CINEMA TICKETING MANAGEMENT

LOGICAL DESIGN – PHASE TWO

(COMPUER STREAM - 4TH YEAR)

GROUP-9	ID
DAGMAWI GEBREWOLD	ATR/9848/10
SAMUEL NEGALIGN	ATR/2574/10
HENOK MITIKU	ATR/2159/10
KALEB TESFAYE	ATR/8865/10

SUBMITTED TO: MR TESFAMICHAEL
DATE: 3, MAY, 2021

<u>Relational Model</u>

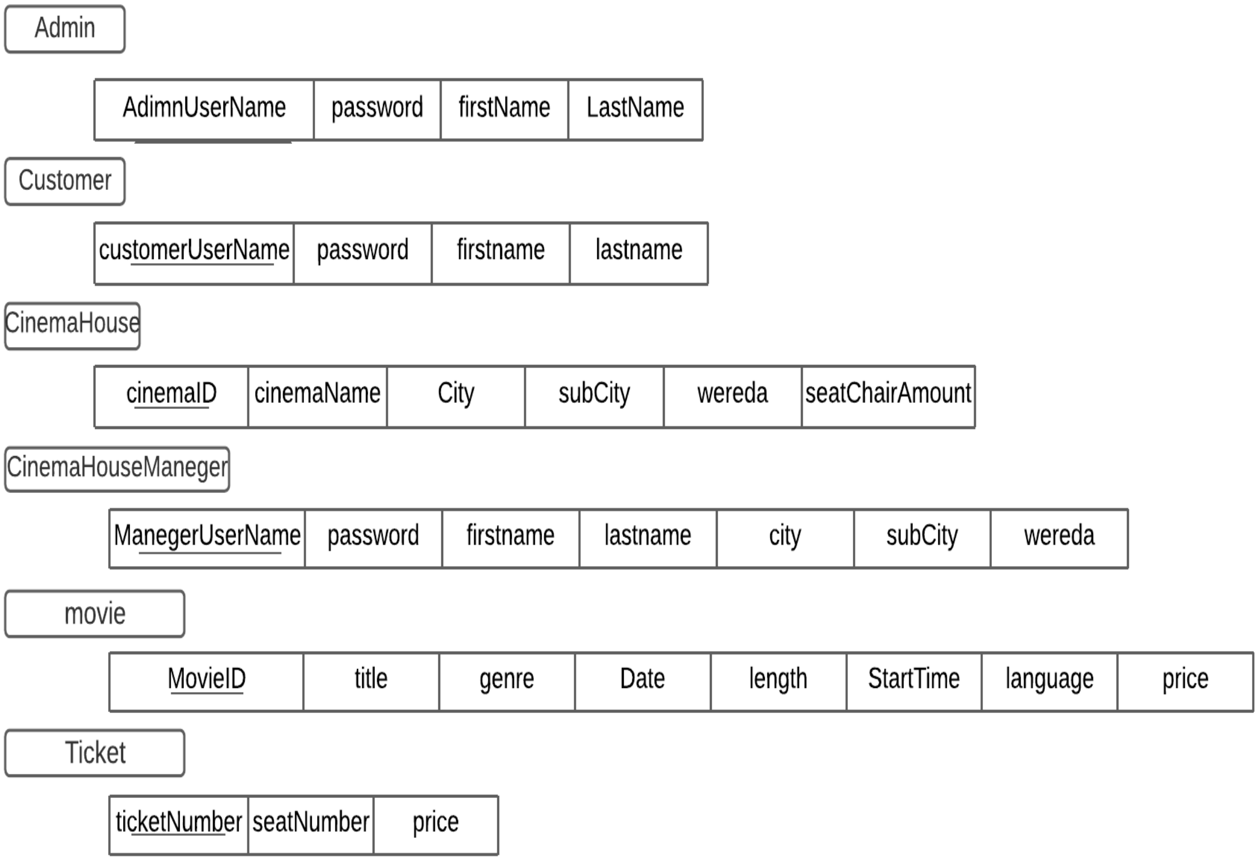
ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into relational model, but an approximate schema can be generated.

There are several processes and algorithms available to convert ER Diagrams into Relational Schema.

