

Database

Final Assignment

- Movie Hub -

소프트웨어학과

201620976 박현진

E-mail: qokkop@ajou.ac.kr

Github address : <https://github.com/hhhhjjjj96/Database-Project>

목차

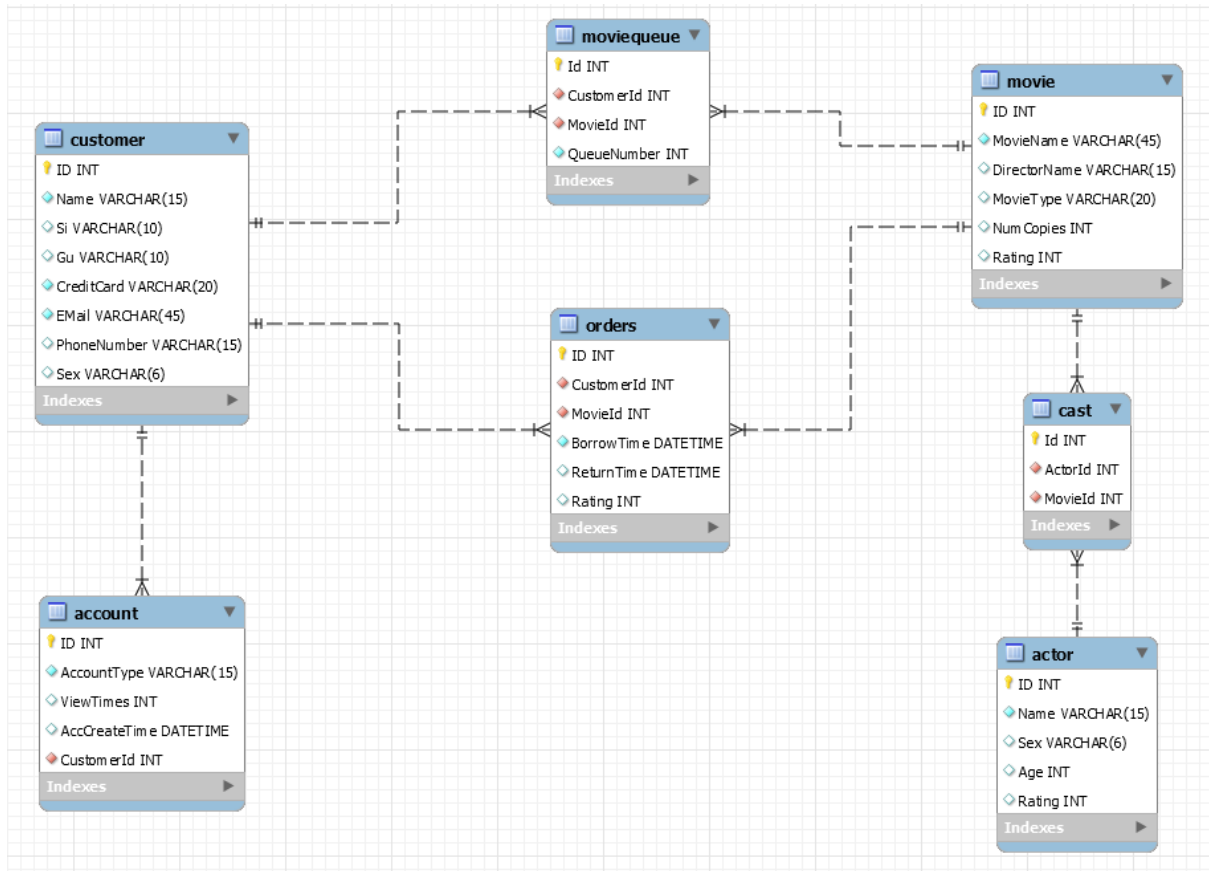
1. E-R Diagram (EER Diagram).....	5
1.1. Customer	5
1.2. Account	5
1.3. Movie Queue.....	5
1.4. Orders.....	5
1.5. Movie.....	6
1.6. Cast.....	6
1.7. Actor.....	6
2. Database Tables	6
2.1. Customer Table	6
2.2. Account Table	8
2.3. Movie Table.....	9
2.4. Actor Table	10
2.5. Orders Table.....	11
2.6. MovieQueue Table	12
2.7. Cast Table	13
3. User-level Transactions	14
3.1. A customer's currently held movies	14
3.2. A customer's queue of movies it would like to see	14
3.3. A customer's account settings	15
3.4. Movies available of a particular type.....	16
3.5. Movies available with a particular keyword or set of keywords in the movie name ..	16
3.6. Movies available starring a particular actor or group of actors	17
3.7. Best-Seller list of movies.....	17
3.8. Rate the movies they have rented	18

