**Database Design Project**

**Home Video Rental**

**Course:** CS6360.002

**Submitted by:** Xin Yang xxy120130

Feiyue Wu fxw160230

Qiaowei Li qxl161930

**1.Introduction**

The purpose of this project is to implement the software requirements of the Home Video Rental System. The system will facilitate a database system that a video rental company can use to manage the rentals of videos and video inventory. The purpose of this report is to provide a summary of data requirements, database design, and application guide for the PHP web service of Home Video Rental System (HVRS), such as the Create, Read, Update and Delete operations by users.

A brief description diagram of this system is show in Figure 1. Customers (users) can log into the web server through internet and rent videos. All information about videos, transactions, customers are kept and maintained in a database server by SQL.

**Home Video Rental**

**User**

**User**

**PHP**

**Application server**

**SQL**

**Database server**

Internet

**User**

Figure 1. High level diagram of HVRS.

**2.Data Requirements**

For this HVRS, video is the core entity with ID, title, description, price, format and so on.

Each video is assigned an ID to be track of storage, transaction information.

Videos are stored in various stores which is identified by its ID. Each store has an address and some contact information. And every store has one and only one manager. Each manger has his ID, name and contact information

Each video is acted by some actors, and one actor can act in more than one videos. Actor information is necessary since the customer may need it to find the video they like.

Customer is the second core entity in this HVRS. Each customer is assigned one ID with their name, address, contact information. Each customer can have one account, but some new customers may not register for an account. Account entity keeps the information of credit score and discount of the only one related customer. One account can use multiple payment method, such as cash, credit card or PayPal.

Customer is connected to video through rental\_order. Every rental\_order is ordered by one customer, but one customer can have multiple orders. In one order, multiple video can be rented, and one video can appear in many orders. Every order has its own ID and date information to track the order status.

Transaction is the third core entity. Every order is paid by customer’s account through transaction. Every transaction has its own ID, date when the transaction happened and comments of customer. Transactions have several types, such as refund or cancel. If one customer wants to get refund of previous transaction, the system needs to track back to old transaction through its previous ID.

**3.ER Diagram**

Figure 2 shows the ER diagram of HVRS, which contains 10 entities and 11 relationships.

Manage and Own are one-to-one binary relationships,

Store, Act, Rent and Has are many-to-many binary relationships,

The other 5 relationships are one-to-many binary relationships.

