Assignment 4 - CSC 484 - John, Izak, Frezer

Students

John Akujobi
Izak Benitez Lopez
Frezer Berhanu

Question

A video rental store used the following table to store the video rental information. Each video has a unique identifier and has one value for its Title, Star, Year Filmed, and Length
Each customer is uniquely identified (cuID) For each customer, their name (cuName) and address (cuAddress) are also recorded

Each time a customer rents a video, the Ship Date and Return Date are recorded

VideoRental

Video ID	Title	Star	Year Filmed	Length	cuID	cuName	cuAddress	Ship Date	Return Date
001	Deadpool	Reynolds	2016	1:48	c001	John	110 Main Ave	4/11/18	4/18/18
					c003	Sally	302 5th St	5/5/18	5/12/18
					c002	Bill	101 1st Ave	1/1/19	1/8/19
002	Empire Records	LaPaglia	1995	1:30	c002	Bill	101 1st Ave	1/1/19	1/8/19
					c004	Mary	202 6th St	4/1/19	4/8/19
					c005	Twila	333 Medary Ave	7/4/18	7/11/18

Apply Normalization to decompose the table VideoRental into a set of 3NF relations

- Show this data in 1NF
- Show this data in 2NF
- Show this data in 3NF

Show all:

- Primary Keys
- Foreign Keys

> **Info**
>
> I am quite unsure of the restrictions that you want us to abide by for this assignment. This is especially regarding creating new attributes for the data.
>
> So, to be on the safe side, i will not make any adjustments, additions or modifications to the attributes of the data

---

First Normal Form 1NF - Restricted Attributes

Reasoning from UNF to 1NF

- To stop repeating the same video and customer details in every rental record. This reduces the amount of duplicate data.
- To ensure that every data item is stored as its smallest individual unit, making the database cleaner and more organized.
- To help find and organize data by a unique key.

Without adding any new attributes, here is the First Normal Form 1NF

- Each row has a unique identity, and each piece of information stands alone.

Video ID	Title	Star	Year Filmed	Length	cuID	cuName	cuAddress	Ship Date	Return Date
001	Deadpool	Reynolds	2016	1:48	c001	John	110 Main Ave	4/11/18	4/18/18
001	Deadpool	Reynolds	2016	1:48	c003	Sally	302 5th St	5/5/18	5/12/18
001	Deadpool	Reynolds	2016	1:48	c002	Bill	101 1st Ave	1/1/19	1/8/19
002	Empire Records	LaPaglia	1995	1:30	c002	Bill	101 1st Ave	1/1/19	1/8/19
002	Empire Records	LaPaglia	1995	1:30	c004	Mary	202 6th St	4/1/19	4/8/19