그림 목차

[그림 1] EER Diagram	5
[그림 2] Table List	6
[그림 3] Customer Table DDL	6
[그림 4] Demo Data in Customer Table	7
[그림 5] Customer Table	7
[그림 6] Account Table DDL	8
[그림 7] Demo Data in Account Table	8
[그림 8] Account Table	8
[그림 9] Movie Table DDL	9
[그림 10] Demo Data in Movie Table	9
[그림 11] Movie Table	9
[그림 12] Actor Table DDL	10
[그림 13] Demo Data in Actor Table	10
[그림 14] Actor Table	10
[그림 15] Orders Table DDL	11
[그림 16] Demo Data in Orders Table	11
[그림 17] Orders Table	11
[그림 18] MovieQueue Table DDL	12
[그림 19] Demo Data in MovieQueue Table	12
[그림 20] MovieQueue Table	12
[그림 21] Cast Table DDL	13
[그림 22] Demo Data in Cast Table	13
[그림 23] Cast Table	13

[그림 24] A customer's currently held movies Transaction 및 결과물	14
[그림 25] Add a movie to the customer's queue Transaction 및 결과물	14
[그림 26] Show a movie in the queue Transaction 및 결과물	14
[그림 27] Show customer's account settings Transaction 및 결과물	15
[그림 28] Edit customer's account settings Transaction 및 결과물	15
[그림 29] Movies available of a particular type Transaction 및 결과물	16
[그림 30] Movie name keyword Transaction 및 결과물	16
[그림 31] Movie's Actor Transaction 및 결과물	17
[그림 32] Best-Seller list of movies Transaction 및 결과물	17
[그림 33] Rate the movies they have rented Transaction	18
[그림 34] 기존 Anyone 영화의 Rating정보	18
[그림 35] Anyone 영화 반납 후 Rating 하는 예시	18
[그림 36] 변경된 Anyone 영화의 Rating 정보	18

1. E-R Diagram (EER Diagram)



[그림 1] EER Diagram

위의 EER은 MySQL Workbench를 통해 그려진 Diagram이다.

1.1. Customer

Customer는 Customer의 ID, 이름, 주소, 신용카드 정보, 연락처, 성별에 대한 attribute를 갖고 있다.

1.2. Account

Customer는 Movie Hub에 계정을 가지고 있다. 계정에 대한 정보는 Account에 담겨있다. Account에는 ID, 구독 타입, 시청 횟수, 계정 생성일에 대한 attribute를 가진다.

1.3. Movie Queue

Customer는 Movie Queue에 여러 개의 영화를 담을 수 있다. 따라서 N:M의 관계로 주었다.

1.4. Orders

Customer는 한 번에 영화를 2개까지만 볼 수 있다. 따라서 Order를 최대 2개를 할 수 있기 때문

에 Customer와 Order의 관계는 1:2로 설정하였다. - Order에는 주문한 Customer와 주문된 Movie에 대한 정보가 있다. 빌린 시간과 반납한 시간에 대한 정보도 존재한다.

1.5. Movie

Movie에는 ID, 영화 이름, 감독 이름, 영화 장르, 영화의 사본 수, 별점에 대한 attribute가 있다.

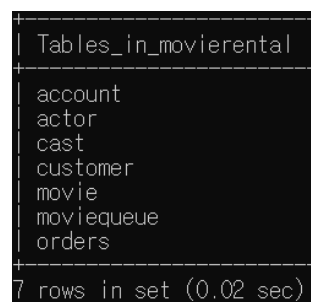
1.6. Cast

Movie에는 Actor들이 등장한다. 하나의 Movie에는 여러 Actor가 존재할 수 있기 때문에 N:M의 관계로 출연진에 대한 설계를 하였다.

1.7. Actor

Actor에는 ID, 배우 이름, 배우 성별, 배우 나이, 배우에 대한 평점이라는 attribute가 있다.