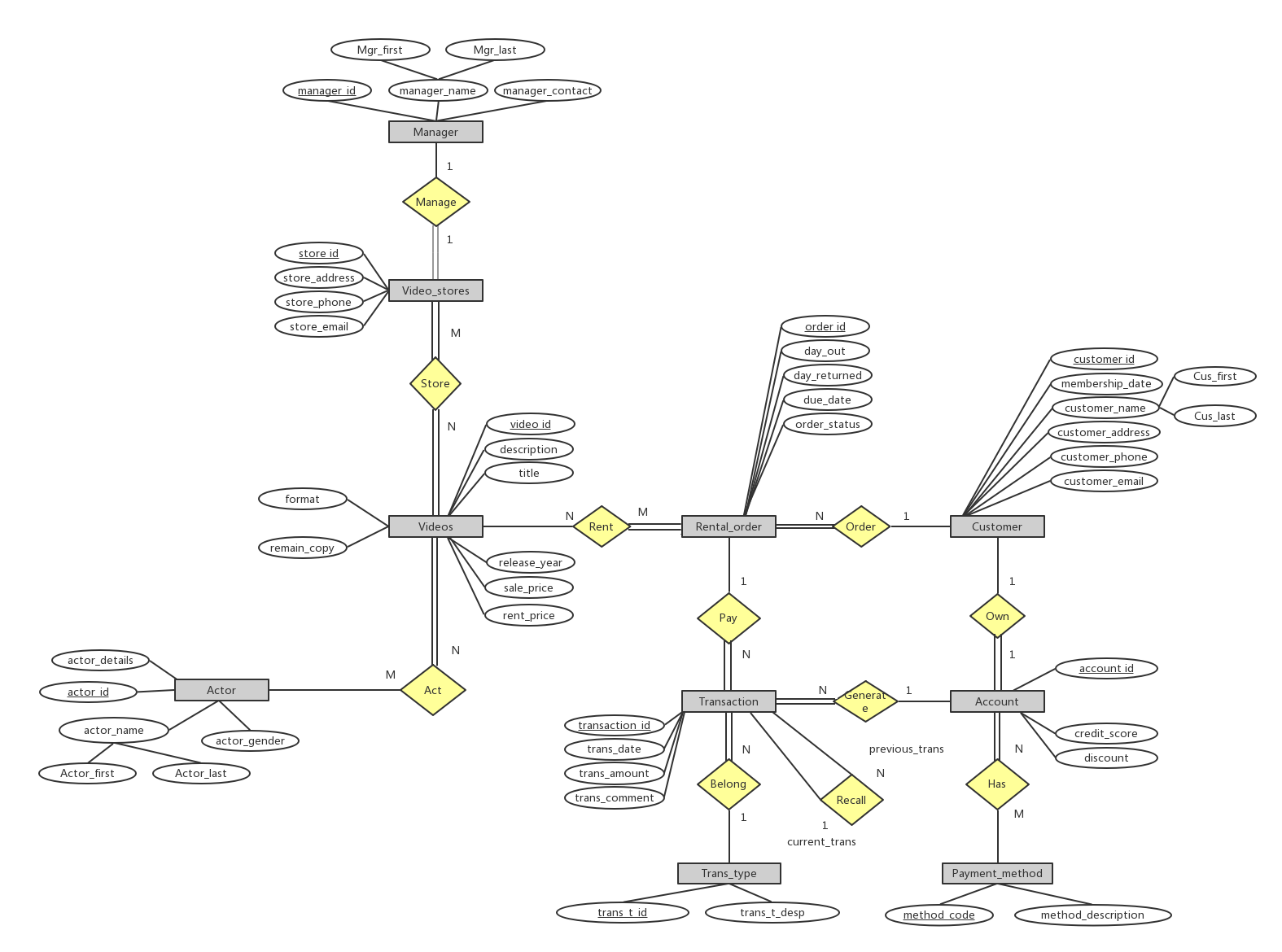


Figure 2. ER diagram of HVRS

**4.Relational Schema**

Figure 3 shows the relational schema of the HVRS, which contains the 10 entities and 4 many-to-many relationships.

For those 2 one-to-one relationships (Manager and Own), add foreign key at the complete participation side. Thus, add Mar\_id of manager to Video\_stores and Cus\_id of Customer to Account.

For those 5 one-to-many relationships, add foreign key at the many side. Thus, add Cus\_id of Customer to Rental\_order and Add Ode\_id of Order, Acc\_id of Account, previous\_Trans\_id of Transaction, Trans\_type\_id of Trans\_type to Transaction. There is one recursive relationship of transaction, that is because customers may want to get refund of previous transaction. In that case, the system needs to go back to previous transaction id through prevous\_Tans\_id.

For those 4 many-to-many relationships, we established 4 new tables with two foreign keys corresponding to those two related entities.

Table 1 in appendix shows a summary of data types and constrains of all attributes table by table.