Step1 *mapping strong entity*

➔Only constitute simple attribute in case of composite attribute

➔Exclude multivalued attribute from mapping into a table



Admin

AdimnUserName	password	firstName	LastName

Customer

customerUserName	password	firstname	lastname

CinemaHouse

cinemaID	cinemaName	City	subCity	wereda	seatChairAmount

CinemaHouseManeger

ManegerUserName	password	firstname	lastname	city	subCity	wereda

movie

MovieID	title	genre	Date	length	StartTime	language	price

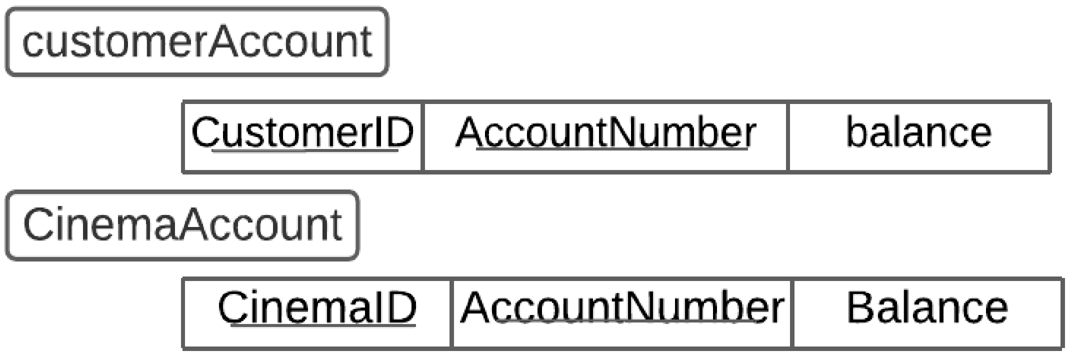
Ticket

ticketNumber	seatNumber	price

Step2 *mapping weak entity*

➔Create separate relation and include all simple attribute

➔Add primary key of the owner entity set into weak entity set



customerAccount

CustomerID	AccountNumber	balance

CinemaAccount

CinemaID	AccountNumber	Balance

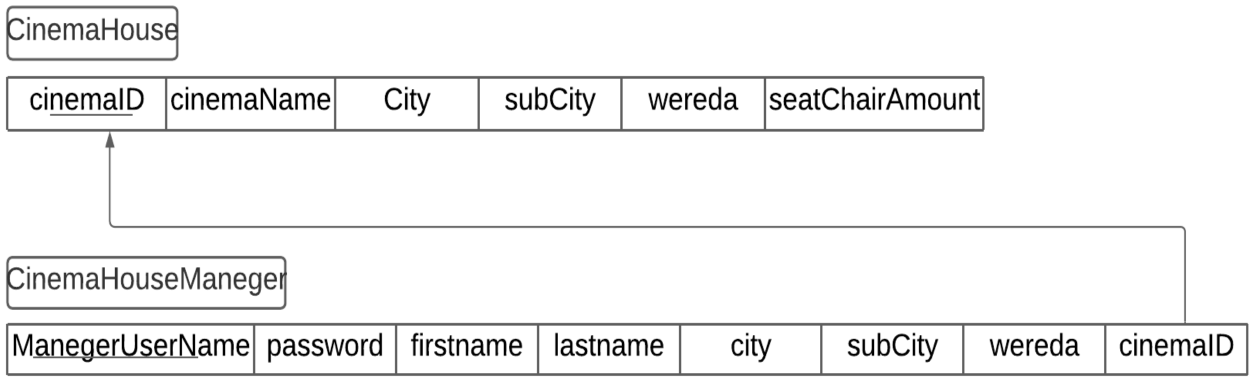
Step 3 *mapping 1:1 relationship type*

<u>We use foreign key approach→</u> identify the entity set with total participation and add a primary key of the other entity

