Relational Schema

Creators(channelId:VARCHAR(50) [PK], channelTitle:VARCHAR(100), categoryId:INT, video_id:VARCHAR(50) [FK to VideoInfo.video_id])

VideoInfo(video_id:VARCHAR(50) [PK], title:VARCHAR(100), publishedAt: TEXT, tags:TEXT, description:TEXT, channelId:VARCHAR(50), categoryId:INT, view_count: INT, likes: INT, dislikes: INT, comment_count: INT)

CategoryInfo(categoryId:INT [PK] [FK to VideoInfo.categoryId], totalPublished:INT, totalLiked:BIGINT, totalChannels:INT, totalViews:BIGINT)

TrendingKeywords(keywords: VARCHAR(50) [PK], categoryId: INT, use_count: INT)

WebsiteUsers(user_id: INT [PK], username: VARCHAR(30), password: VARCHAR(30), email: VARCHAR(50), channelId: VARCHAR(50) [FK to Creators.channelId])

KeywordToVideo(VideoInfo.video_id, TrendingKeywords.category_id)

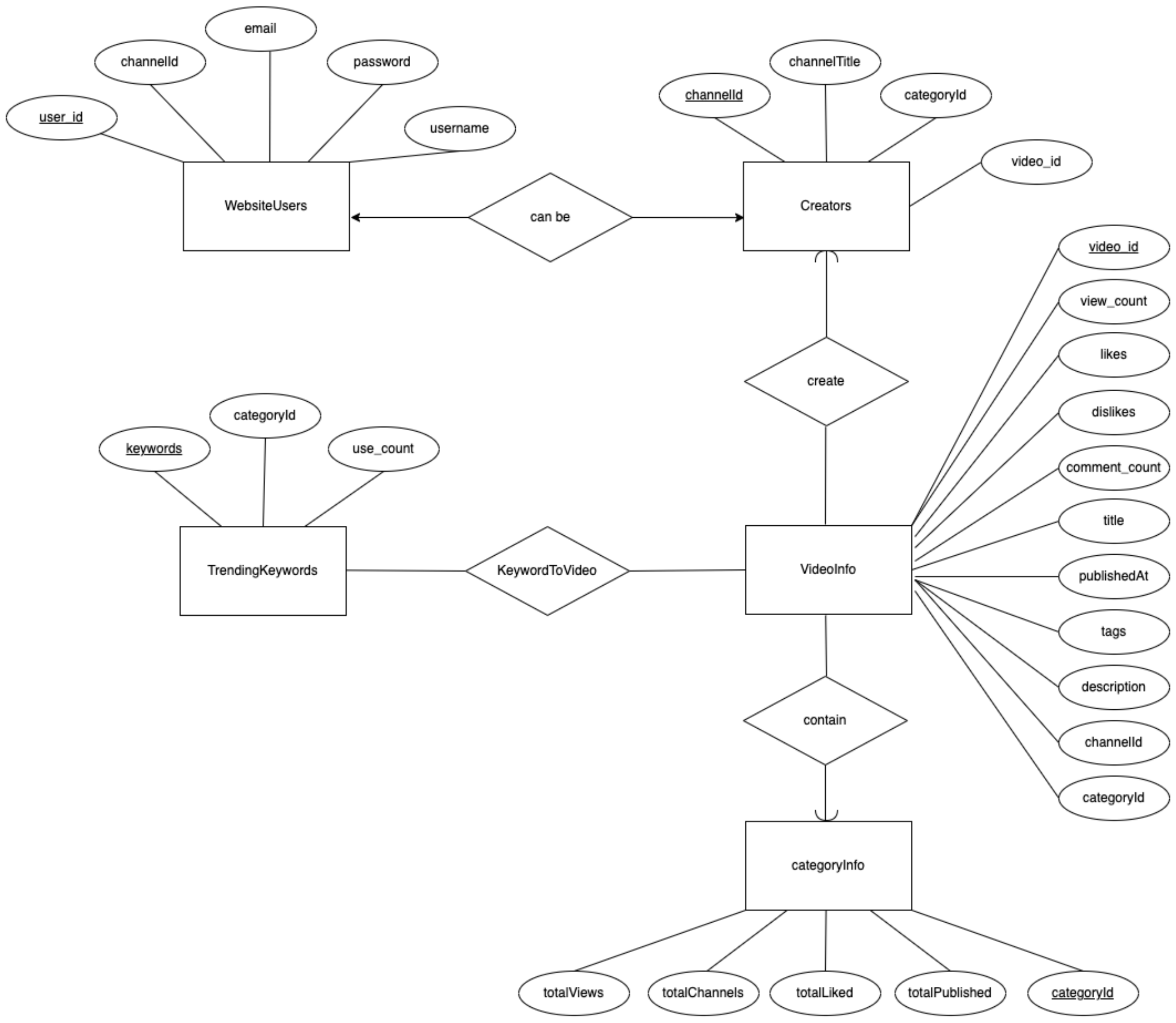
Assumptions made:

1. Creators to VideoInfo: We assume that for every one creator, there will be many videos created, with video information for each video made. As such, it is a one-to-many relationship.
2. VideoInfo to TrendingKeywords: We assume that each video may have multiple trending keywords within its title or tag. We also assume that each keyword must be within many videos since the keywords attribute is generated through analysis on the VideoInfo table. Thus, this relationship is a many-to-many relationship.
3. WebsiteUsers to Creators: Our web app allows creators to sign up as website users to track their own video metrics. In this case, we assume an optional one-to-one relationship to say that if one website user signed up, they can be linked to a content creator. Similarly, a content creator can be linked to one website user. We keep this optional because perhaps not all content creators will sign up to use the website.
4. VideoInfo to CategoryInfo: We assume that each category will have multiple videos within it. Thus, we establish this relationship as one-to-many.
5. When a YouTube channel is deleted, the associated videos are also deleted. However, since this project is focusing on the analysis of what makes a Youtube video popular, we will be including videos regardless if the channel has been deleted.

Description of Relationships:

1. Creators: This table holds all the information related to the people who post videos. It contains a unique channel_id to identify each video as well as the channel title and overall category to which the channel's content is related.
2. VideoInfo: This table holds all the information related to the initial posting of a video as well as all the relevant information pertaining to the performance of a video. It contains a unique video_id to identify each video, and then holds the video's title, published date and time, tags and a description written when posting it, the YouTube channel it was posted to, and the category that the video falls under. Furthermore, this would include metrics such as view count, likes, comment count, etc. The performance metrics will be necessary to determine which keywords and categories result in the most successful youtube videos.
3. TrendingKeywords: This table was created to allow us easy and instant access to popular keywords based on video category. The table holds keywords, the category the keyword is associated with, and tells the use_count which is the total number of times the keyword was used across all videos. This will help us rank the most popular keywords based on the specific category it is in. We will be collecting this data from the VideoInfo Table using the title, tags, view_count, and likes attributes. We can sort the videos by their view_count and likes and then fetch their corresponding title and tags. Then, to determine keywords, we will iterate through the text of the title field as well as the list of associated tags and filter out the most commonly used words using some basic NLP packages. We decided to create a separate table as TrendingKeywords to improve the overall performance of our application. Doing these operations every time a user wants to see the trending keywords would be redundant and inefficient. By storing it in a separate table, the application more efficiently uses computation power.
4. WebsiteUsers: This table includes all the information that is unique to each website user. It contains the user's login, password, channel id, and email. This table also allows us to link users with their individual content platforms.
5. CategoryInfo: This table holds information about trending videos pertaining to their category. It contains the category id, the total number of videos published under that category (totalPublished), the number of videos that have more likes than dislikes (totalLiked), the number of channels that have published videos under each category (totalChannels), and the total view count for all the videos in this category (totalViews). This would be helpful in understanding which categories are most popular and help us understand the barrier to entry into this genre based on how many creators exist within it. This table would also connect to trending keywords, using the categoryId as a foreign key.

ER Diagram


email
channelId
password
user_id
username
WebsiteUsers
can be
Creators
channelTitle
channelId
categoryId
video_id
video_id
view_count
likes
dislikes
comment_count
title
publishedAt
tags
description
channelId
categoryId
create
categoryId
keywords
use_count
TrendingKeywords
KeywordToVideo
VideoInfo
contain
categoryInfo
totalViews
totalChannels
totalLiked
totalPublished
categoryId