2. Database Tables



Tables_in_movierental
account
actor
cast
customer
movie
moviequeue
orders

7 rows in set (0.02 sec)

[그림 2] Table List

2.1. Customer Table

2.1.1. DDL

```
CREATE TABLE `customer` (  
  `ID` int NOT NULL,  
  `Name` varchar(15) NOT NULL,  
  `Si` varchar(10) DEFAULT NULL,  
  `Gu` varchar(10) DEFAULT NULL,  
  `CreditCard` varchar(20) NOT NULL,  
  `EMail` varchar(45) NOT NULL,  
  `PhoneNumber` varchar(15) DEFAULT NULL,  
  `Sex` varchar(6) DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

[그림 3] Customer Table DDL

ID를 Primary Key로 두었다. 이름은 영어로 작성되기 때문에 Name의 글자수를 15자까지 적을 수 있게 하였다. 서비스에서 회원가입 및 로그인을 하기 위해서는 이름과 이메일이 필요하기 때문에

Name과 Email의 경우 Not Null로 설정하였다. 주소와 연락처, 성별, 신용카드 정보는 varchar의 형식을 가지도록 하였고 default 값을 NULL로 하였다.

2.1.2. DML about Demo Data

```
insert into customer values(111111, 'HyeonJin', 'Seoul', 'Mapo', '1111-1111-1111-1111', 'a@a.a', '010-1111-1111', 'M');
```

```
insert into customer values(222222, 'HyunJun', 'Anyang', 'Dongan', '2222-2222-2222-2222', 'b@b.b', '010-2222-2222', 'M');
```

```
insert into customer values(333333, 'EunJi', 'Seoul', 'Mapo', '3333-3333-3333-3333', 'c@c.c', '010-3333-3333', 'F');
```

```
insert into customer values(444444, 'HyeJi', 'Suwon', 'Yeongtong', '4444-4444-4444-4444', 'd@d.d', '010-4444-4444', 'F');
```

	ID	Name	Si	Gu	CreditCard	EEmail	PhoneNumber	Sex
	111111	HyeonJin	seoul	mapo	1111-1111-1111-1111	a@a.a	010-1541-1541	M
	222222	HyunJun	Anyang	Dongan	2222-2222-2222-2222	b@b.b	010-2222-2222	M
	333333	EunJi	Seoul	Mapo	3333-3333-3333-3333	c@c.c	010-3333-3333	F
	444444	HyeJi	Suwon	Yeongtong	4444-4444-4444-4444	d@d.d	010-4444-4444	F

[그림 4] Demo Data in Customer Table

2.1.3. Table Information

	Field	Type	Null	Key	Default	Extra
▶	ID	int	NO	PRI	NULL	
	Name	varchar(15)	NO		NULL	
	Si	varchar(10)	YES		NULL	
	Gu	varchar(10)	YES		NULL	
	CreditCard	varchar(20)	NO		NULL	
	EEmail	varchar(45)	NO		NULL	
	PhoneNumber	varchar(15)	YES		NULL	
	Sex	varchar(6)	YES		NULL	

[그림 5] Customer Table

2.2. Account Table

2.2.1. DDL

```
CREATE TABLE `account` (  
  `ID` int NOT NULL,  
  `AccountType` varchar(15) NOT NULL,  
  `ViewTimes` int DEFAULT NULL,  
  `AccCreateTime` datetime DEFAULT NULL,  
  `CustomerId` int NOT NULL,  
  PRIMARY KEY (`ID`),  
  KEY `Account_CID` (`CustomerId`),  
  CONSTRAINT `Account_CID` FOREIGN KEY (`CustomerId`) REFERENCES `customer` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

[그림 6] Account Table DDL

ID를 Primary Key로 두었고, Customer ID를 Customer Table의 ID를 Reference 하여 Foreign Key로 사용하였다. Account Type은 Limit / Unlimit이 있기에 15글자의 varchar로 설정하였다. ViewTimes는 시청 횟수로 int로 설정하였다. Account Create Time은 datetime으로 설정하였다.