In our ER model we have two 1:1 Relationship

These are cinema house manager manage cinema house and

Ticket saved selected movie

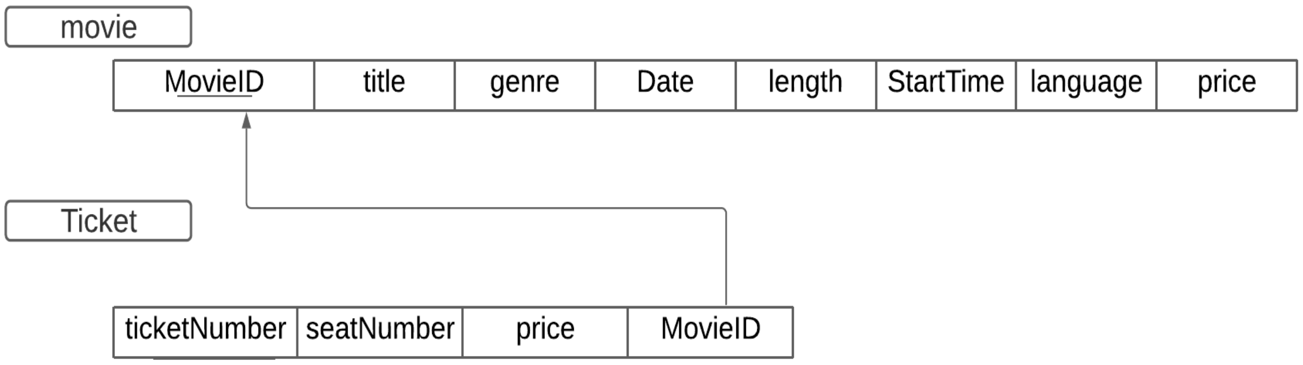


CinemaHouse

cinemaID	cinemaName	City	subCity	wereda	seatChairAmount

CinemaHouseManeger

ManegerUserName	password	firstname	lastname	city	subCity	wereda	cinemaID



movie

MovieID	title	genre	Date	length	StartTime	language	price

Ticket

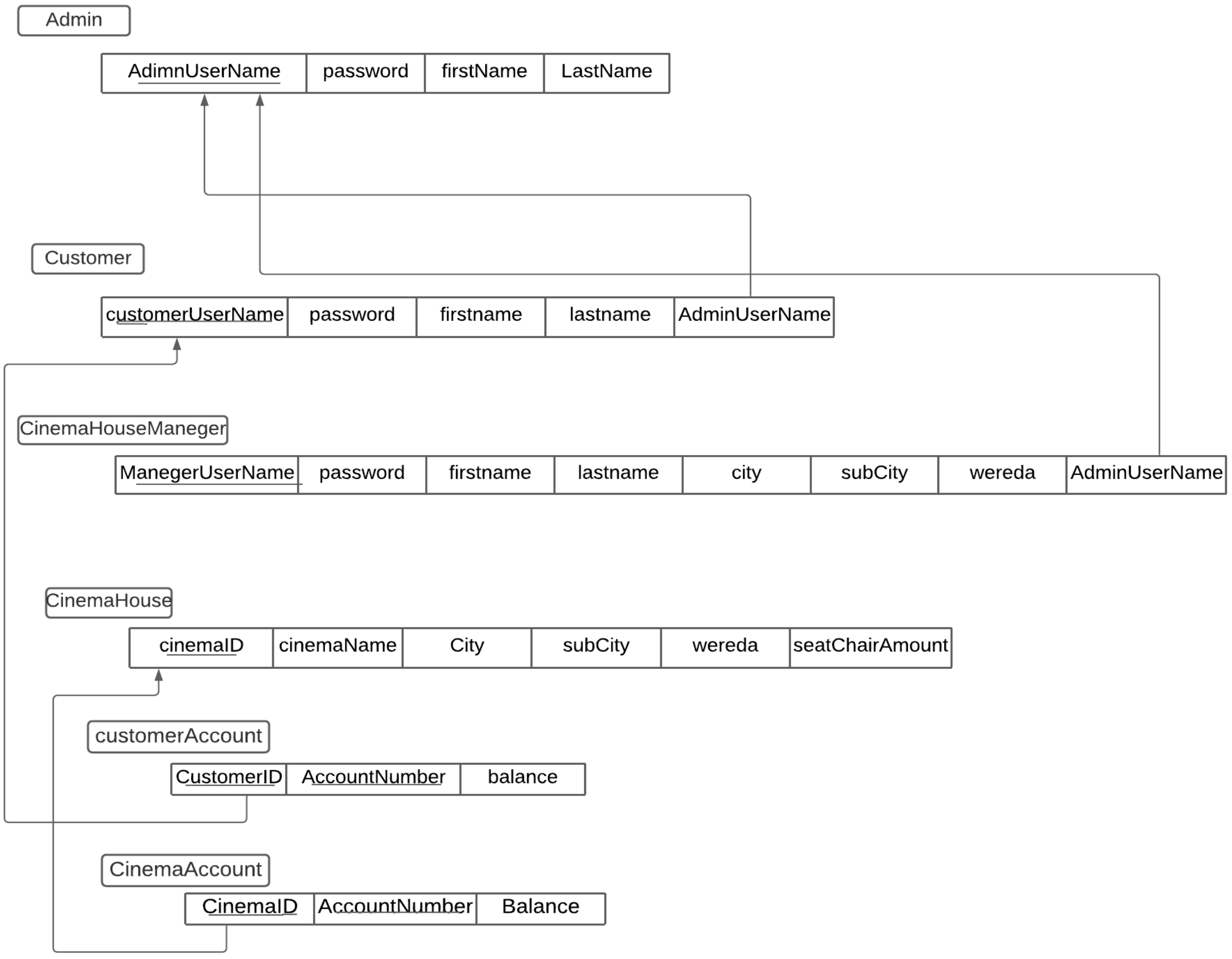
