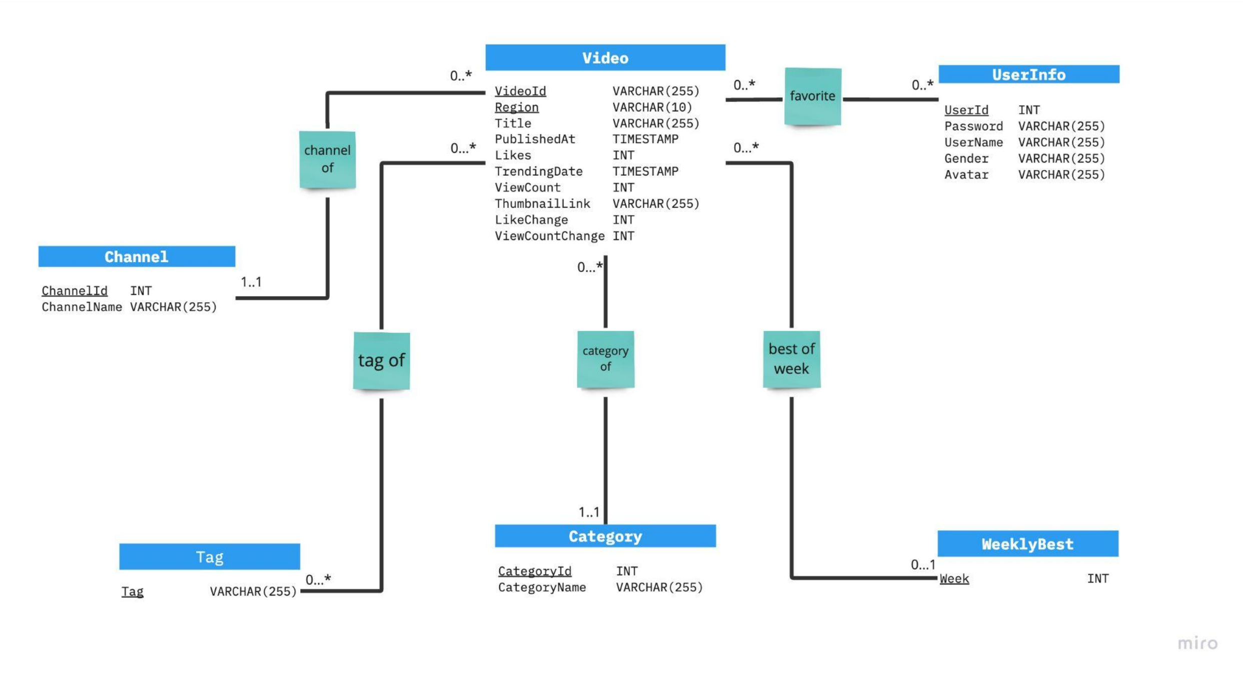
# Stage 2: Database Design

## 1. UML



## 2. Descriptions

We have six entities in total, where four of them mainly utilize the attributes from Youtube Trending Video Dataset and the rest are created by ourselves serving for implementing the functionalities of user login and weekly recommendations.

We contain most attributes in the **Video** entity using VideoId and Region as primary keys since there might be the same video in more than one region. By combining them, we can uniquely identify each video. It has:

- **many-to-one** relationship with **Channel** since all the videos are separated into limited number of channels, while each channel can have many videos;
- **many-to-many** relationship with **Tag** due to their diversity(eg. each video may have zero to many tags, and each tag serves for zero to many videos);
- **many-to-one** relationship with **Category** since all the videos are separated into limited number of categories, and each category contains hundreds of videos;
- **many-to-one** relationship with **WeeklyBest**, all the trending videos will only be selected as weekly best for exactly one week, while for each week there are several best videos;
- **many-to-many** relationship with **User**, we have a relation table **Favorite** indicating that each user can have multiple favorite videos and each video can be the favorite of multiple users

## 3. Assumptions

We make the following assumptions regarding our entities and relationships:

a) Each video should have exactly one channel
b) Each video should have exactly one category
c) Each tag can have many videos associated and video can have many tags
d) Each user can have many favorite videos and each video can be liked by multiple users
e) Each video can have at most one week in which it is selected as weely best
f) Each (VideoId, Region) pair should be unique
g) Each ChannelId should be unique
h) Each CategoryId should be unique
i) Each tag should be unique
j) Likes and ViewCount should always be a natural number
k) LikeChange and ViewCountChange should always be a natural number
l) TrendingDate should not precede PublishedAt
m) Passwords should be longer than 8 characters and contain only both upper and lower letters, numbers, and special characters.

