



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: *«Проектування бази даних та ознайомлення з базовими  
операціями СУБД PostgreSQL»*

Виконав: студент 3 курсу

ФПМ групи КВ-83

Хаустович Олександр

Перевірів: Павловський В.І.

**Мета роботи:** здобуття вмінь проектування бази даних та практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

**Завдання:**

1. Розробити модель «сутність-зв'язок» предметної галузі, обраної студентом самостійно, відповідно до пункту «Вимоги до ER-моделі»;
2. Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL;
3. Виконати нормалізацію схеми бази даних до третьої нормальної форми (3НФ);
4. Ознайомитись із інструментарієм PostgreSQL та pgAdmin 4 та внести декілька рядків даних у кожен з таблиць засобами pgAdmin 4.

**Варіант (предметна область):** база даних для музичного стрімінгового сервісу.

## Звіт

В музичних сервісах користувачі можуть слухати пісні як окремо, так і в плей листах. Плей листи – це збірки пісень створені користувачами або системою. Пісні можуть знаходитись в декількох плей листах одночасно. Також кожна пісня входить в альбом. Альбом – збірка пісень, яку формує автор пісень, групує композиції по жанру/стилю/настрою/ідеї. Пісня може знаходитися лише в одному альбомі. У кожного альбому є один або декілька авторів. У автора може не бути альбомів (якщо автор тільки починає писати пісні) або може бути довільна кількість. У кожного користувача є відповідний рахунок в банку, з якого він щомісяця платить за користування сервісом.

1. В концептуальній моделі предметної області "Музичний стрімінговий сервіс" (Рисунок 1) виділяються наступні сутності та зв'язки між ними:

- Сутність «User» з атрибутами: id, name, email, age;
- Сутність «Playlist» з атрибутами: id, name, is\_explicit, creator;
- Сутність «Song» з атрибутами: id, name, is\_explicit;
- Сутність «Album» з атрибутами: id, name, is\_explicit, publication\_date;
- Сутність «Author» з атрибутами: id, name, age, subscribers.

Між сутностями «User» та «Playlist» зв'язок «uses» типу N:M, тому що юзер може використовувати 0 і більше плейлистів, і плейлист можуть використовувати 0 і більше юзерів.

Між сутностями «User» та «Song» зв'язок «listens» типу N:M, тому що юзер може слухати 0 і більше пісень, і пісню можуть слухати 0 і більше юзерів.

Між сутностями «Playlist» та «Song» зв'язок «contains» типу N:M, тому що в плей листі може бути 0 і більше пісень, і пісня може бути в 0 і більше плей листах.

Між сутностями «Song» та «Album» зв'язок «contains» типу 1:N , тому що в альбомі може бути 1 і більше пісень, але пісня може бути лише в одному альбомі.

Між сутностями «Album» та «Author» зв'язок «publishes» типу N:N , тому що у автора може бути 0 і більше альбомів, а у альбому може бути 1 і більше авторів.

Між сутностями «User» та «Bank\_account» зв'язок «has» типу 1:1 , тому що одному користувачу відповідає один банківський рахунок і навпаки.