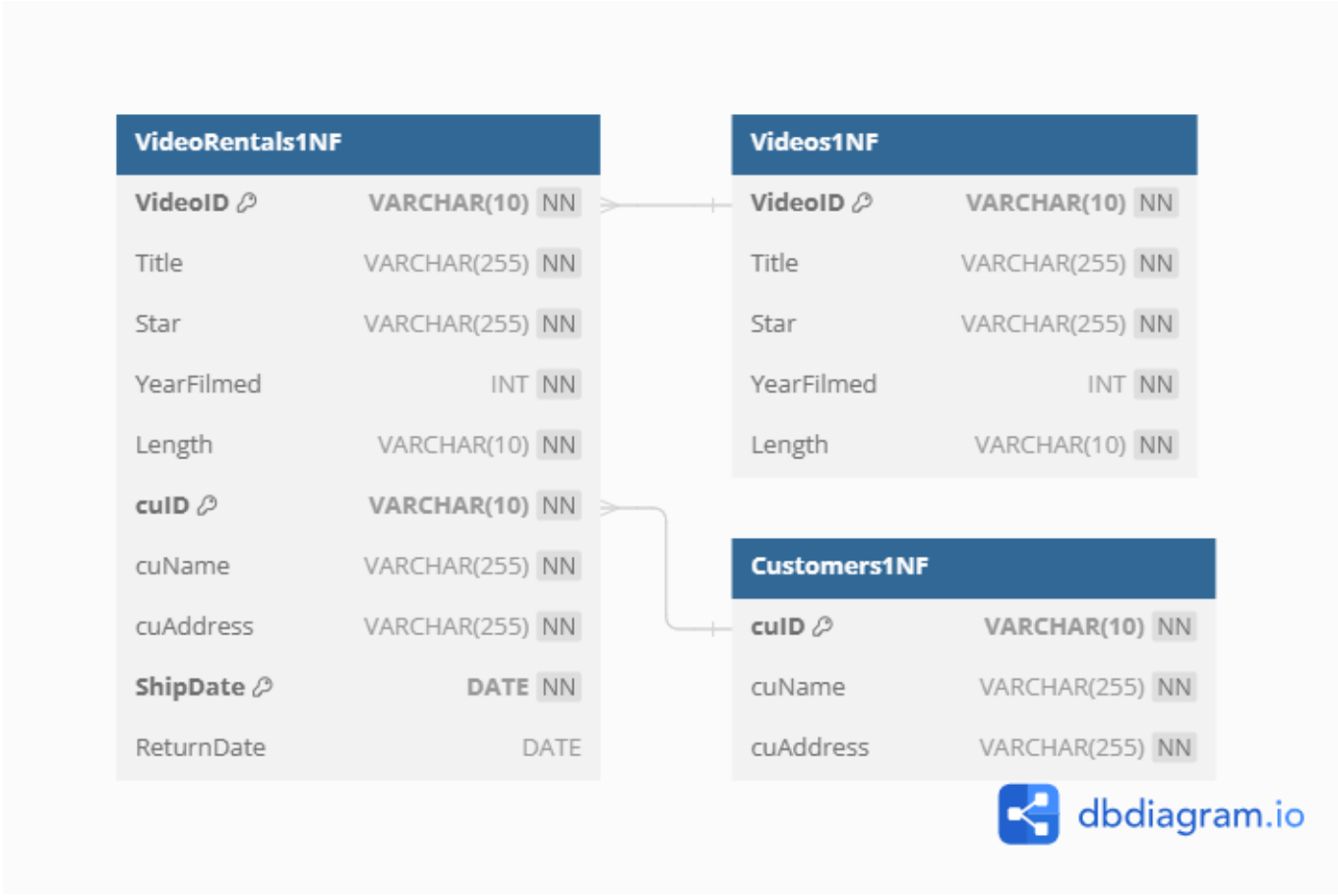
Video ID	Title	Star	Year Filmed	Length	cuID	cuName	cuAddress	Ship Date	Return Date
002	Empire Records	LaPaglia	1995	1:30	c005	Twila	333 Medary Ave	7/4/18	7/11/18

Keys

- **Primary Key**:
  - Here, we use a combination of `VideoID`, `cuID`, and `ShipDate`.
- **Foreign Keys**:
  - `VideoID` references a `Videos` table, which is not shown here but would contain columns `VideoID`, `Title`, `Star`, `YearFilmed`, and `Length`.
  - `cuID` references a `Customers` table, which would contain `cuID`, `cuName`, and `cuAddress`.
- Code to make Video Rentals

```
CREATE TABLE VideoRentals (
    VideoID VARCHAR(10) NOT NULL,
    Title VARCHAR(255) NOT NULL,
    Star VARCHAR(255) NOT NULL,
    YearFilmed INT NOT NULL,
    Length VARCHAR(10) NOT NULL,
    cuID VARCHAR(10) NOT NULL,
    cuName VARCHAR(255) NOT NULL,
    cuAddress VARCHAR(255) NOT NULL,
    ShipDate DATE NOT NULL,
    ReturnDate DATE,
    PRIMARY KEY (VideoID, cuID, ShipDate),
    FOREIGN KEY (VideoID) REFERENCES Videos(VideoID),
    FOREIGN KEY (cuID) REFERENCES Customers(cuID)
);
```

DB Diagram



Code used in dbdiagram.io

```
Table "VideoRentals1NF" {
  "VideoID" VARCHAR(10) [not null]
  "Title" VARCHAR(255) [not null]
  "Star" VARCHAR(255) [not null]
  "YearFilmed" INT [not null]
  "Length" VARCHAR(10) [not null]
  "cuID" VARCHAR(10) [not null]
  "cuName" VARCHAR(255) [not null]
  "cuAddress" VARCHAR(255) [not null]
  "ShipDate" DATE [not null]
  "ReturnDate" DATE

Indexes {
  (VideoID, cuID, ShipDate) [pk]
}
}

Table "Videos1NF" {
  "VideoID" VARCHAR(10) [not null]
  "Title" VARCHAR(255) [not null]
  "Star" VARCHAR(255) [not null]
```

```
  "YearFilmed" INT [not null]
  "Length" VARCHAR(10) [not null]

Indexes {
  VideoID [pk]
}
}

Table "Customers1NF" {
  "cuID" VARCHAR(10) [not null]
  "cuName" VARCHAR(255) [not null]
  "cuAddress" VARCHAR(255) [not null]

Indexes {
  cuID [pk]
}
}

Ref:"Videos1NF"."VideoID" < "VideoRentals1NF"."VideoID"

Ref:"Customers1NF"."cuID" < "VideoRentals1NF"."cuID"
```

Second Normal Form (2NF)

Reasons from 1NF to 2NF

- To make sure each piece of information is fully dependent on a unique key.
  For example, video details (like title and year) should not need to be repeated for each rental if they can be identified by `VideoID` alone.
- To eliminate cases where part of the primary key determines some information but not all. This helps to further reduce unnecessary duplication.