2.2.2. DML about Demo Data

```
insert into account values(1,'unlimit',1,20200101,111111);
```

```
insert into account values(2,'limit',1,20200202,222222);
```

```
insert into account values(3,'unlimit',1,20200303,333333);
```

```
insert into account values(4,'limit',1,20200404,444444);
```

	ID	AccountType	ViewTimes	AccCreateTime	CustomerId
▶	1	unlimit	1	2020-01-01 00:00:00	111111
	2	unlimit	1	2020-02-02 00:00:00	222222
	3	unlimit	1	2020-03-03 00:00:00	333333
	4	limit	1	2020-04-04 00:00:00	444444

[그림 7] Demo Data in Account Table

2.2.3. Table Information

	Field	Type	Null	Key	Default	Extra
▶	ID	int	NO	PRI	NULL	
	AccountType	varchar(15)	NO		NULL	
	ViewTimes	int	YES		NULL	
	AccCreateTime	datetime	YES		NULL	
	CustomerId	int	NO	MUL	NULL	

[그림 8] Account Table

2.3. Movie Table

2.3.1. DDL

```
CREATE TABLE `movie` (  
  `ID` int NOT NULL,  
  `MovieName` varchar(45) NOT NULL,  
  `DirectorName` varchar(15) DEFAULT NULL,  
  `MovieType` varchar(20) DEFAULT NULL,  
  `NumCopies` int DEFAULT NULL,  
  `Rating` int DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

[그림 9] Movie Table DDL

ID를 Primary Key로 두었다. 영화 제목은 길이가 길 수 있기에 최대 길이로 하였고 Not Null 속성으로 두었다. 감독의 이름, 영화 장르는 varchar로 설정하였다. Number of Copies와 Rating는 int 타입으로 설정하였다.

2.3.2. DML about Demo Data

```
insert into movie values(1111,'Anyone','James','Romance',2,5);
```

```
insert into movie values(2222,'Anything','Sin','Comedy',2,3);
```

```
insert into movie values(3333,'Awesome','Ben','Action',1,3);
```

```
insert into movie values(4444,'Good','Ben','Action',4,4);
```

```
insert into movie values(5555,'Wow','Lee','Action',1,2);
```

	ID	MovieName	DirectorName	MovieType	NumCopies	Rating
▶	1111	Anyone	James	Romance	2	5
	2222	Anything	Sin	Comedy	2	3
	3333	Awesome	Ben	Action	1	3
	4444	Good	John	Comedy	4	4
	5555	Wow	Lee	Action	1	2

[그림 10] Demo Data in Movie Table

2.3.3. Table Information

	Field	Type	Null	Key	Default	Extra
▶	ID	int	NO	PRI	NULL	
	MovieName	varchar(45)	NO		NULL	
	DirectorName	varchar(15)	YES		NULL	
	MovieType	varchar(20)	YES		NULL	
	NumCopies	int	YES		NULL	
	Rating	int	YES		NULL	

[그림 11] Movie Table

2.4. Actor Table

2.4.1. DDL

```
CREATE TABLE `actor` (  
  `ID` int NOT NULL,  
  `Name` varchar(15) NOT NULL,  
  `Sex` varchar(6) DEFAULT NULL,  
  `Age` int DEFAULT NULL,  
  `Rating` int DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

[그림 12] Actor Table DDL

ID를 Primary Key로 두었다. 이름은 영어로 작성되기 때문에 Name의 글자수를 15자까지 적을 수 있게 하였다. 이름은 있어야 하기 때문에 Not Null로 설정하였다. 성별은 Male/Female이 가능하므로 6글자의 varchar로 설정하였다. Age와 Rating은 int로 설정하였다.

2.4.2. DML about Demo Data

```
insert into actor values(11111, 'John', 'Doe', 'M', 11, 4);
```

```
insert into actor values(22222, 'Jane', 'Doe', 'F', 22, 5);
```

```
insert into actor values(33333, 'Jenifer', 'Roe', 'F', 33, 3);
```

```
insert into actor values(44444, 'Richard', 'Roe', 'M', 44, 4);
```

	ID	Name	Sex	Age	Rating
▶	11111	John	M	11	4
	22222	Jane	F	22	5
	33333	Jenifer	F	33	3
	44444	Richard	M	44	4

[그림 13] Demo Data in Actor Table

2.4.3. Table Information

	Field	Type	Null	Key	Default	Extra
▶	ID	int	NO	PRI	NULL	
	Name	varchar(15)	NO		NULL	
	Sex	varchar(6)	YES		NULL	
	Age	int	YES		NULL	
	Rating	int	YES		NULL	