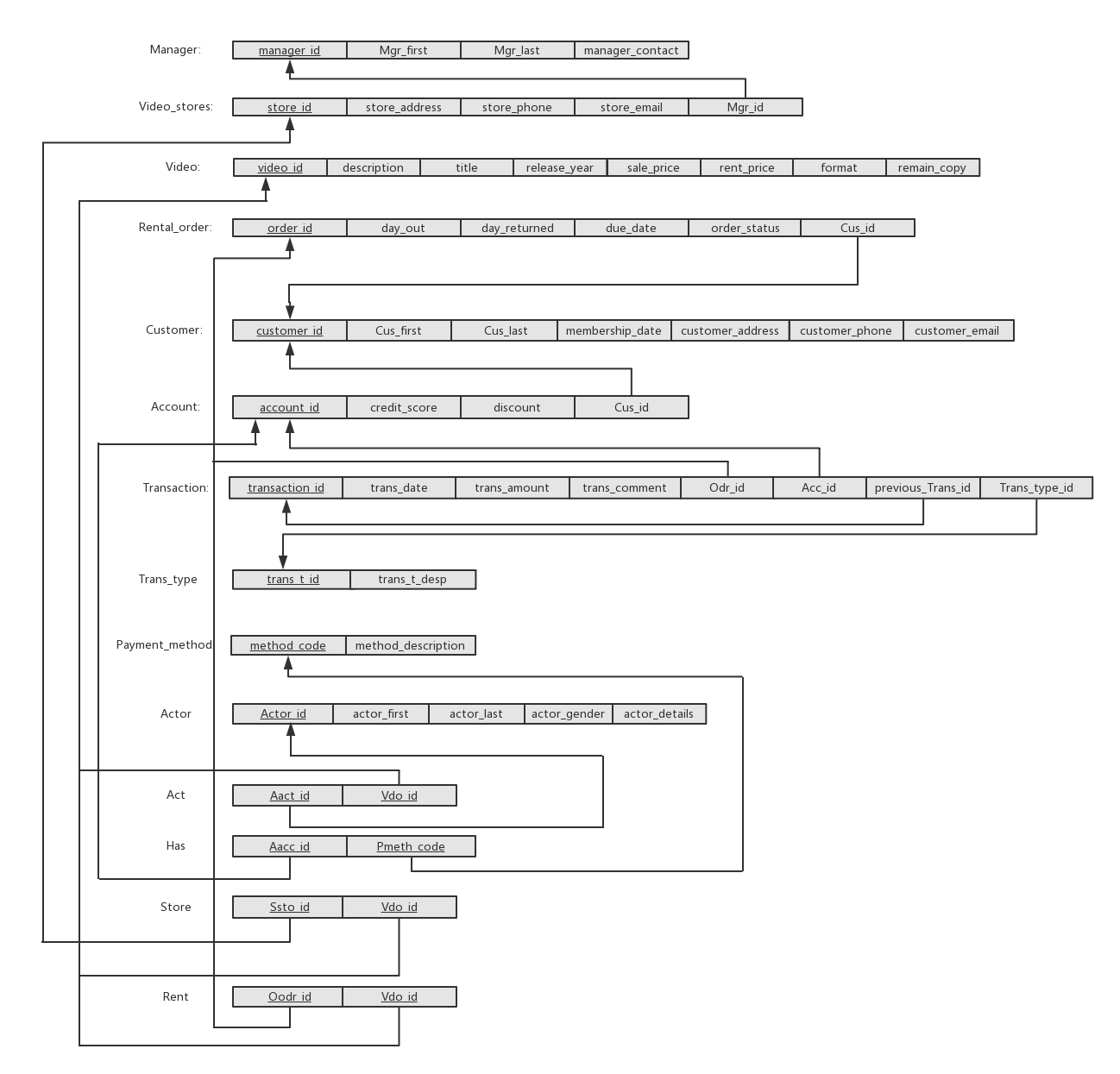
****

Figure 3 Relational schema of the HVRS.

**5.Normalizition to 3NF**

Figure 4 shows the 3NF relational schema of the HVRS. Since the design of HVRS is nice and easy, there is only one table “transaction” is not in 3NF. There is a transitive dependency from Ord\_id to Acc\_id. Build a new table for those two entities. Then the relational schema is normalized to 3NF.