## 4. Normalization

We adopt BCNF for normalization because the result of 3NF would yield much more tables based on the existing FDs and make the interaction with the database inefficient. Our functional dependencies are:

- VideoId, Region → Title, PublishedAt, ChannelId, Likes, CategoryId, TrendingDate, ViewCount, ThumbnailLink, ViewCountChange, LikesChange
- ChannelId → ChannelTitle
- CategoryId → CategoryName
- UserId → Password, UserName, Gender, Avatar
- VideoId → Week

We start with an entity containing all attributes. The superkey for this entity is (VideoId, Region, UserId). We normalize the entity by following:

a) We process the first FD and separate the entity with {VideoId, Region}+ and VideoId union (R - {VideoId, Region}+):
(VideoId, Title, PublishedAt, ChannelId, ChannelTitle, Likes, CategoryId, CategoryName, TrendingDate, ViewCount, ThumbnailLink, ViewCountChange, LikesChange, Region, Week),
(VideoId, UserId, Password, UserName, Gender, Avatar, Tag)
b) We process the second FD and separate the entity with {ChannelId}+ and ChannelId union (R - {ChannelId}+):
(VideoId, Title, PublishedAt, ChannelId, Likes, CategoryId, CategoryName, TrendingDate, ViewCount, ThumbnailLink, ViewCountChange, LikesChange, Region, Week),
(ChannelId, ChannelTitle),

(VideoId, Region, UserId, Password, UserName, Gender, Avatar, Tag)

c) We process the third FD and separate the entity with {CategoryId}+ and CategoryId union (R - {CategoryId}+):
(VideoId, Title, PublishedAt, ChannelId, CategoryId, Likes, TrendingDate, ViewCount, ThumbnailLink, ViewCountChange, LikesChange, Region, Week),
(ChannelId, ChannelTitle),
(CategoryId, CategoryName),
(VideoId, Region, UserId, Password, UserName, Gender, Avatar, Tag)

d) We process the fourth FD and separate the entity with {UserId}+ and UserId union (R - {UserId}+):
(VideoId, Title, PublishedAt, ChannelId, CategoryId, Likes, TrendingDate, ViewCount, ThumbnailLink, ViewCountChange, LikesChange, Region, Week),
(ChannelId, ChannelTitle),
(CategoryId, CategoryName),
(VideoId, Region, UserId, Tag),
(UserId, Password, UserName, Gender, Avatar)

e) We process the fifth FD and separate the entity with {VideoId}+ and VideoId union (R - {VideoId}+):
(VideoId, Region, Title, PublishedAt, ChannelId, CategoryId, Likes, TrendingDate, ViewCount, ThumbnailLink, ViewCountChange, LikesChange),
(VideoId, Week),
(ChannelId, ChannelTitle),
(CategoryId, CategoryName),
(VideoId, Password, UserName, Gender, Avatar),
(VideoId, Region, Tag, UserId)

**5. Relational schema**

**a) Entities:**

**Video** (
VideoId: VARCHAR(255) [PK],
Region: VARCHAR(10) [PK],
Title: VARCHAR(255),
PublishedAt: TIMESTAMP,
Likes: INT,
TrendingDate: TIMESTAMP,
ViewCount: INT,
ThumbnailLink: VARCHAR(255),
LikesChange: INT,
ViewCountChange: INT
ChannelId: INT [FK to **Channel**.ChannelId]

CategoryId: INT [FK to **Category**.CategoryId]
Week: INT [FK to **WeeklyBest**.Week]
)

**Channel** (
ChannelId: INT [PK],
ChannelTitle: VARCHAR(255),
)

**Category** (
CategoryId: INT [PK],
CategoryName: VARCHAR(255)
)

**UserInfo** (
UserId: INT [PK],
Password: VARCHAR(255)
UserName: VARCHAR(255)
Gender: VARCHAR(31)
Avatar: VARCHAR(255)
)

**WeeklyBest** (
Week: TIMESTAMP [PK]
)

**Tag** (
Tag: VARCHAR(255) [PK]
)

**b) Relation Tables:**
**Favorite** (
VideoId: VARCHAR(255) [PK] [FK to **Video**.VideoId],
UserId: INT [PK] [FK to **User**.UserId]
)

**TagOf** (
VideoId: VARCHAR(255) [PK] [FK to **Video**.VideoId],
Tag: VARCHAR(255) [PK] [FK to **Tag**.tag]
)