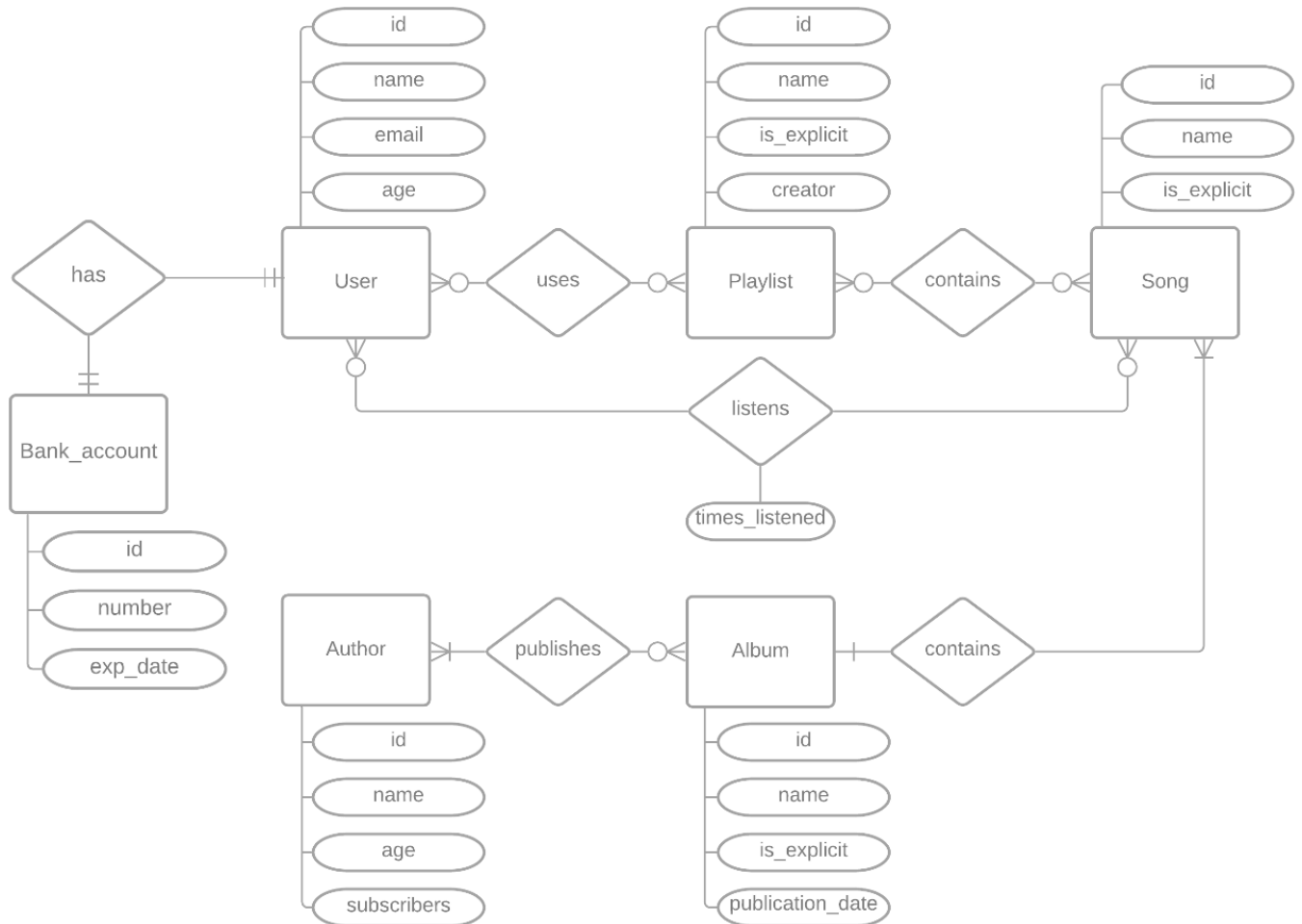


Рисунок 1 — ER-модель предметної області "Музичний стрімінговий сервіс".  
Використано сервіс Lucidchart

1. Схема бази даних PostgreSQL на основі ER-моделі предметної області "Музичний стрімінговий сервіс" (Рисунок 2).

Сутність «User» було перетворено в таблицю «User».

Сутність «Playlist» було перетворено в таблицю «Playlist».

Зв'язок «uses» типу N:M призвів до появи таблиці «user\_uses\_playlist».

Сутність «Bank\_account» було перетворено в таблицю «Bank\_account».

Зв'язок «has» типу 1:1 призвів до появи в таблиці «Bank\_account» додаткового поля user\_id, що є зовнішнім ключем до поля user\_id таблиці «User» та допускає тільки унікальні значення.

Сутність «Song» було перетворено в таблицю «Song».

Зв'язок «contains» типу призвів до появи таблиці «playlist\_contains\_song».

Зв'язок «listens» типу N:M призвів до появи таблиці «user\_listens\_song».

Сутність «Album» було перетворено в таблицю «Album».

Зв'язок «contains» типу 1:N призвів до появи в таблиці «Song» додаткового поля album\_id.

Сутність «Author» було перетворено в таблицю «Author».

Зв'язок «publishes» типу N:M призвів до появи таблиці «author\_publishes\_album».