We'll create three separate tables. To do so, we split up tables when a piece of information doesn't depend on the whole key.
Made one for `Videos`, one for `Customers`, and a new `Rentals` table to handle the relationships and transaction-specific details (like dates).

- This stops the same details from being entered multiple times across different rentals.

Videos Table

Stores details about each video, including its unique identifier, title, star, year filmed, and length.

- It eliminates redundancy by centralizing all data related to the videos themselves.
- This makes sure that each video's details are stored only once.

VideoID	Title	Star	Year Filmed	Length
001	Deadpool	Reynolds	2016	1:48
002	Empire Records	LaPaglia	1995	1:30

SQL for Videos Table

```
CREATE TABLE Videos (
    VideoID VARCHAR(10) NOT NULL,
    Title VARCHAR(255) NOT NULL,
    Star VARCHAR(255) NOT NULL,
    YearFilmed INT NOT NULL,
    Length VARCHAR(10) NOT NULL,
    PRIMARY KEY (VideoID)
);
```

Customers Table

Holds information about each customer, such as their unique identifier, name, and address.

- By storing each customer's details in one place, the system avoids duplication.
- Our system also simplifies updates (like changing an address, name).

cuID	cuName	cuAddress
c001	John	110 Main Ave
c002	Bill	101 1st Ave
c003	Sally	302 5th St
c004	Mary	202 6th St
c005	Twila	333 Medary Ave

SQL for Customers Table

```
CREATE TABLE Customers (
    cuID VARCHAR(10) NOT NULL,
    cuName VARCHAR(255) NOT NULL,
    cuAddress VARCHAR(255) NOT NULL,
```

```
    PRIMARY KEY (cuID)
);
```

Rentals Table

Records each rental transaction, linking customers to the videos they rent, along with the shipping and return dates.

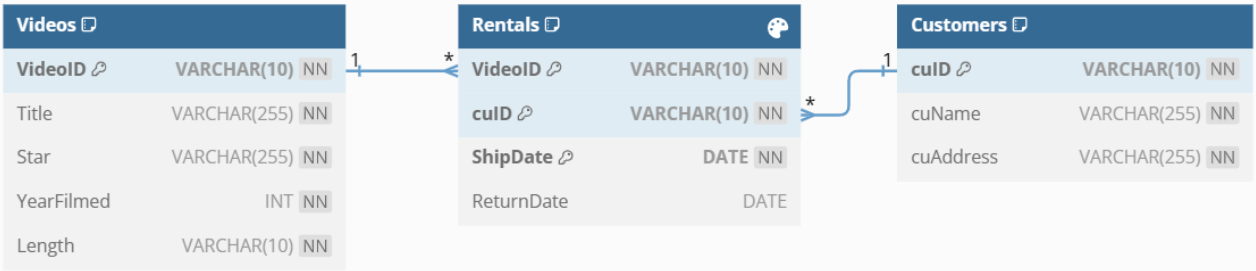
- It uniquely identifies each rental instance by combining `VideoID`, `cuID`, and `ShipDate`, which together form a composite primary key.
- This way, even if the same customer rents the same video multiple times, each transaction is uniquely recorded.

VideoID	cuID	ShipDate	ReturnDate
001	c001	4/11/18	4/18/18
001	c003	5/5/18	5/12/18
001	c002	1/1/19	1/8/19
002	c002	1/1/19	1/8/19
002	c004	4/1/19	4/8/19
002	c005	7/4/18	7/11/18

SQL code for Rentals Table

```
CREATE TABLE Rentals (
    VideoID VARCHAR(10) NOT NULL,
    cuID VARCHAR(10) NOT NULL,
    ShipDate DATE NOT NULL,
    ReturnDate DATE,
    PRIMARY KEY (VideoID, cuID, ShipDate),
    FOREIGN KEY (VideoID) REFERENCES Videos(VideoID),
    FOREIGN KEY (cuID) REFERENCES Customers(cuID)
);
```