[그림 14] Actor Table

2.5. Orders Table

2.5.1. DDL

```
CREATE TABLE `orders` (  
  `ID` int NOT NULL,  
  `CustomerId` int NOT NULL,  
  `MovieId` int NOT NULL,  
  `BorrowTime` datetime NOT NULL,  
  `ReturnTime` datetime DEFAULT NULL,  
  `Rating` int DEFAULT NULL,  
  PRIMARY KEY (`ID`),  
  KEY `Order_CID` (`CustomerId`),  
  KEY `Order_MID` (`MovieId`),  
  CONSTRAINT `Order_CID` FOREIGN KEY (`CustomerId`) REFERENCES `customer` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE,  
  CONSTRAINT `Order_MID` FOREIGN KEY (`MovieId`) REFERENCES `movie` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

[그림 15] Orders Table DDL

ID를 Primary Key로 사용하였고, Customer의 ID와 Movie의 ID를 각각의 Table에서 Foreign Key로 가져온다. 빌린 시간과 반납 시간을 datetime으로 설정하고, 주문을 받으면 빌린 시간이 생기므로 BorrowTime은 Not Null로 설정하였다.

2.5.2. DML about Demo Data

ID	CustomerId	MovieId	BorrowTime	ReturnTime
11	111111	1111	2020-11-11 00:00:00	2020-11-14 00:00:00
22	222222	2222	2020-12-12 02:00:00	NULL

[그림 16] Demo Data in Orders Table

```
insert into orders values(11, 111111, 1111, 20201111, 20201114, 3);
```

```
insert into orders(id, customerid, movieid, borrowtime) values(22, 222222, 2222, 20201212020000);
```

2.5.3. Table Information

	Field	Type	Null	Key	Default	Extra
▶	ID	int	NO	PRI	NULL	
	CustomerId	int	NO	MUL	NULL	
	MovieId	int	NO	MUL	NULL	
	BorrowTime	datetime	NO		NULL	
	ReturnTime	datetime	YES		NULL	
	Rating	int	YES		NULL	

[그림 17] Orders Table

2.6. MovieQueue Table

2.6.1. DDL

```
CREATE TABLE `moviequeue` (  
  `Id` int NOT NULL AUTO_INCREMENT,  
  `CustomerId` int NOT NULL,  
  `MovieId` int NOT NULL,  
  `QueueNumber` int NOT NULL,  
  PRIMARY KEY (`Id`),  
  KEY `Queue_CID` (`CustomerId`),  
  KEY `Queue_MID` (`MovieId`),  
  CONSTRAINT `Queue_CID` FOREIGN KEY (`CustomerId`) REFERENCES `customer` (`ID`),  
  CONSTRAINT `Queue_MID` FOREIGN KEY (`MovieId`) REFERENCES `movie` (`ID`)  
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

[그림 18] MovieQueue Table DDL

Customer의 ID와 Movie의 ID를 각각의 Table에서 Foreign Key로 가져온다. 그리고 그 두개의 ID를 Primary Key로 사용하였다. Queue의 순서를 나타내기 위해 Queue Number 속성을 추가하였다.

2.6.2. DML about Demo Data

```
insert into moviequeue values(111111,2222,1);
```

```
insert into moviequeue values(111111,1111,2);
```

```
insert into moviequeue values(111111,3333,3);
```

```
insert into moviequeue values(222222,3333,1);
```

	Id	CustomerId	MovieId	QueueNumber
▶	1	111111	2222	1
	2	111111	1111	2
	3	111111	3333	3
	4	222222	3333	1

[그림 19] Demo Data in MovieQueue Table

2.6.3. Table Information

	Field	Type	Null	Key	Default	Extra
▶	Id	int	NO	PRI	NULL	auto_increment
	CustomerId	int	NO	MUL	NULL	
	MovieId	int	NO	MUL	NULL	
	QueueNumber	int	NO		NULL	