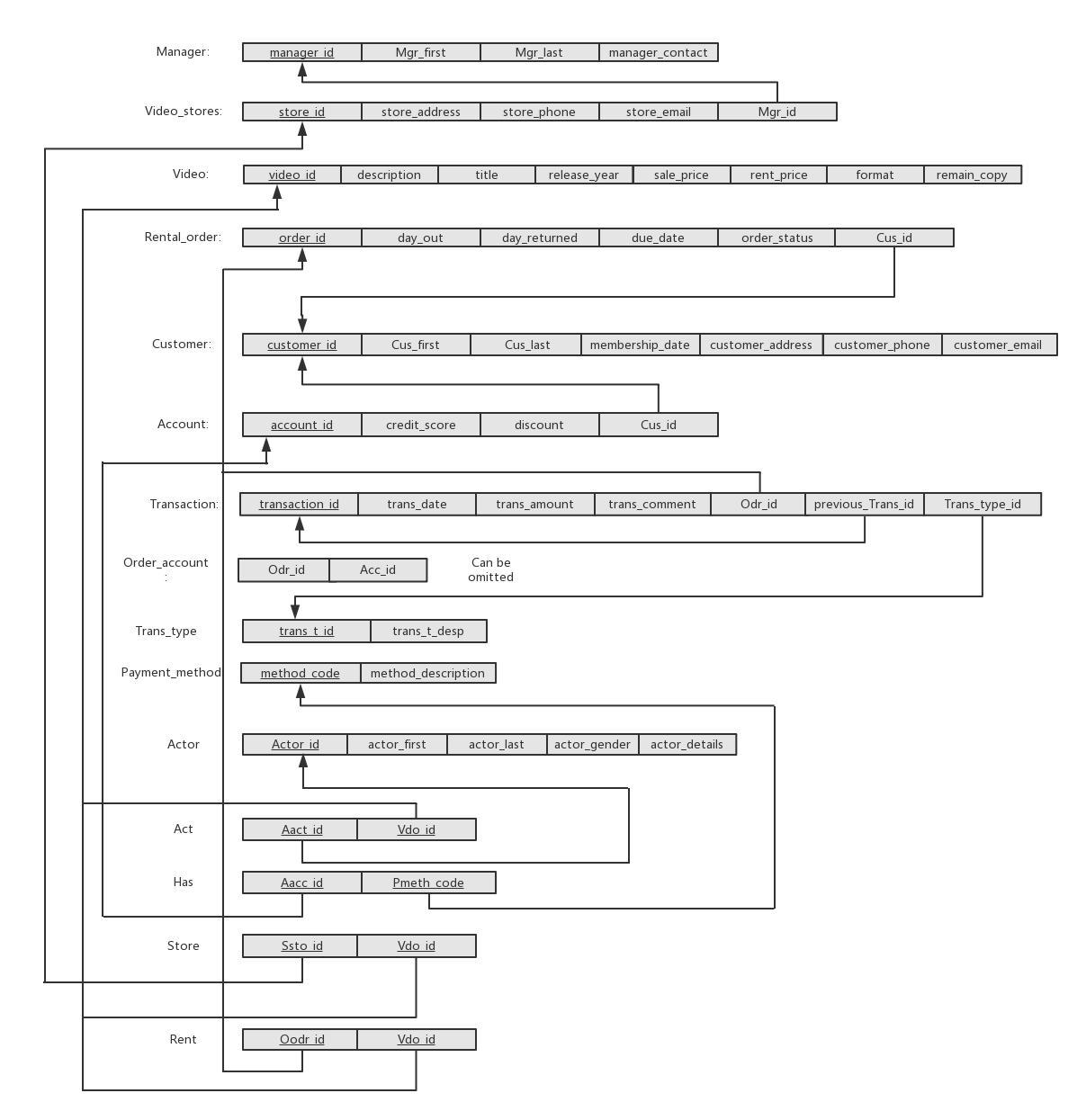
****

Figure 4 3NF relational schema.

**6.Source Code**

The codes for creating tables, triggers and procedures are attached in the Zip file. Since there is some grammar difference between SQL and Mysql excepting creating tables, we implemented two different codes for the SQL developer and MAMP web separately. The file “project final.sql” packages all the code needed for SQL, and the files “Mysql trigger.txt” and “Mysql procedure.txt” contains the triggers and procedures needed for MAMP web.

The code for CRUD operation of Account table in PHP is stored in the folder PHP, including creating one account, reading one account, reading all account, updating one account and deleting one account php files.

**7.Procedures and Triggers**

We wrote two procedures. One of them is aimed to find all the customer\_ids who have video overdue. The other one is aimed to find all the customer\_ids who have ordered more than 5 orders to set them as VIP.

There are also two triggers. One trigger is aimed to set discount for each account according to its credit\_score. If the credit\_score is over 100, the account can get 20% discount. If the credit\_score is less than 50, the discount is 0. If the credit\_score is between 50 and 100, the account can get 10% discount. The other trigger will block transaction if the remain\_copy of the rented video is zero.

**8.PHP Demo**

The screenshots of demo of CRUD operation are stored in 4 folders, create, read, update and delete represents C, R, U and D operation respectively.

Initially, there are 4 customers but only 2 of them have an account. Create demo creates account 2234567 for customer id = 1223355678. Read demo reads the account id = 1234567’s information. It can read all account’s information in case the account id is unknown. Update demo updates the information of account id = 1234567. Delete demo deletes account 2234567.

**Appendix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Nulls | Domain | Reference |
| Manager Table | | | | |
| manager\_id | int (9) | N |  | Unique identifier for a manager |
| Mgr\_first | varchar(15) | N |  | Manager first name |
| Mgr\_last | varchar(15) | N |  | Manager last name |
| manager\_contact | int (10) | N |  | Manager cell phone number |
| Video\_stores | | | | |
| store\_id | int (9) | N |  | Unique identifier for a store |
| store\_address | varchar(200) | N |  | store address (street,city, state, zip) |
| store\_phone | int (10) | N |  | store phone number |
| store\_email | varchar(30) | Y |  | store computer email |
| Mgr\_id | int (9) | N |  | foreign key from manager |
| Video | | | | |
| video\_id | int (15) | N |  | Unique identifier for a video |
| description | varchar(200) | N |  | brief description of the video |
| title | varchar(30) | N |  | video name |
| release\_year | int (4) | Y |  | year it is released |
| sale\_price | decimal(6,2) | N | ####.## | price to sale if want to buy it |
| rent\_price | decimal(5,2) | N | ###.## | price to rent |
| format | varchar(20) | Y |  | opera, science fiction and so on |
| remain\_copy | int (5) | N |  | remain copies. Can not rent if = 0 |
| Rental\_order | | | | |
| order\_id | int(16) | N |  | unique identifier for an order |
| day\_out | date | N |  | date customer borrow video |
| day\_returned | date | N |  | date customer return video |
| due\_data | date | N |  | date video due |
| order\_status | varchar(20) | N |  | order status |
| Cus\_id | int(10) | N |  | foreign key for customer |
| Customer | | | | |
| customer\_id | int(10) | N |  | unique identifier for a customer |
| Cus\_first | varchar(15) | Y |  | customer fisrt name |
| Cus\_last | varchar(15) | Y |  | customer last name |
| membership\_date | date | Y |  | customer membership due date |
| customer\_address | varchar(200) | Y |  | customer address |
| customer\_phone | int(10) | Y |  | customer phone |
| customer\_email | varchar(30) | Y |  | customer email |
| Account | | | | |
| account\_id | int(10) | N |  | unique identifier for customer account |
| credit\_score | int(10) | N |  | credit score for customer account |
| discount | decimal(0,2) | Y |  | discount for account |
| Cus\_id | int(10) | N |  | foreign key for customer |
| Transaction | | | | |
| transaction\_id | int(16) | N | Sequence # | Unique identifier for a transaction |
| trans\_date | date | Y |  | Transaction date |
| trans\_amount | int(10) | Y |  | Transaction amount |
| trans\_comment | varchar(200) | Y |  | Transaction comment |
| Odr\_id | int(16) | N | Order id # | Reference an order |
| previous\_Trans\_id | int(16) | Y |  | Previous transaction id |
| trans\_type\_id | int(10) | N |  | Reference a transaction type |
| Transaction\_type | | | | |
| transaction\_type\_id | int(10) | N |  | Unique identifier for a transaction type id |
| trans\_type\_description | varchar(20) | Y |  | Transaction type description |
| Payment\_method | | | | |
| method\_code | int(10) | N |  | Unique identifier for a payment method code |
| method\_description | varchar(20) | Y |  | Payment method description |
| Actor | | | | |
| Actor\_id | int(10) | N | Actor # | Unique identifier for an actor |
| actor\_first | varchar(20) | Y |  | Actor first name |
| actor\_last | varchar(20) | Y |  | Actor last name |
| actor\_gender | varchar(10) | Y |  | Actor gender |
| actor\_detail | varchar(200) | Y |  | Actor information |
| Act | | | | |
| Aact\_id | int(10) | N |  | foreign key from actor |
| Vdo\_id | int (15) | N |  | foreign key from video |
| Has | | | | |
| Aacc\_id | int(10) | N |  | foreign key from account |
| Pmeth\_code | int(10) | N |  | foreign key from Payment\_method |
| Store | | | | |
| Ssto\_id | int (9) | N |  | foreign key from Video\_stores |
| Vdo\_id | int (15) | N |  | foreign key from Video |
| Rent | | | | |
| Oodr\_id | int (16) | N |  | foreign key from Rental\_order |
| Vdo\_id | int (15) | N |  | foreign key from Video |

Table 1 A summary of data types and constrains of all attributes.