ticketNumber	seatNumber	price	MovieID

Step 4 *mapping 1:N relationships*

Let R and S be the entity set with 1:N where S is having total participation in relationship

Add primary key of R and S as foreign key

In our ER diagram we have four 1:N relationship

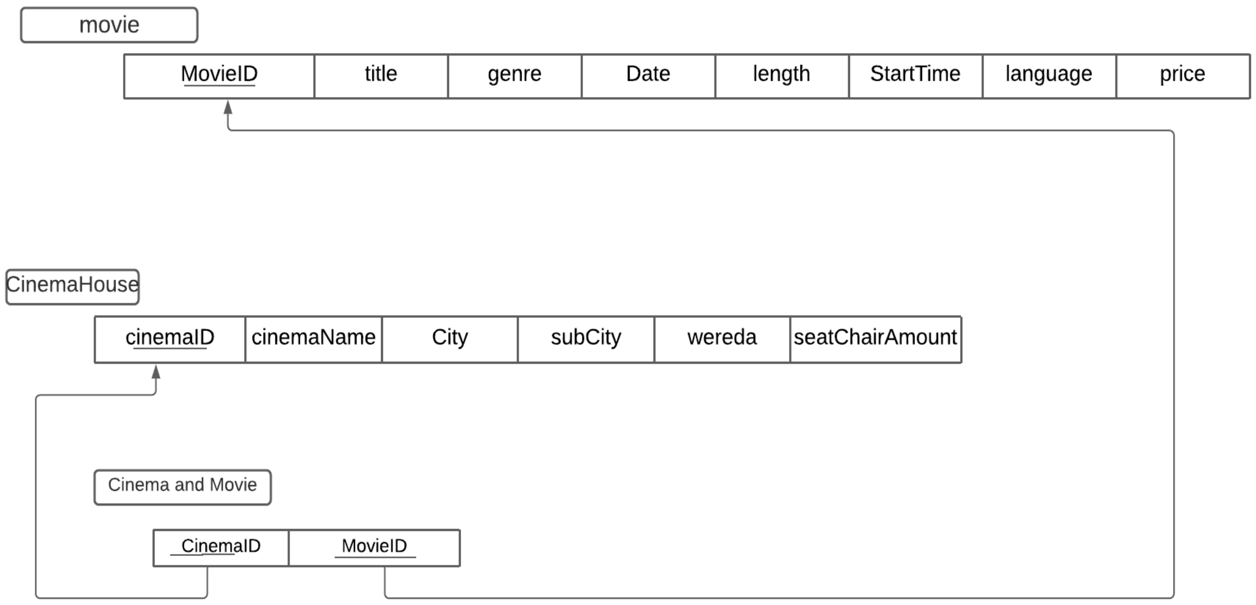


Step 5 *mapping many to many relationships*

Create a third relation containing the primary key of both the entity set and descriptive attribute (if any)

In our ER diagram we have one many to many relationship

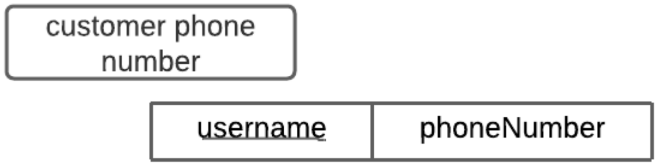
Cinema house present Movie



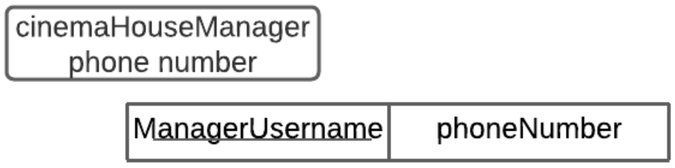
Step 6 *mapping multivalued attribute*

For each multivalued attribute create separate relation and add primary key of the entity set in new relation as a foreign key

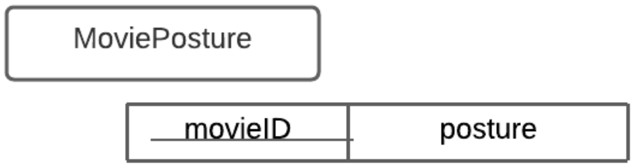
Customer



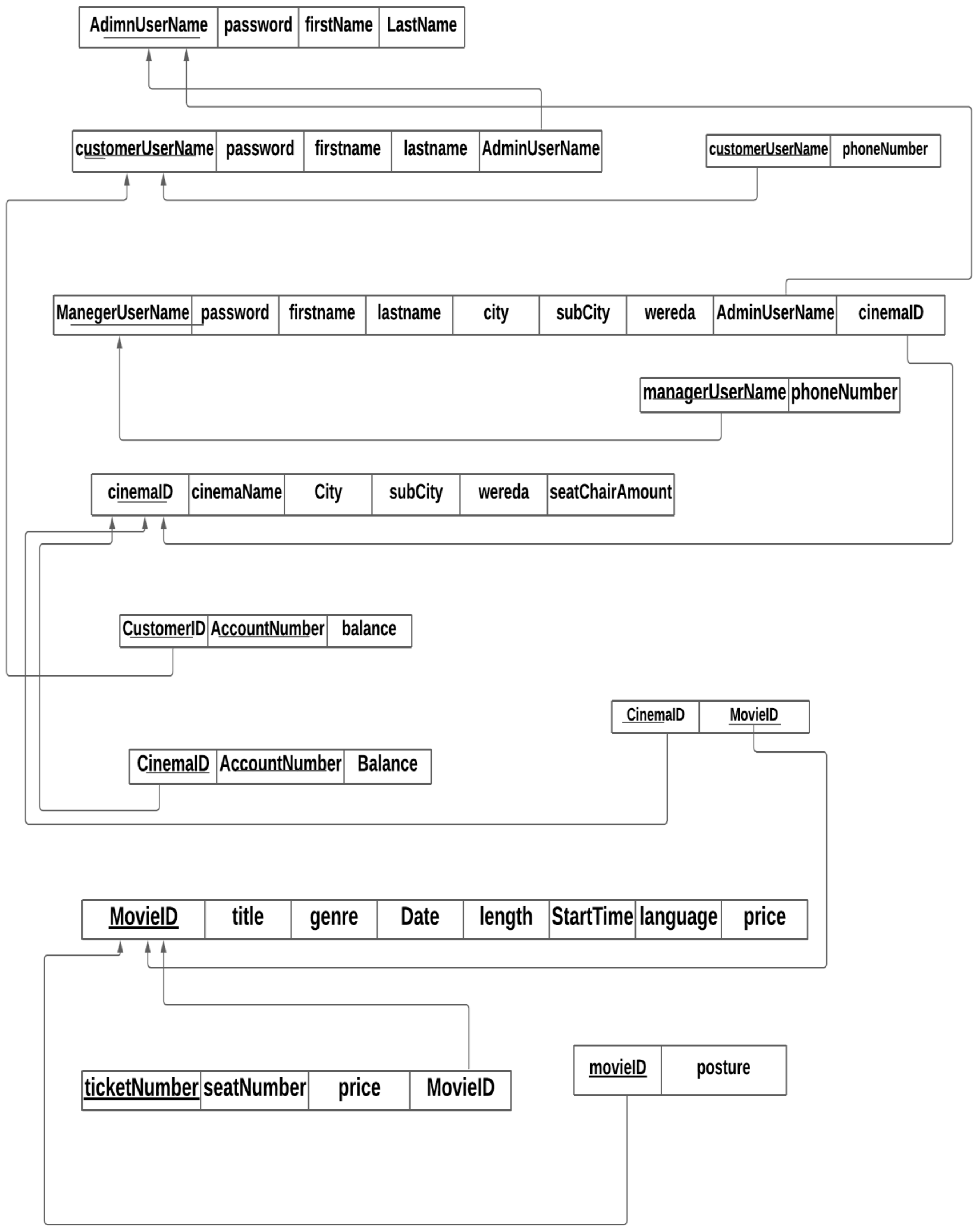
Cinema House Manager



Movie



As we see we finish the steps and we did everything needed to do relational model so on the next page we show the complete relational model


AdimnUserName | password | firstName | LastName
customerUserName | password | firstname | lastname | AdminUserName
customerUserName | phoneNumber
ManegerUserName | password | firstname | lastname | city | subCity | wereda | AdminUserName | cinemaID
managerUserName | phoneNumber
cinemaID | cinemaName | City | subCity | wereda | seatChairAmount
CustomerID | AccountNumber | balance
CinemaID | MovieID
CinemaID | AccountNumber | Balance
MovieID | title | genre | Date | length | StartTime | language | price
ticketNumber | seatNumber | price | MovieID
movieID | posture

Functional Dependencies

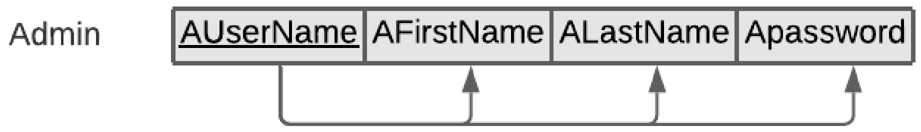
A functional dependency (FD) is a relationship between two attributes, typically between the PK and other non-key attributes within a table. For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y. This relationship is indicated by the representation below:

X ————> Y

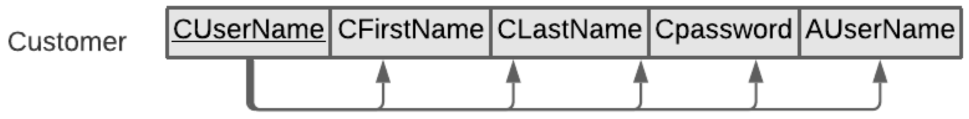
The left side of the above FD diagram is called the *determinant*, and the right side is the *dependent*. Here are a few examples.

From our relational model we distinguish the functional dependencies as follow

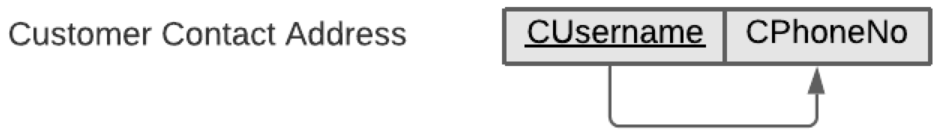
$\{\text{AUserName}\}^+$ = { AUserName, AFirstName, ALastName, APassword} = Admin



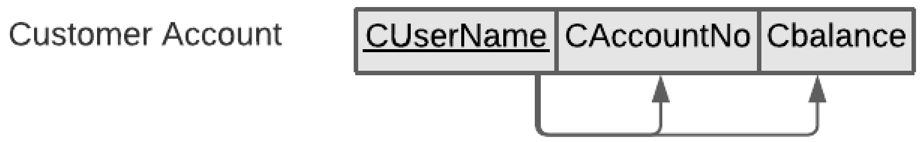
$\{\text{CUserName}\}^+$ = { CUserName, CFirstName, ALastName, CPassword, AUserName} = Customer