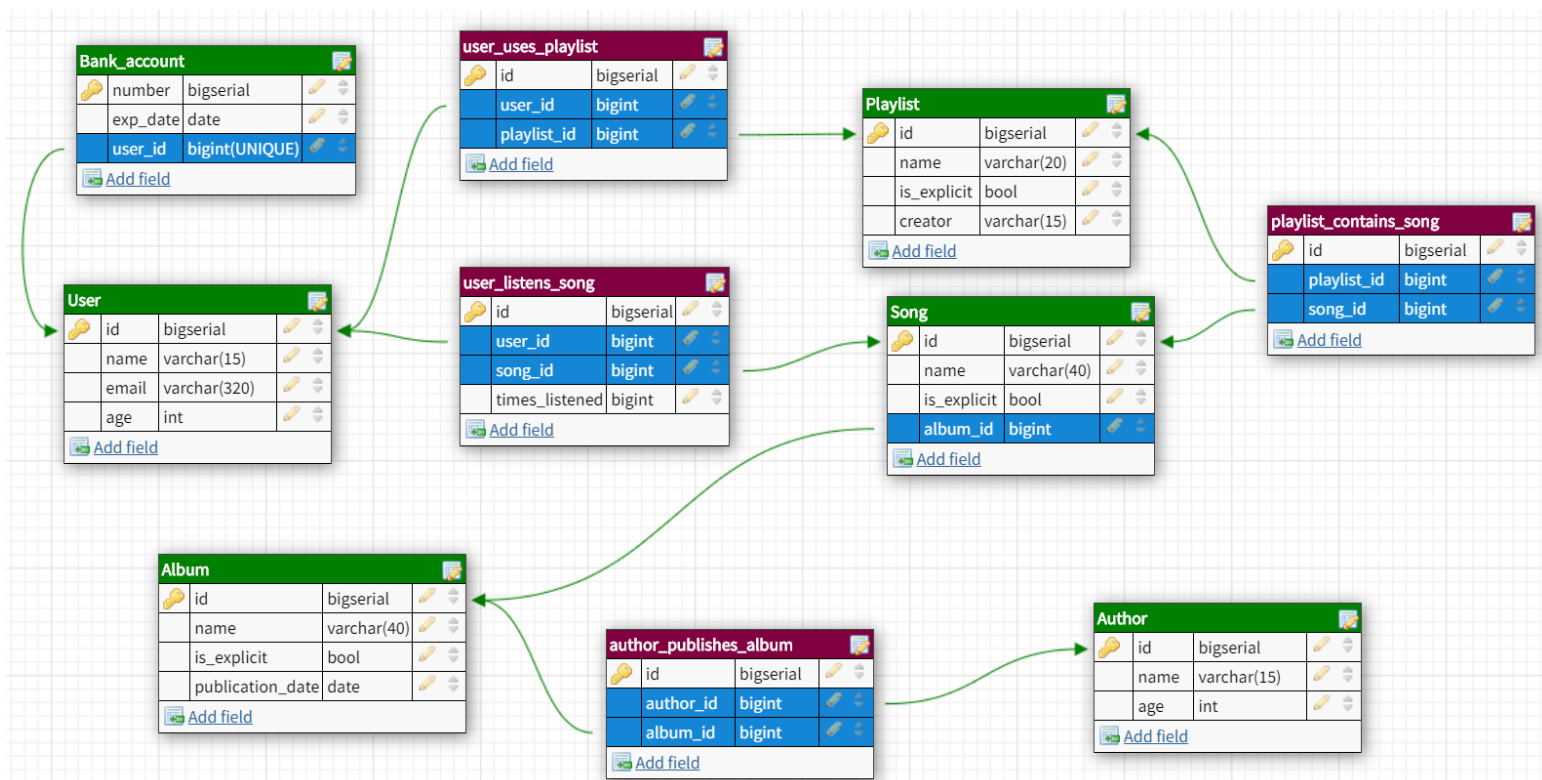











Рисунок 2 — Схема бази даних PostgreSQL на основі ER-моделі предметної області "Музичний стрімінговий сервіс".

*Примітка.* При побудові схеми БД використано сервіс Dbdesigner

Таблиця	Атрибути	Тип
User – таблиця для сутності User. Зберігає дані про користувачів	id – унікальний номер користувача. Не допускає NULL. name – ім'я користувача. Не допускає NULL.	bigserial character varying(15) character varying(320)