Diagram for 2NF



Code used for dbdiagram.io

```
Table "Videos" {
  "VideoID" VARCHAR(10) [not null]
  "Title" VARCHAR(255) [not null]
  "Star" VARCHAR(255) [not null]
  "YearFilmed" INT [not null]
  "Length" VARCHAR(10) [not null]

Indexes {
  VideoID [pk]
}
}

Table "Customers" {
  "cuID" VARCHAR(10) [not null]
  "cuName" VARCHAR(255) [not null]
  "cuAddress" VARCHAR(255) [not null]

Indexes {
  cuID [pk]
}
}

Table "Rentals" {
  "VideoID" VARCHAR(10) [not null]
  "cuID" VARCHAR(10) [not null]
  "ShipDate" DATE [not null]
  "ReturnDate" DATE

Indexes {
  (VideoID, cuID, ShipDate) [pk]
}
}
```

```
Ref:"Videos"."VideoID" < "Rentals"."VideoID"

Ref:"Customers"."cuID" < "Rentals"."cuID"
```

Third Normal Form (3NF)

Reasons from 2NF to 3NF

- To ensure that no piece of information depends on another piece of non-key information. For instance, a customer's name should not be defined or altered by their rental details or vice versa.
- To make the database more efficient and easier to maintain because each piece of information directly depends on the primary key, without being affected by other non-key attributes.

Check:

1. It is in Second Normal Form (2NF).
2. It has no transitive dependencies (i.e., non-key attributes do not depend on other non-key attributes).

1. Videos Table

No Changes Needed: This table is already in 3NF as there are no transitive dependencies, and all non-key attributes depend solely on the primary key.

- `VideoID` is the primary key, and all other attributes (`Title` , `Star` , `YearFilmed` , `Length`) are dependent only on `VideoID` and not on each other.

VideoID	Title	Star	Year Filmed	Length
001	Deadpool	Reynolds	2016	1:48
002	Empire Records	LaPaglia	1995	1:30

---

2. Customers Table

No Changes Needed: All attributes are directly related to the primary key, and no attribute is dependent on another non-key attribute.

- `cuID` is the primary key, and both `cuName` and `cuAddress` depend only on `cuID`.

cuID	cuName	cuAddress
c001	John	110 Main Ave
c002	Bill	101 1st Ave
c003	Sally	302 5th St
c004	Mary	202 6th St
c005	Twila	333 Medary Ave

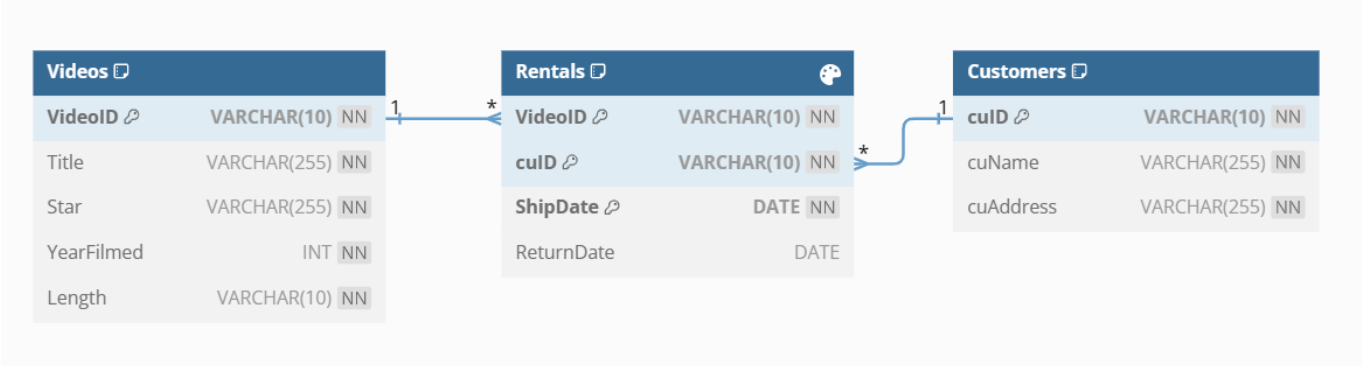
---

3. Rentals Table

No Changes Needed: In this table, `ReturnDate` depends only on the primary key components (specifically, it can be argued to be a function of `ShipDate` and the others as part of the transaction details). There is no dependency between non-key attributes.

- The composite primary key is (`VideoID`, `cuID`, `ShipDate`), and `ReturnDate` is dependent on the primary key.

VideoID	cuID	ShipDate	ReturnDate
001	c001	4/11/18	4/18/18
001	c003	5/5/18	5/12/18
001	c002	1/1/19	1/8/19
002	c002	1/1/19	1/8/19
002	c004	4/1/19	4/8/19
002	c005	7/4/18	7/11/18



✓ **Success**

All three tables (Videos, Customers, Rentals) are structured to meet the requirements of 3NF: