



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих
комп'ютерних систем**

Лабораторна робота №1

з дисципліни

«Бази даних і засоби управління»

**Тема: «Створити БД в СУБД PostgreSQL
з допомогою конструктора PgAdmin 4»**

Виконав: студент III курсу

ФПМ групи KB-83

Пащенко Антон

Перевірив: Павловський В.І.

Київ – 2020

Лабораторна робота №1.
Створити БД "Відеохостинг" в СУБД PostgreSQL
з допомогою конструктора pgAdmin 4

Мета роботи: створити БД "Відеохостинг" та сформувати обмеження цілісності на значення даних.

Порядок виконання роботи

1. Розробити концептуальну модель вибраного предметного середовища. Концептуальна модель предметного середовища "Блог" наводиться в Додатку А до лабораторної роботи;
2. Розробити логічну модель (схему даних) БД;
3. Вивчити склад та правила роботи с СУБД PostgreSQL;
4. Створити в СУБД PostgreSQL БД "Блог", використовуючи конструктори таблиць та стовпчиків. Схема даних БД "Блог" наводиться в Додатку Б до лабораторної роботи. Перелік атрибутів наводиться в Додатку В до лабораторної роботи;
5. Сформувати обмеження цілісності, що забезпечують:
 - унікальність та обов'язковість вводу первинних ключів для всіх таблиць;
 - перевірка на відповідність зовнішніх ключів таблиць;
 - обмеження на значення даних для атрибутів і вивід відповідних повідомлень при їх порушенні;
 - обов'язковість вводу даних атрибутів;
 - сформувати маску вводу для атрибутів;
6. Заповнити створену БД даними (порядку 5-10 записів в кожній таблиці).
7. Вивести вміст таблиць створеної БД.

Зміст звіту

1. Склад СУБД PostgreSQL;
2. Опис предметної галузі;
3. Концептуальна модель предметної області;
4. Логічна модель БД;
5. Список обмежень цілісності в термінах СУБД PostgreSQL;
6. Представлення БД в pgAdmin 4.

Опис предметної галузі

При проектуванні бази даних “Відеохостинг” можна виділити такі сутності: Відео (Video), Категорія (Category), Користувач (User), Паспорт (Passport), Коментар (Comment).

До Категорії може належати декілька відео (один до багатьох).

Відео може мати 0 або більше коментарів (один до багатьох).

Користувач може написати 0 або більше коментарів (один до багатьох).

Користувач може подивитись багато відео і одне відео може бути переглянуте багатьма користувачами (багато до багатьох).

Користувач має паспорт (один до одного).

Додаток А. Концептуальна модель предметної області “Відеохостинг”

Нижче (Рисунок 1) наведена концептуальна модель предметної області "Відеохостинг"

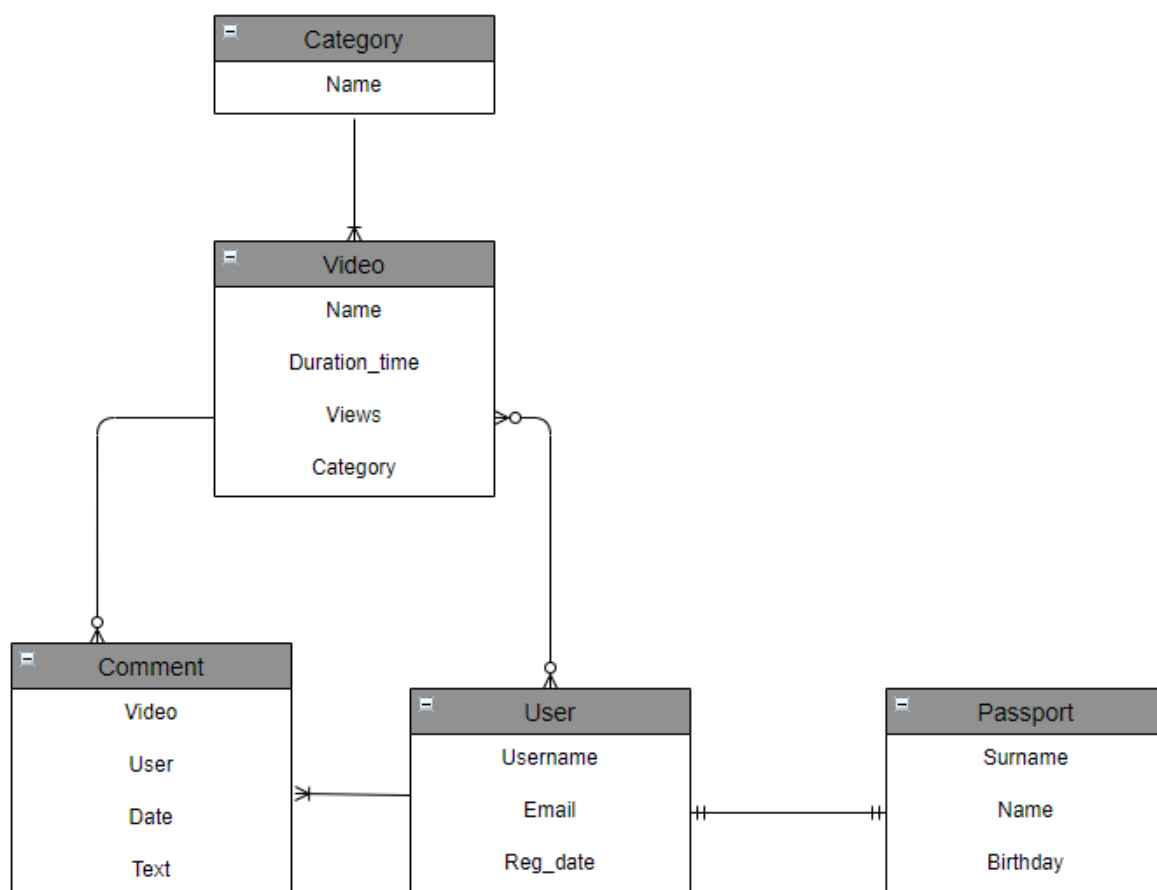


Рисунок 1 - Концептуальна модель предметної області “Відеохостинг”

Таблиці схеми бази даних відповідають 1НФ тому, що всі їх атрибути прості і містять лише скалярні значення.

Таблиці схеми бази даних відповідають 2НФ тому, що вони відповідають 1НФ та кожний їх неключовий атрибут залежить від первинного ключа, а не від його частини.

Таблиці схеми бази даних відповідають 3НФ тому, що вони відповідають 2НФ і всі їх атрибути нетранзитивно залежні від первинного ключа.

Додаток Б. Структура БД “Відеохостинг”

Нижче (Рисунок 2) наведено структуру БД предметної області "Відеохостинг"

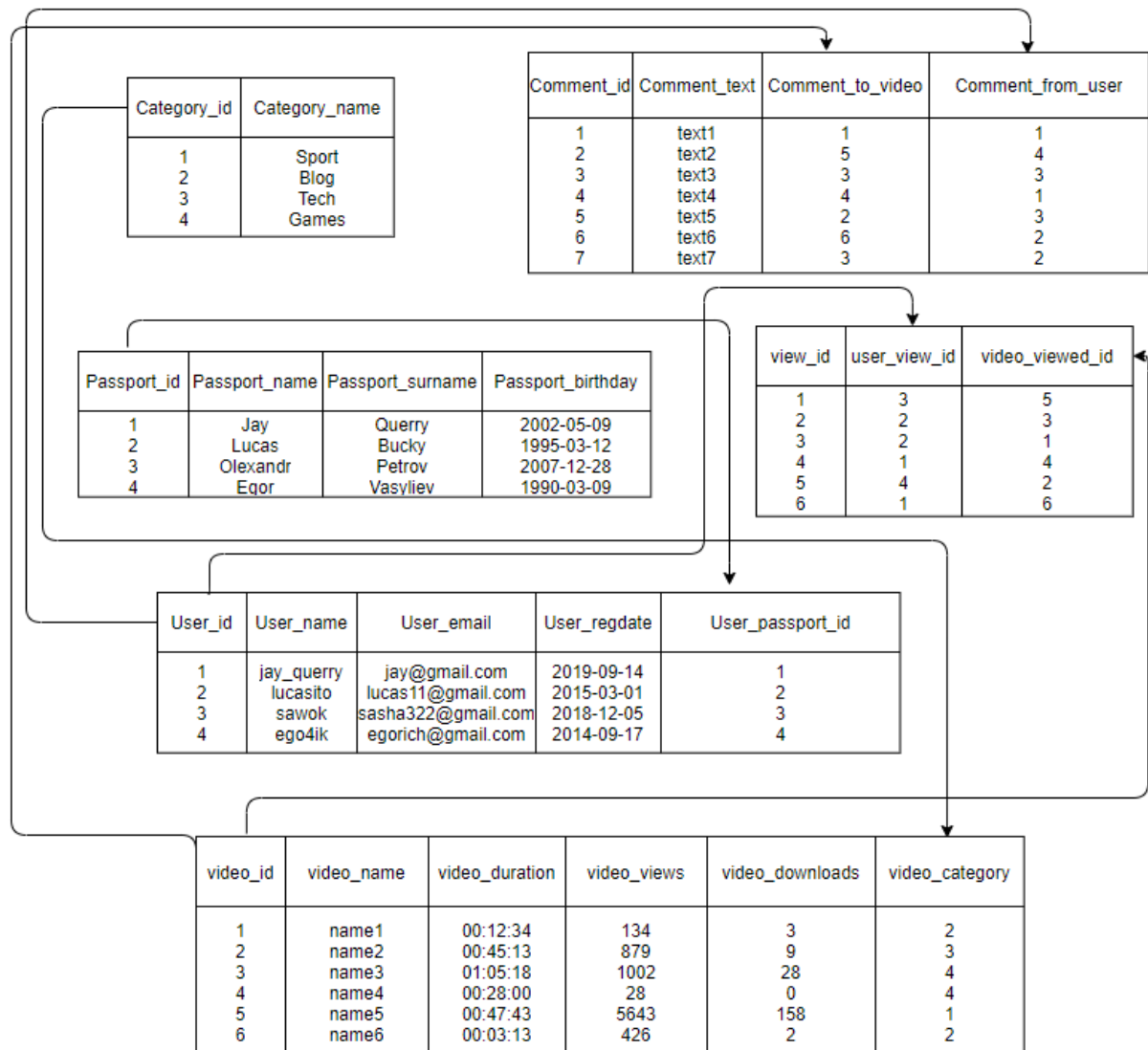


Рисунок 2 - Структура БД “Відеохостинг”

Додаток В. Опис структури БД “Відеохостинг”

ВІДНОШЕННЯ	АТРИБУТ	ТИП(Розмір)
Відношення “Category” Містить інформацію про категорію відео	Category_id – унікальний ID категорії в БД Category_name – назва категорії	Числовий Текстовий(50)
Відношення “Video” Містить інформацію про відео	Video_id - унікальний ID відео в БД Video_name – назва відео Video_views – кількість переглядів Video_duration – тривалість відео Video_downloads – кількість завантажень VideoCategory_id – ID категорії до якої належить відео	Числовий Текстовий(50) Числовий Час Числовий Числовий
Відношення “Viewed_playlist” Вміщує інформацію про переглянуті користувачами відео	Read_id – унікальний ID перегляд відео статті в БД UserView_id – ID користувача, який переглянув відео VideoViewed_id – ID відео, яке було переглянуто	Числовий Числовий Числовий
Відношення “User” Вміщує інформацію про користувача в хостингу	User_id – унікальний ID користувача User_login – логін користувача User_email – e-mail користувача User_reg_date– дата реєстрації користувача User_passport_id – ID паспорту	Числовий Текстовий(50) Текстовий(50) Дата
Відношення “Comment” Вміщує в собі інформацію про коментарі	Comment_id – унікальний ID коментаря Comment_text – текст коментаря Comment_date – дата написання коментаря UserComment_id – ID автора коментаря VideoComment_id – ID відео до якого написано відгук	Числовий Текст Дата Числовий Числовий
Відношення “Passport” Вміщує в собі інформацію про коментар про паспорт користувача	Passport_id – унікальний ID паспорта Passport_name – ім’я в паспорті Passport_surname – прізвище в паспорті Passport_birthday – дата народження	Числовий Текстовий(50) Текстовий(50) Дата

Зв'язки між category і video та user і passport при переході від концептуальної моделі до логічної залишаються такими ж самими. Для реалізації зв'язку багато до багатьох між user та video ми створюємо проміжну таблицю viewed_playlist колонками якої мають бути відповідні зовнішні ключі що посилаються на id таблиць user та video. Зв'язки один до багатьох між comment і video та comment і user реалізуються завдяки зовнішнім ключам в таблиці comment які посилаються на id video та user.

Додаток Г. Логічна модель БД “Відеохостинг” (засобами SqlDBM)

Нижче (Рисунок 3) наведено Логічна модель предметної області "Відеохостинг"

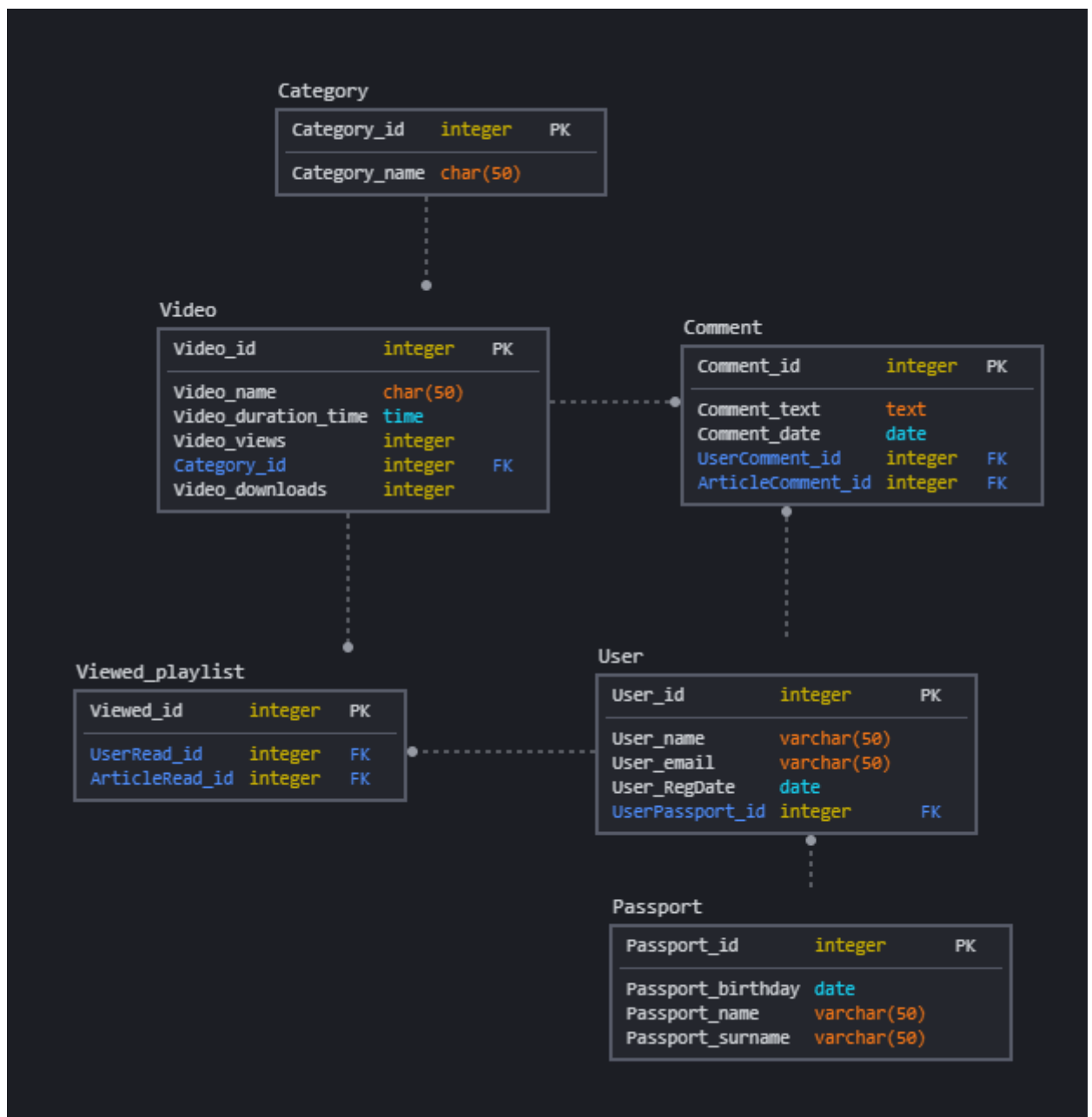
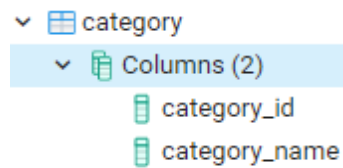


Рисунок 3 - Логічна модель БД “Відеохостинг”

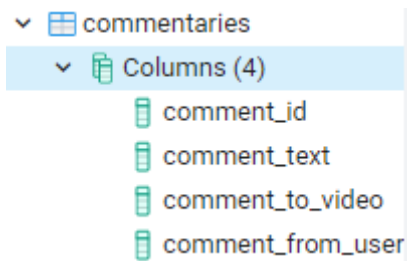
Додаток Д. Структура БД “Блог” в pgAdmin 4

Таблиця Category



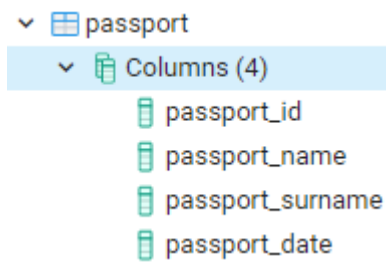
category
Columns (2)
category_id
category_name

Таблиця Commentaries



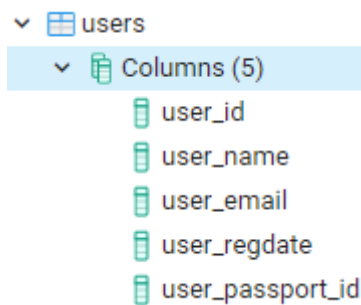
commentaries
Columns (4)
comment_id
comment_text
comment_to_video
comment_from_user

Таблиця Passport



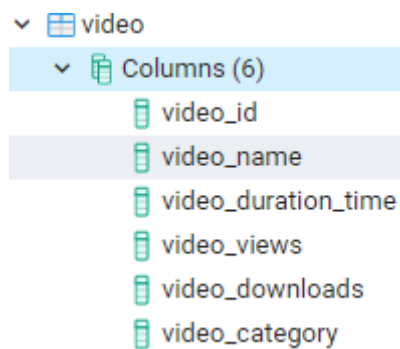
passport
Columns (4)
passport_id
passport_name
passport_surname
passport_date

Таблиця Users



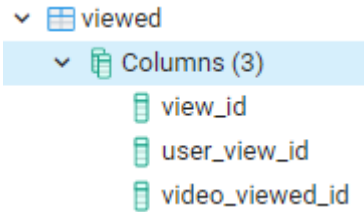
users
Columns (5)
user_id
user_name
user_email
user_regdate
user_passport_id

Таблиця Video



video
Columns (6)
video_id
video_name
video_duration_time
video_views
video_downloads
video_category

Таблица Viewed



▼ viewed

▼ Columns (3)

- view_id
- user_view_id
- video_viewed_id

Таблица Category

```
5 CREATE TABLE public.viewed
6 (
7     view_id integer NOT NULL,
8     user_view_id integer NOT NULL,
9     video_viewed_id integer NOT NULL,
10    CONSTRAINT view_idpk PRIMARY KEY (view_id),
11    CONSTRAINT user_view_idfk FOREIGN KEY (user_view_id)
12        REFERENCES public.users (user_id) MATCH SIMPLE
13        ON UPDATE CASCADE
14        ON DELETE CASCADE,
15    CONSTRAINT video_viewed_idfk FOREIGN KEY (video_viewed_id)
16        REFERENCES public.video (video_id) MATCH SIMPLE
17        ON UPDATE CASCADE
18        ON DELETE CASCADE
19 )
20
21 TABLESPACE pg_default;
22
23 ALTER TABLE public.viewed
24     OWNER to postgres;
```

Таблица Commentaries

```
5 CREATE TABLE public.commentaries
6 (
7     comment_id integer NOT NULL,
8     comment_text text COLLATE pg_catalog."default" NOT NULL,
9     comment_to_video integer NOT NULL,
10    comment_from_user integer NOT NULL,
11    CONSTRAINT comment_idpk PRIMARY KEY (comment_id),
12    CONSTRAINT comment_from_user_idfk FOREIGN KEY (comment_from_user)
13        REFERENCES public.users (user_id) MATCH SIMPLE
14        ON UPDATE CASCADE
15        ON DELETE CASCADE,
16    CONSTRAINT comment_to_video_idfk FOREIGN KEY (comment_to_video)
17        REFERENCES public.video (video_id) MATCH SIMPLE
18        ON UPDATE CASCADE
19        ON DELETE CASCADE
20 )
21
22 TABLESPACE pg_default;
23
24 ALTER TABLE public.commentaries
25     OWNER to postgres;
```


Таблица Passport

```
5 CREATE TABLE public.passport
6 (
7     passport_id integer NOT NULL,
8     passport_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
9     passport_surname character varying(50) COLLATE pg_catalog."default" NOT NULL,
10    passport_date date NOT NULL,
11    CONSTRAINT passport_id_pk PRIMARY KEY (passport_id)
12 )
13
14 TABLESPACE pg_default;
15
16 ALTER TABLE public.passport
17     OWNER to postgres;
```

Таблица Users

```
5 CREATE TABLE public.users
6 (
7     user_id integer NOT NULL,
8     user_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
9     user_email character varying(50) COLLATE pg_catalog."default" NOT NULL,
10    user_regdate date NOT NULL,
11    user_passport_id integer NOT NULL,
12    CONSTRAINT user_id_pk PRIMARY KEY (user_id),
13    CONSTRAINT user_userspassport_id_key UNIQUE (user_passport_id),
14    CONSTRAINT user_passport_idfk FOREIGN KEY (user_passport_id)
15        REFERENCES public.passport (passport_id) MATCH SIMPLE
16        ON UPDATE CASCADE
17        ON DELETE CASCADE
18 )
19
20 TABLESPACE pg_default;
21
22 ALTER TABLE public.users
23     OWNER to postgres;
```

Таблица Video

```
5 CREATE TABLE public.video
6 (
7     video_id integer NOT NULL,
8     video_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
9     video_duration_time time without time zone NOT NULL,
10    video_views integer,
11    video_downloads integer,
12    video_category integer NOT NULL,
13    CONSTRAINT video_id_pk PRIMARY KEY (video_id),
14    CONSTRAINT video_category_idfk FOREIGN KEY (video_category)
15        REFERENCES public.category (category_id) MATCH SIMPLE
16        ON UPDATE CASCADE
17        ON DELETE CASCADE
18 )
19
20 TABLESPACE pg_default;
21
22 ALTER TABLE public.video
23     OWNER to postgres;
```

Таблица Viewed

```
5 CREATE TABLE public.viewed
6 (
7     view_id integer NOT NULL,
8     user_view_id integer NOT NULL,
9     video_viewed_id integer NOT NULL,
10    CONSTRAINT view_idpk PRIMARY KEY (view_id),
11    CONSTRAINT user_view_idfk FOREIGN KEY (user_view_id)
12        REFERENCES public.users (user_id) MATCH SIMPLE
13        ON UPDATE CASCADE
14        ON DELETE CASCADE,
15    CONSTRAINT video_viewed_idfk FOREIGN KEY (video_viewed_id)
16        REFERENCES public.video (video_id) MATCH SIMPLE
17        ON UPDATE CASCADE
18        ON DELETE CASCADE
19 )
20
21 TABLESPACE pg_default;
22
23 ALTER TABLE public.viewed
24     OWNER to postgres;
```

Вміст таблиць в pgAdmin 4

Таблиця Category

	category_id [PK] integer	category_name character varying (50)
1	1	Sport
2	2	Blog
3	3	Tech
4	4	Games

Таблиця Comment

	comment_id [PK] integer	comment_text text	comment_to_video integer	comment_from_user integer
1	1	text1	1	1
2	2	text2	5	4
3	3	text3	3	3
4	4	text4	4	1
5	5	text5	2	3
6	6	text6	6	2
7	7	text7	3	2

Таблиця Passport

	passport_id [PK] integer	passport_name character varying (50)	passport_surname character varying (50)	passport_date date
1	1	Jay	Query	2002-05-09
2	2	Lucas	Bucky	1995-03-12
3	3	Olexandr	Petrov	2007-12-28
4	4	Egor	Vasyliiev	1990-03-09

Таблиця User

	user_id [PK] integer	user_name character varying (50)	user_email character varying (50)	user_regdate date	user_passport_id integer
1	1	jay_query	jay@gmail.com	2019-09-14	1
2	2	lucasito	lucas11@gmail.com	2015-03-01	2
3	3	sawok	sasha322@gmail.com	2018-12-05	3
4	4	ego4ik	egorich@gmail.com	2014-09-17	4

Таблица Video

	video_id [PK] integer	video_name character varying (50)	video_duration_time time without time zone	video_views integer	video_downloads integer	video_category integer
1	1	Name_1	00:12:34	134	3	2
2	2	Name_2	00:45:13	879	9	3
3	3	Name_3	01:05:18	1002	28	4
4	4	Name_4	00:28:00	28	0	4
5	5	Name_5	00:47:43	5643	158	1
6	6	Name_6	00:03:13	426	2	2

Таблица Viewed

	view_id [PK] integer	user_view_id integer	video_viewed_id integer
1	1	3	5
2	2	2	3
3	3	2	1
4	4	1	4
5	5	4	2
6	6	1	6