	email – електронна пошта користувача. Не допускає NULL та є унікальним. age – вік користувача. Не допускає NULL.	integer
Bank_account – таблиця для сутності Bank_account. Зберігає дані про банківський рахунок користувача.	 number – унікальний номер банківського рахунку. Не допускає NULL. exp_date – дійсний до. Не допускає NULL. user_id – унікальний номер власника рахунку. Не допускає NULL та є <b>унікальним</b> .	character varying(10)  date bigint
Playlist – таблиця для сутності Playlist. Зберігає дані про плей листи.	 id – унікальний номер плей листа. Не допускає NULL. name – ім'я плей листа. Не допускає NULL. is_explicit – чи є відвертий контент. Не допускає NULL. creator-ім'я автора плей листа.	bigserial character varying(20) Boolean character varying(15)
Song – таблиця для сутності Song. Зберігає дані про пісні.	 id – унікальний номер пісні name – назва пісні. Не допускає NULL. is_explicit – чи є відвертий контент. Не допускає NULL. album_id – id альбому, до якого належить пісня. Не допускає NULL.	bigserial character varying(40) boolean bigint
Album – таблиця для сутності Album. Зберігає дані про альбоми.	 id – унікальний номер альбому. Не допускає NULL. name – назва альбому. Не допускає NULL. is_explicit – чи є відвертий контент. Не допускає NULL. publication_date – дата публікації альбому. Не допускає NULL.	bigserial character varying(40) boolean date
Author – таблиця для сутності Author. Зберігає дані про авторів.	 id – унікальний номер автора name – ім'я автора. Не допускає NULL. age – вік автора.	bigserial character varying(15) integer
user_uses_playlist – містить інформацію про те, яким плейлистом користувався користувач.	 id – унікальний номер зв'язку. Не допускає NULL. user_id – унікальний номер користувача, який використовує плейлист. Не допускає NULL. playlist_id – унікальний номер, плей листа, який використовує користувач. Не допускає NULL.	bigserial  big  bigint
user_listens_song – містить інформацію про те, яку пісню слухав користувач.	 id – унікальний номер зв'язку. Не допускає NULL. user_id – унікальний номер користувача, що слухає пісню. Не допускає NULL. song_id – унікальний номер пісні, яку слухає користувач. Не допускає NULL. times_listened – кількість прослуховувань конкретної пісні конкретним користувачем. За замовчуванням 0.	bigserial  big  bigint  bigint
playlist_contains_song – містить інформацію про те, в якому альбомі зберігається пісня.	 id – унікальний номер зв'язку. Не допускає NULL. playlist_id – унікальний номер плей листа. Не допускає NULL. song_id – унікальний номер пісні. Не допускає NULL.	bigserial  bigint  bigint
author_publishes_album – містить інформацію про те, хто автор(-и) альбому.	 id – унікальний номер зв'язку. Не допускає NULL. author_id – унікальний номер автора альбому. Не допускає NULL. album_id – унікальний номер альбому. Не допускає NULL.	bigserial  bigint  bigint

## 2. Нормалізація БД до третьої нормальної форми

У всіх таблицях бази даних атрибути зберігають лише одне значення та кожен запис унікальний, тому база даних вже нормалізована до першої форми.

Оскільки у таблицях БД не використовуються композитні ключі, то база даних нормалізована до другої форми.

Перевіримо всі таблиці на наявність транзитивних зв'язків:

a) «User»:

- id → name, email, age;
- id → name;
- id → email;
- id → age;
- name ↔ email;
- email ↔ age;
- age ↔ name.

b) «user\_uses\_playlist»:

- id → user\_id, playlist\_id;
- id → user\_id;
- id → playlist\_id;
- user\_id ↔ playlist\_id.

c) «Playlist»:

- id → name, is\_explicit, creator;
- id → is\_explicit;
- id → creator;
- name ↔ is\_explicit;
- is\_explicit ↔ creator;
- creator ↔ name.

d) «Song»:

- id → name, is\_explicit, album\_id;
- id → is\_explicit;
- id → album\_id;
- id → name;
- name ↔ is\_explicit;
- is\_explicit ↔ album\_id;
- album\_id ↔ name.

e) «user\_listens\_song»:

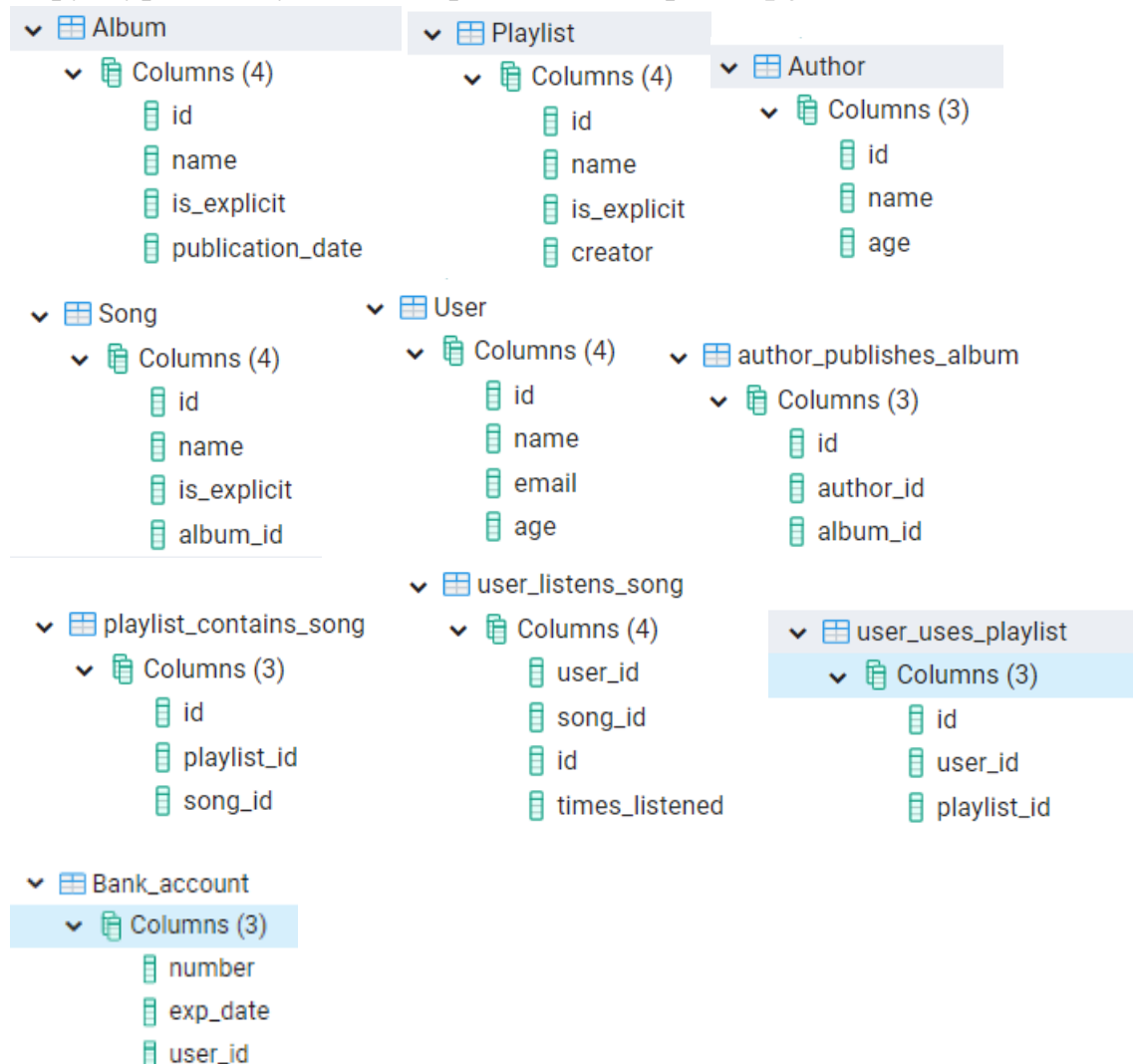
- id → user\_id, song\_id, times\_listened;
- id → user\_id;
- id → song\_id;
- id → times\_listened;
- user\_id ↔ playlist\_id;

- playlist\_id  $\leftrightarrow$  times\_listened;
- times\_listened  $\leftrightarrow$  user\_id.
- f) «playlist\_contains\_song»:
  - id  $\rightarrow$  song\_id, playlist\_id
  - id  $\rightarrow$  song\_id
  - id  $\rightarrow$  playlist\_id
  - song\_id  $\leftrightarrow$  playlist\_id
- g) «Album»:
  - id  $\rightarrow$  name, is\_explicit, publication\_date
  - id  $\rightarrow$  is\_explicit
  - id  $\rightarrow$  publication\_date
  - id  $\rightarrow$  name
  - name  $\leftrightarrow$  is\_explicit
  - is\_explicit  $\leftrightarrow$  publication\_date
  - publication\_date  $\leftrightarrow$  name
- h) «author\_publishes\_album»:
  - id  $\rightarrow$  album\_id, author\_id
  - id  $\rightarrow$  album\_id
  - id  $\rightarrow$  author\_id
  - album\_id  $\leftrightarrow$  author\_id
- i) «Author»:
  - id  $\rightarrow$  name, age
  - id  $\rightarrow$  name
  - id  $\rightarrow$  age
  - name  $\leftrightarrow$  age

В таблицях немає транзитивних зв'язків – отже вона нормалізована до третьої форми.

