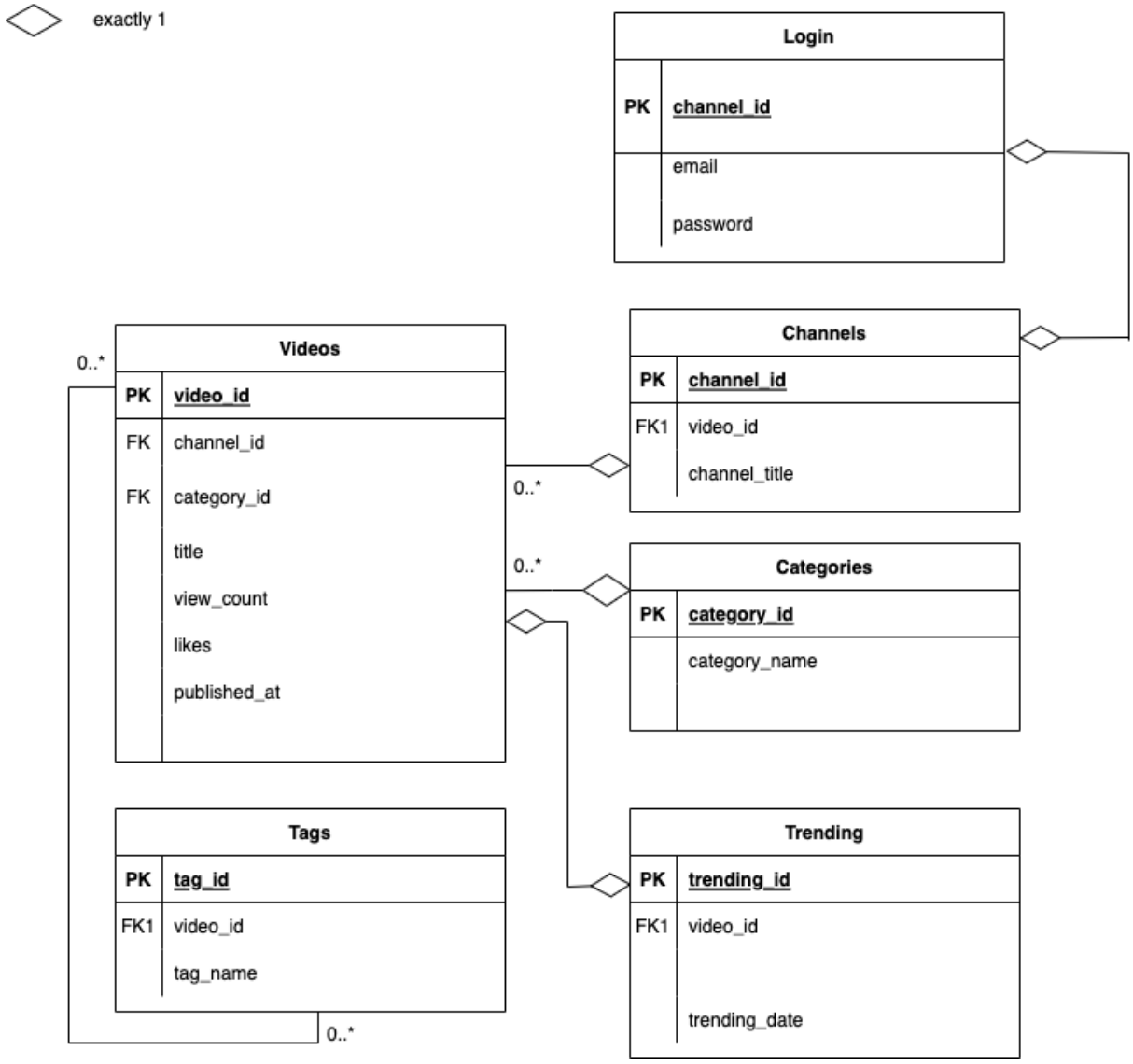
Stage 2: Database Design

MyTube UML Diagram



- **Should it be a diamond or 1-to-1? Should the foreign keys be indicated on the**

Normalization and Process:

Looking at our schema, “Channels”, “Trendings”, and “Categories” tables are already in 3NF since they have a primary key and contain no keys that are dependent on anything but their respective primary key.

The “Videos” table is the most complex of them all. It relates to categories, tags, trendings, and channels. It’s in 3NF since there are no partial dependencies and also no transitive dependencies.

The “Tags” table has a many to many relationship with the “Videos” table. For our project, we want our tags to be universal and can be reused across videos. Therefore, further normalization is needed to remove redundancy.

Looking at all the above summaries of each individual table, we think that 3NF is more suitable. Because there can be many videos in categories, videos in channels, videos that are trending, etc, we think that redundancy will be a big theme for our project. 3NF is comparatively a lot more redundant than BCNF.

To prove that our table is already in 3NF, here’s a list of the functional dependencies.

- Login -> Channels
- Channels -> Login; Videos
- Videos -> Channels; Categories; Trending; Tags
- Categories -> Videos
- Trending -> Videos
- Tags -> Videos

Find Minimal Basis:

- Login -> Channels
- Channels -> Login
- Channels -> Videos
- Videos -> Channels
- Videos -> Categories
- Categories -> Videos
- Videos -> Trending
- Trending -> Videos
- Tags -> Videos
- Videos -> Tags

From our functional dependencies and the rules of 3NF, we find that our table is already in 2NF and there are no transitive functional dependencies of non-foreign/non-primary attributes on the super key.

From our minimal basis, we see many bidirectional relationships, indicating that there are multiple

Description of Relationship and Cardinality:

Videos can be considered as our main table. It has a many to many relationship with Tags. It has a one to one relationship with trending. It has a one to one relationship with channels. Channels also has a many to many relationship with categories and a one to one relationship with login. Below is an explanation of the assumptions we are making when defining relationships:

Videos - A video has exactly 1 channel since only a single channel can upload a video, A video has exactly 1 category associated with it since only 1 category can be assigned to a video, A video has exactly 1 trending since a video can only appear on the trending tab once, A video can have 0 to many tags since tags describe the video and there are many ways to describe the video.

Login - A login has exactly 1 channel since 1 login has only a single channel associated with it

Channels - A channel has exactly 1 login since 1 channel has exactly a single channel associated with it. A channel has 0 to many videos associated with it since a single channel can upload multiple videos.

Categories - A single category can have 0 to many videos since for example the category "Educational" can contain videos about Python to Volcanoes.

Trending - A trending id has exactly 1 video on it since it would be redundant to put the same video on trending multiple times.

Tags - A tag can have multiple videos associated with it. Consider the tag "podcast", which can have videos from NPR to JJ Reddick for example.

Relational Schema

Videos(video-id:VARCHAR(255) [PK], channel_id: VARCHAR(255) [FK to Channels.channel_id], category_id:VARCHAR(255)[FK to Categories.category_id], title:VARCHAR(255), view_count: INT, likes: INT, published_at: DATE/TIME)

Login(channel_id:VARCHAR(255) [PK], email:VARCHAR(255) , password:VARCHAR(255))

Channels(channel_id:VARCHAR(255) [PK], video_id:VARCHAR(255) [FK to Videos.video_id], channel_title: VARCHAR(255))

Categories(category_id:VARCHAR(255) [PK], category_name:VARCHAR(255))

Trending(trending_id:VARCHAR(255) [PK], video_id:VARCHAR(255) [FK to Videos.video_id], trending_date:DATE)

Tags(tag_id:VARCHAR(255)[PK], video_id:VARCHAR(255) [FK to Videos.video_id], tag_name:VARCHAR(255))