Cyril Saidane
Pierre Venereau
Aleks Murauskas
Paul Seimandi

Movie Theater Management System Database

I. Requirements Analysis

Introduction

The goal of this application is to handle individual management of movie theaters that are part of the same company. The database is used to store information which would be useful for the cinemas to keep track of movie reservations, client purchases, screening plannings… The purpose is therefore to have a constantly updated database of all transactions and event times happening across movie theaters.

We have limited the scope of the project to a couple of core entity sets and relationships. We chose our constraints to adapt the project to the deliverable requirements and chose ~ 10 entity sets and almost as many relationships. To fulfill the requirements we chose to therefore focus on the parts we judged most relevant to implement for this database.

Data Requirements

Entities and their attributes:

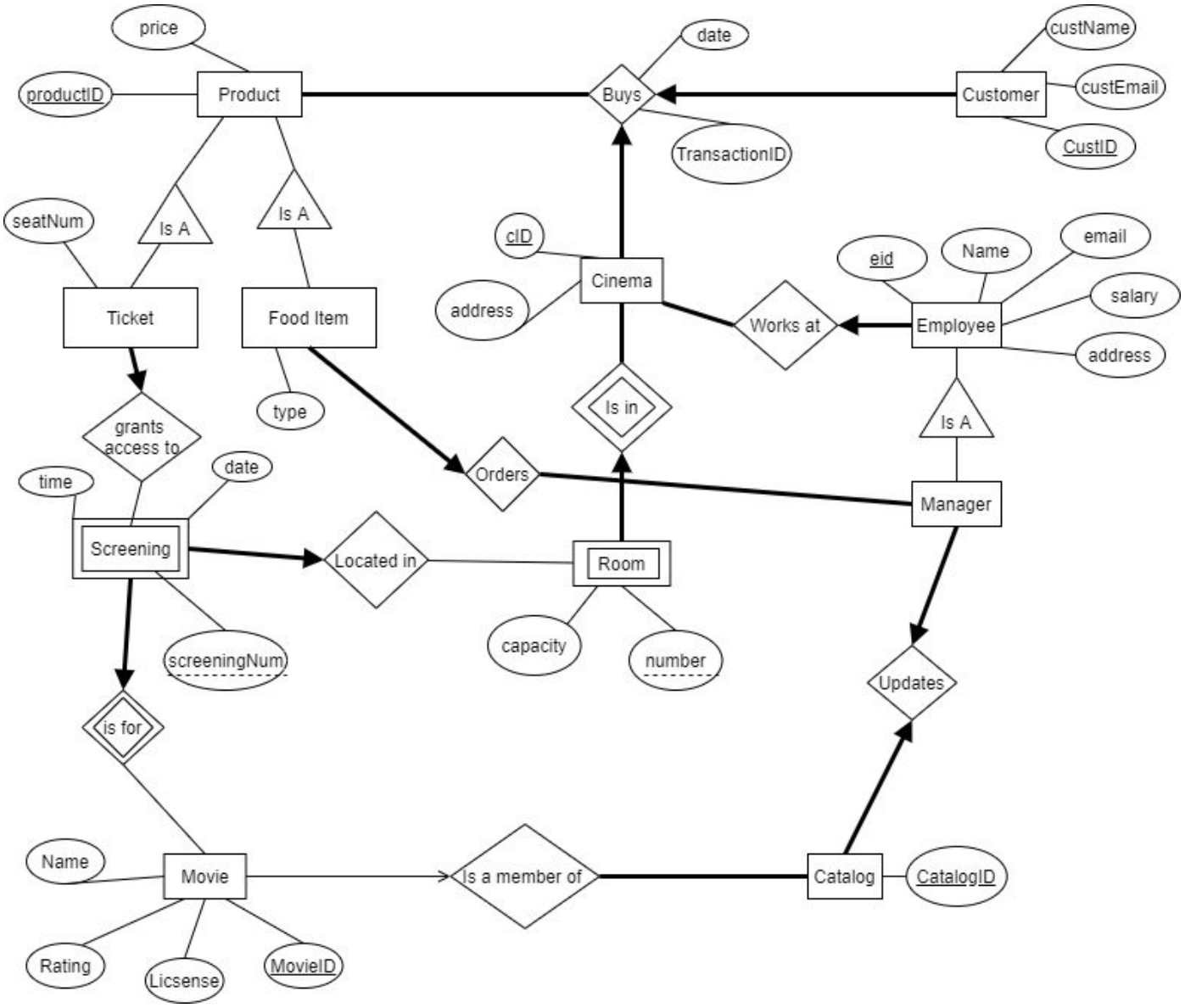
- **Customer**: custId, name, email : *Customers interact with the cinema and with products.*
- **Cinema**: cId, address : *There can be several cinemas all of which have associated rooms and products.*
- **Product**: productId, price : *Cinemas propose products which all have a price.*
- **Movie**: movieId, name, director, year : *This entity set represents movies which are then managed into a movie catalog* .
- **Screening**: screeningNum, time, date : *Screening is a weak entity to the Movie entity and represents times and date at which the movie theater screens particular movies.*
- **Employee**: eid, name, email, salary : *Employees work in the cinemas.*
- **Manager***: This entity set inherits from Employees but has more relationships (for management of the cinema).*
- **Ticket**: seatNum : *This is one type of product which is associated to a screening.*
- **Food**: type : *This is the other product type which customers can buy.*
- **Movie Catalog**: catalogID : *The catalog lists all the movies owned and screened by the movie theater.*

