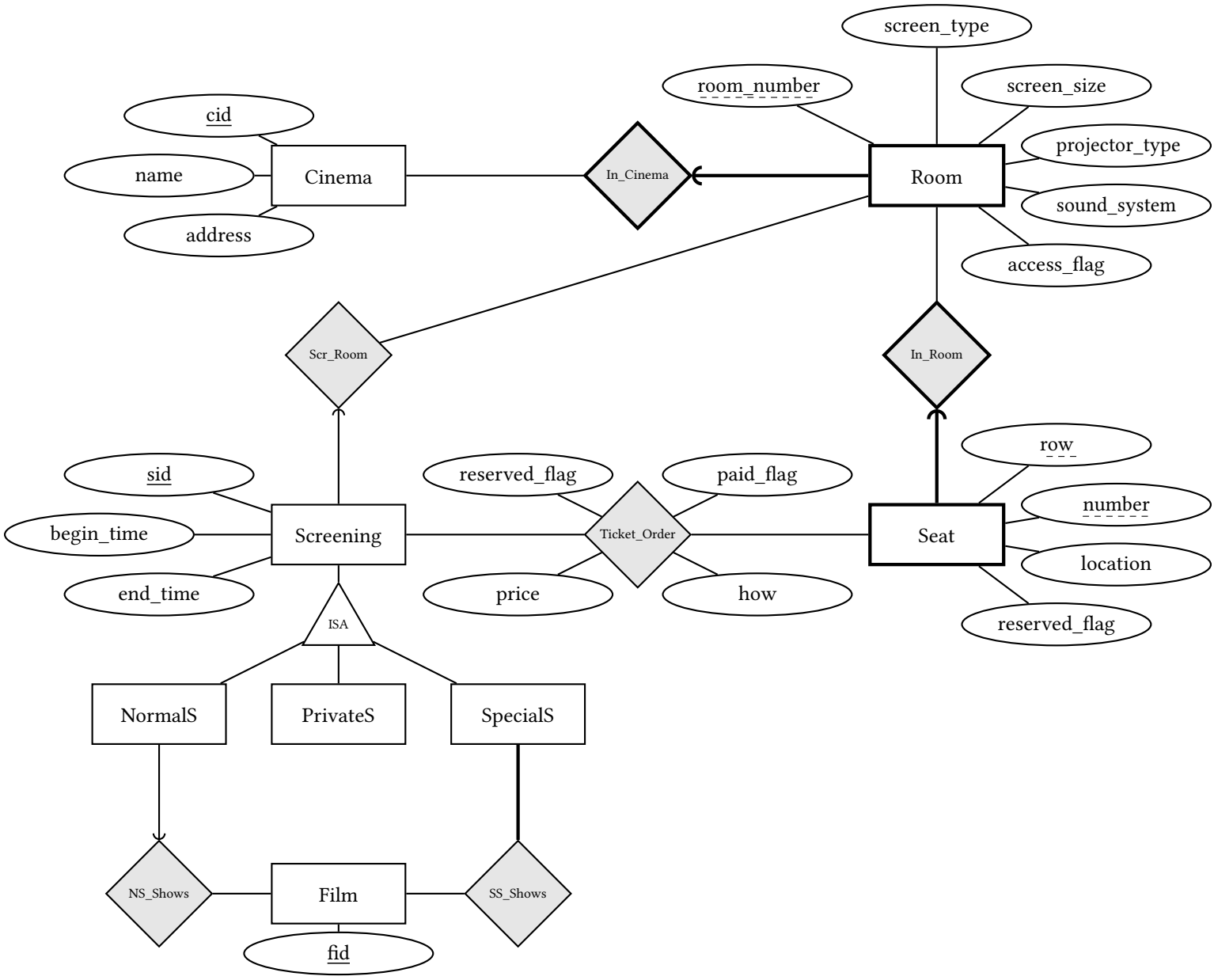

Figure 1: The ER-Diagram for the renewed cinema information system.