[그림 20] MovieQueue Table

2.7. Cast Table

2.7.1. DDL

```
CREATE TABLE `cast` (  
  `Id` int NOT NULL AUTO_INCREMENT,  
  `ActorId` int NOT NULL,  
  `MovieId` int NOT NULL,  
  PRIMARY KEY (`Id`),  
  KEY `Cast_AID` (`ActorId`),  
  KEY `Cast_MID` (`MovieId`),  
  CONSTRAINT `Cast_AID` FOREIGN KEY (`ActorId`) REFERENCES `actor` (`ID`),  
  CONSTRAINT `Cast_MID` FOREIGN KEY (`MovieId`) REFERENCES `movie` (`ID`)  
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

[그림 21] Cast Table DDL

2.7.2. DML about Demo Data

	Id	ActorId	MovieId
▶	1	11111	1111
	2	11111	2222
	3	22222	2222
	4	44444	3333
	5	33333	4444
	6	44444	5555

[그림 22] Demo Data in Cast Table

```
insert into cast values(11111,1111);
```

```
insert into cast values(11111,2222);
```

```
insert into cast values(22222,2222);
```

```
insert into cast values(44444,3333);
```

```
insert into cast values(33333,4444);
```

```
insert into cast values(44444,5555);
```

2.7.3. Table Information

	Field	Type	Null	Key	Default	Extra
▶	Id	int	NO	PRI	<small>NULL</small>	auto_increment
	ActorId	int	NO	MUL	<small>NULL</small>	
	MovieId	int	NO	MUL	<small>NULL</small>	

[그림 23] Cast Table

3. User-level Transactions

3.1. A customer's currently held movies

```
start transaction;  
SELECT * FROM movierental.movie;  
commit;
```

	ID	MovieName	DirectorName	MovieType	NumCopies	Rating
▶	1111	Anyone	James	Romance	2	5
	2222	Anything	Sin	Comedy	2	3
	3333	Awesome	Ben	Action	1	3
	4444	Good	John	Comedy	4	4
	5555	Wow	Lee	Action	1	2

[그림 24] A customer's currently held movies Transaction 및 결과물

이 Transaction은 현재 상영중인 영화를 볼 수 있는 Transaction이다. 현재 상영중인 영화는 Movie Table에 있는 영화들이므로 해당 Table의 Data들을 가져온다.

3.2. A customer's queue of movies it would like to see

3.2.1. Add a movie to the customer's queue Transaction

```
start transaction;  
insert into movierental.moviequeue  
values("?(cid)","?(mid)","?(qn)");  
commit;
```

✓ 9 15:55:15 insert into movierental.moviequeue values(111111,2222,1)

[그림 25] Add a movie to the customer's queue Transaction 및 결과물

3.2.2. Show a movie in the queue Transaction

```
start transaction;  
select * from movierental.moviequeue where CustomerId = "?(cid)";  
commit;
```

CustomerId	MovieId	QueueNumber
111111	2222	1

[그림 26] Show a movie in the queue Transaction 및 결과물

Customer의 큐에 넣을 때, cid, mid, qn을 넣는데, 이는 장고의 current_user.id의 기능을 사용하여

cid의 값을 설정하고, 선택한 영화에 대한 id를 가져와 mid를 설정하고, 해당 Customer의 큐에 들어 있는 영화의 수를 파악하고 새로 넣을 경우 큐의 번호를 파악하여 추가해줄 것이다.

3.3. A customer's account settings

3.3.1. Show customer's account settings Transaction

```
start transaction;
select Si, Gu, CreditCard, EMail, PhoneNumber from movierental.customer where id = "(cid)";
select AccountType, AccCreateTime from movierental.account where CustomerId = "(cid)";
commit;
```

Si	Gu	CreditCard	EMail	PhoneNumber
Seoul	Mapo	1111-1111-1111-1111	a@a.a	010-1111-1111

[그림 27] Show customer's account settings Transaction 및 결과물

3.3.2. Edit customer's account settings Transaction

```
start transaction;
update movierental.customer
set si = "(si change)", gu = "(gu change)",
    creditcard = "(creditcard change)", email="(email change)",
    phonenumber = "(phonenumber change)"
where id = "(cid)";
update movierental.account
set accounttype = "(accounttype change)"
where customerid = "(cid)";
commit;
```

25 16:48:48 update movierental.account set accounttype = "unlimit" where customerid = 222222

ID	AccountType	ViewTimes	AccCreateTime	CustomerId
2	unlimit	1	2020-02-02 00:00:00	222222

[그림 28] Edit customer's account settings Transaction 및 결과물

이 Transaction은 Customer가 자신의 정보를 확인할 수 있고, 정보를 수정하는 Transaction이다. 자신의 Account Type도 바꿀 수 있고, 자신의 이메일, 주소, 전화번호, 카드 정보 또한 바꿀 수 있도록 Parameter를 설정하였다

3.4. Movies available of a particular type