Functional Requirements

Operations/relationships:

- **Buys**: date, transactionId : *This is a ternary relationship that handles the purchase of products by a customer inside a particular cinema. It has attributes to keep track of all transactions.*
- **Works at**: *This relationship associates employees to the cinema in which they work.*
- **Is in**: *This weak relationship is used to associate the weak entity set Room to the cinemas (1 to many rooms per cinema but a room is -only- in one cinema).*
- **Grants access to**: *This describes the relationship between tickets and screenings where a ticket only grants access to a particular screening.*
- **Located in**: *This relationship shows that a screening is located in only one room whilst a room can host 0 to many screenings.*
- **Orders**: *We used this relationship for the manager sub entity set to handle the ordering of food items for the cinema.*
- **Is for**: *This second weak relationship relates screenings to movies because a screening may not exist without its corresponding movie.*
- **Is a member of**: *Movies are part of at most one catalog through this relationship.*
- **Updates**: *This is a one-to-one relationship for which catalogs may only have one manager and managers can only manage one catalog (their cinema's).*

II. E/R Diagram



III. Relational Model

Product(<u>productId</u>, price)
Ticket(<u>productId</u>, seatNum) (productId ref Product)
Food Item(<u>productId</u>) (productId ref Product)
Movie(<u>movieId</u>, name, rating, license)
Screening(<u>movieId</u>, <u>screeningNum</u>, time, date) (movieId ref Movie)
Catalog(<u>catalogId</u>)
Employee(<u>eid</u>, name, email, salary, address)
Manager(<u>eid</u>) (eid ref Employee)
Customer(<u>custId</u>, cust_name, cust_email)
Cinema(<u>cId</u>, address)
Room(<u>cId</u>, <u>number</u>, capacity) (cId ref Cinema)

IV. Creativity / Complexity

Our system enables the handling of sales (ticketing and food) as well as the management of these resources through the manager. To implement these functionalities we had to incorporate ‘is A’ relationships for different types of products for them to inherit from a super entity set with a price attribute. Similarly, we created a sub entity set manager to handle the movie catalog and the food stock.

We also chose to implement two weak entities to handle the miscreation of impossible entities. For example, we chose to have a weak entity set ‘Screening’ which is made of relative entities who only exist when linked to an existing movie from the ‘Movie’ entity set. The same was done for the rooms inside cinemas which are uniquely identified by the corresponding cinema with a room number.

The majority of our system relies on the ternary relationship ‘buys’ which relates a customer to a product and a cinema. It can be seen that key constraints were used to limit this interaction to:

- One cinema
- One customer
- 1 or many products

per transaction.