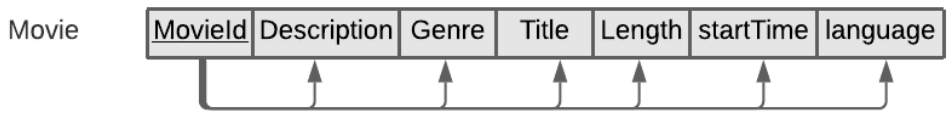
$\{\text{CUserName}\}^+$ = {CUserName, CPhoneNo} = Customer Contact



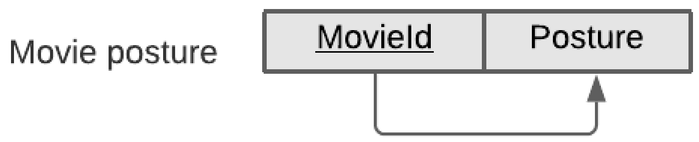
$\{\text{CUserName}\}^+$ = {CUserName, CAccountNo, Cbalance} = Customer Account



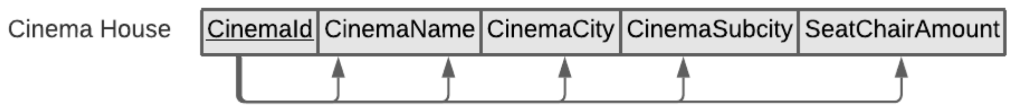
$\{\text{MovieId}\}^+$ = {MovieId, Description, Genre, Title, Length, startTime, language} = Movie



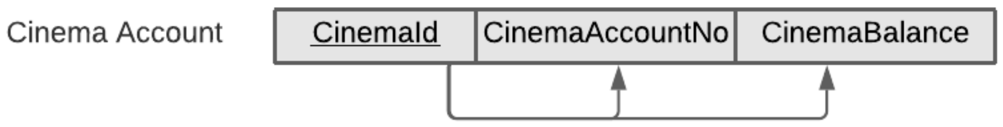
$\{\text{MovieId}\}^+$ = {MovieId, Posture} = Movie Posture



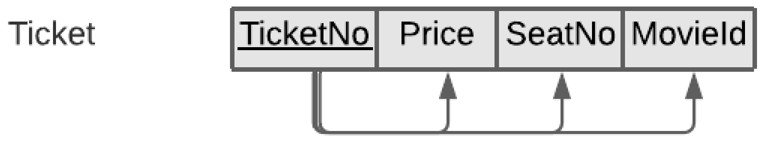
$\{\text{CinemaId}\}^+$ = {CinemaId, CinemaCity, CinemaSubcity, SeatChairAmount} = Cinema House



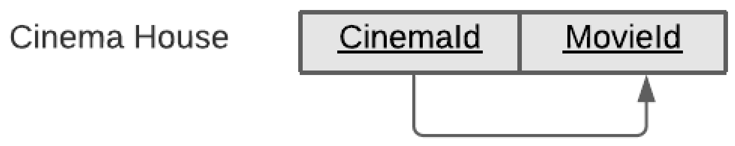
$\{\text{CinemaId}\}^+$ = {CinemaId, CinemaAccountNo, CinemaBalance} = Cinema Account



$\{\text{TicketNo}\}^+$ = {TicketNo, Price, SeatNo, MovieId} = Ticket



$\{\text{CinemaId}\}^+$ = {CinemaId, MovieId} = Cinema House



Normalization

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

- Eliminating redundant(useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

Our entities is already normalized to third normalized form which means in our entities.

Our table has only single (atomic) value attribute because of this our entities satisfy first Normal form.

In our table there is no partial dependencies because in our table we don’t have composite primary key. Because of this we satisfy second normal form.

In our table there is no **transitive dependency (***non-prime attribute depend on other non-prime attribute rather than depending upon the prime attribute***). So our entities are third normalized form**