GraphQL

1. Favourite Website: YouTube

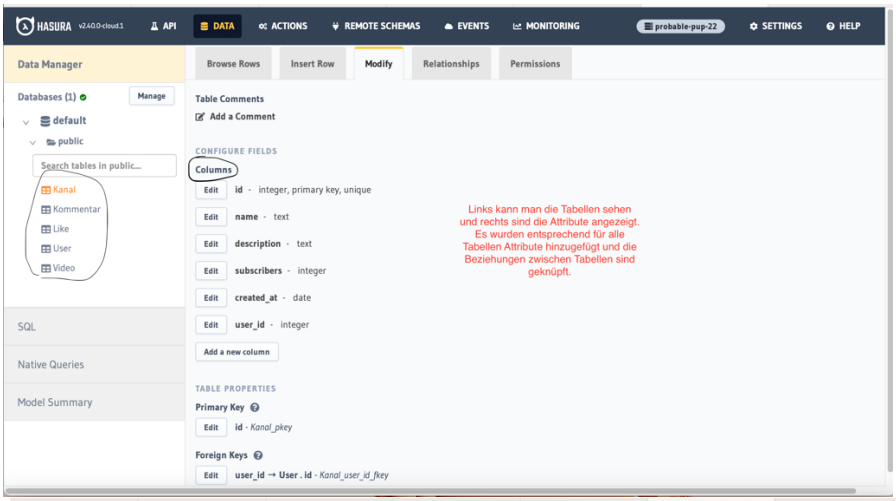
2. Entitäten und Attribute:

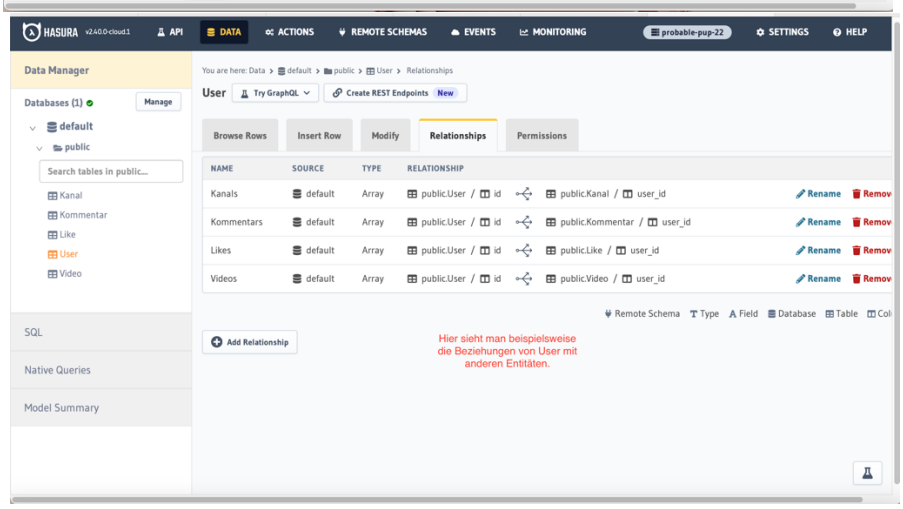
- **Video**: id, title, description, like, dislike, uploaded_at, user_id, kanal_id, url
- **Kanal**: id, name, description, created_at, User_id
- **Kommentar**: id, content, created_at, user_id, video_id
- **User:** id, username, password, email, created_at
- **Like:** id, user_id, video_id

3. Aktionen:

- Create a channel: Benötigte Entität: Channel, User
- Create a Video: Benötigte Entität: Video, Channel, User
- Video liken/disliken: Benötigte Entität: Video, Like, User
- Register a User: Benötigte Entitäten: User
- Fetch Videos: Benötigte Entitäten: Video, Channel, User
- Subscribe to a channel: Benötigte Entitäten: Channel, User

4., 5., 6. Aufgabe:





7. Aufgabe:

Mutation -> Änderung der Daten auf dem Server (Schreibeoperation)
Query -> Datenabfrage vom Server (Leseoperation)

Action- Create a Channel: mutation CreateChannel($name: String!, $description: String!, $user_id: Int!) { insert_channel(objects: {name: $name, description: $description, user_id: $user_id, subscribers: 0}) {
returning {
id
name
description
created_at
subscribers }}}

Action- Create a Video:
mutation CreateVideo($title: String!, $description: String!, $url: String!, $user_id: Int!, $channel_id: Int!) {
insert_video(objects: {title: $title, description: $description, url: $url, user_id: $user_id, channel_id: $channel_id, likes: 0, dislikes: 0}) { returning {
id
title
description
url
uploaded_at
likes
dislikes }}}

Erklärung zum obigen Code:
Mutation CreateChannel bzw. CreateVideo ist der Name der Mutation
$titel, $name,$description etc. repräsentieren die Parameter, währen titel, name, description etc. die Datenfelder repräsentieren. Das Ausrufezeichen ! kenntzeichnet diese Parameter als Pflichtfeld. Auch ist der Typ der Parameter vorgegeben. Insert_video bzw. Insert_channel ist der name der Operation. Hier ist es das Einfügen von Datensätzen in die channel- bzw. Video-Tabelle. „objects“ ist ein Objekt, welches die neuen Datensätze beinhaltet und entsprechend setzt. Das „returning“ gibt die Datenfelder innerhalb der Klammer zurück. So kann man sehen, ob die Mutation am Server erfolgreich war oder nicht.

Action- Like a Video: mutation LikeVideo($user_id: Int!,
$video_id: Int!) { insert_like(objects: {user_id: $user_id,
video_id: $video_id}) { returning {
id }}

Action- Register a User: mutation RegisterUser($username: String!, $email: String!,
$password: String!) { insert_user(objects: {username: $username, email: $email,
password: $password}) { returning {
id
username
email
created_at }}}

Action- Fetch Videos from Max Mustermann: query
FetchVideosByUser {
video(where: {user: {username: {_eq: "Max Mustermann"}}}) {
id
title
description
url
uploaded_at
likes dislikes user {
username
}
channel { name
subscribers }}}

Query-Name ist FetchVideosByUser mit der Bedingung, dass der User Max Mustermann als username hat. Von ihm werden die Felder id, title, description, url, uploaded_at, likes, dislikes abgefragt. Zusätzlich wird von ihm durch den Fremdschlüssel auf die Tabelle user sein username und von der Tabelle sein channel-name und subscribers abgefragt.

Action- Subscribe to a channel: mutation
SubscribeChannel($channel_id: Int!) {
update_channel(where: {id: {_eq: $channel_id}}, _inc: {subscribers: 1}) {
returning {
id
subscribers
}}}