```
start transaction;
select moviename, movietype, rating
from movierental.movie
where MovieType = "%(search movie type)";
commit;
```

✓ 29 17:11:04 select moviename, movietype from movierental.movie where MovieType = "romance" LIMIT 0, 1000

moviename	movietype	rating
Anyone	Romance	4

[그림 29] Movies available of a particular type Transaction 및 결과물

이 Transaction은 현재 상영중인 영화들 가운데 Customer가 원하는 Type의 영화를 검색하는 Transaction이다. Movie Type에 대한 Parameter를 통해 검색 가능하다. 영화 이름과 장르, 평점을 보여준다.

3.5. Movies available with a particular keyword or set of keywords in the movie name

```
start transaction;
select moviename, movietype, rating
from movierental.movie
where MovieName like "%?(search movie keyword)%";
commit;
```

moviename	movietype	rating
Anyone	Romance	4
Anything	Comedy	5

[그림 30] Movie name keyword Transaction 및 결과물

이 Transaction은 영화의 이름에 들어있는 키워드를 통해 영화를 검색 가능하게 해주는 Transaction이다. Like문을 사용하여 검색하였고, "%?(search movie name)%"의 parameter를 주었다.

3.6. Movies available starring a particular actor or group of actors

```
start transaction;
select moviename, actor.firstname, actor.lastname
from movierental.movie, movierental.actor, movierental.cast
where movie.id = "(mid)" AND movie.id = cast.movieid AND cast.actorid = actor.id;
commit;
```

moviename	firstname	lastname
Anything	John	Doe
Anything	Jane	Doe

[그림 31] Movie's Actor Transaction 및 결과물

이 Transaction은 영화에 출연한 배우에 대한 정보를 보여주는 Transaction이다. Movie Table과 Actor Table, Cast Table의 Data들을 비교하여 해당 영화에 출연한 배우에 대한 정보를 검색하도록 할 수 있다.

3.7. Best-Seller list of movies

```
start transaction;
select NumCopies, moviename, MovieType, Rating
from movierental.movie
order by NumCopies desc;
commit;
```

NumCopies	moviename	MovieType	Rating
2	Anything	Comedy	5
1	Anyone	Romance	4
1	Awesome	Action	3

[그림 32] Best-Seller list of movies Transaction 및 결과물

Number of Copies의 수는 영화가 Copy된 횟수를 나타내므로 Movie Table에 있는 NumCopies의 값을 내림차순으로 나열하여 가장 장 팔린 영화의 순서대로 보여준다. Movie Table의 정보를 보여주면 되므로 따로 input parameter가 필요하지 않다.

3.8. Rate the movies they have rented

```
start transaction;
update movierental.orders
set rating = "?(rate)"
where customerid = "?(cid)" AND
      movieid = "?(mid)" AND
      returntime is not null;
update movierental.movie
set rating =
  ((select rating from (select rating from movierental.movie where id="?(mid)") as tmp) + "?(rate)")
  div ((select count(*) from movierental.orders where movieid = "?(mid)" + 1)
where id = "?(mid)";
commit;
```

[그림 33] Rate the movies they have rented Transaction

ID	MovieName	DirectorFirstName	DirectorLastName	MovieType	NumCopies	Rating
1111	Anyone	James	Smith	Romance	1	4

[그림 34] 기존 Anyone 영화의 Rating정보

```
update movierental.movie
set rating =
  ((select rating from (select rating from movierental.movie where id=1111) as tmp) + 2)
  div ((select count(*) from movierental.orders where movieid = 1111) + 1)
where id = 1111;
```

[그림 35] Anyone 영화 반납 후 Rating 하는 예시

ID	MovieName	DirectorFirstName	DirectorLastName	MovieType	NumCopies	Rating
1111	Anyone	James	Smith	Romance	1	3

[그림 36] 변경된 Anyone 영화의 Rating 정보

기존의 Movie Table의 Rating 정보를 가져와 Customer가 반납을 하면서 준 평점을 평균을 내서 계산하였다.

((기존 Rating) + (새로운 Rating)) / ((order 테이블에서 해당 영화가 order된 횟수) + (기존 영화에 평점에 대한 횟수인 1))의 계산 식을 통해 계산하였다.