### 3. Створення БД в PostgreSQL з використанням pgAdmin4

#### Структура БД Музичний стрімінговий сервіс в pgAdmin4



User:

```
1 -- Table: public.User
2
3 -- DROP TABLE public."User";
4
5 CREATE TABLE public."User"
6 (
7     id bigint NOT NULL DEFAULT nextval('"User_id_seq"'::regclass),
8     name character varying(15) COLLATE pg_catalog."default" NOT NULL,
9     email character varying(320) COLLATE pg_catalog."default" NOT NULL,
10    age integer NOT NULL,
11    CONSTRAINT "User_pkey" PRIMARY KEY (id),
12    CONSTRAINT "User_email_key" UNIQUE (email)
13 )
14
15 TABLESPACE pg_default;
16
17 ALTER TABLE public."User"
18     OWNER to postgres;
```

	id [PK] bigint	name character varying (15)	email character varying (320)	age integer
1	1	Alex	khaustovych.alex@gmail.com	19
2	2	Taras	tkachuk@gmail.com	19
3	3	Michael	topchenskiy@gmail.com	20
4	4	Dmytro	glomozdui@ukr.net	19
5	5	Vlad	whatislove@gmail.com	19
6	6	Artem	zasupalo228@gmail.com	19



Playlist:

```
1 -- Table: public.Playlist
2
3 -- DROP TABLE public."Playlist";
4
5 CREATE TABLE public."Playlist"
6 (
7     id bigint NOT NULL DEFAULT nextval('"Playlist_id_seq"'::regclass),
8     name character varying(20) COLLATE pg_catalog."default" NOT NULL,
9     is_explicit boolean NOT NULL,
10    creator character varying(15) COLLATE pg_catalog."default",
11    CONSTRAINT "Playlist_pkey" PRIMARY KEY (id)
12 )
13
14 TABLESPACE pg_default;
15
16 ALTER TABLE public."Playlist"
17     OWNER to postgres;
```

	id [PK] bigint	name character varying (20)	is_explicit boolean	creator character varying (15)
1	1	Best songs	true	Alex
2	2	Your taste	false	[null]

Song:

```
1 -- Table: public.Song
2
3 -- DROP TABLE public."Song";
4
5 CREATE TABLE public."Song"
6 (
7     id bigint NOT NULL DEFAULT nextval('"Song_id_seq"'::regclass),
8     name character varying(40) COLLATE pg_catalog."default" NOT NULL,
9     is_explicit boolean NOT NULL,
10    album_id bigint NOT NULL,
11    CONSTRAINT "Song_pkey" PRIMARY KEY (id),
12    CONSTRAINT "Song_album_id_fkey" FOREIGN KEY (album_id)
13        REFERENCES public."Album" (id) MATCH FULL
14        ON UPDATE CASCADE
15        ON DELETE CASCADE
16        NOT VALID
17 )
18
19 TABLESPACE pg_default;
20
21 ALTER TABLE public."Song"
22     OWNER to postgres;
```

	id [PK] bigint	name character varying (40)	is_explicit boolean	album_id bigint
1	1	Weast coast	false	4
2	2	Birds	false	4
3	3	Zero	false	4
4	4	Bad liar	false	4
5	5	He исправлюсь	false	1
6	6	Oh My My	false	2
7	7	A.I.	false	2
8	8	Lift Me Up	false	2
9	9	bellyache	false	3
10	10	ocean eyes	false	3
11	11	idontwannameyouanym	false	3
12	13	Don't	true	5
13	14	Axerao	false	6
14	15	Акрарбинка	true	7

## Album:

```
1 -- Table: public.Album
2
3 -- DROP TABLE public."Album";
4
5 CREATE TABLE public."Album"
6 (
7     id bigint NOT NULL DEFAULT nextval('"Album_id_seq"'::regclass),
8     name character varying(40) COLLATE pg_catalog."default" NOT NULL,
9     is_explicit boolean NOT NULL,
10    publication_date date NOT NULL,
11    CONSTRAINT "Album_pkey" PRIMARY KEY (id)
12 )
13
14 TABLESPACE pg_default;
15
16 ALTER TABLE public."Album"
17     OWNER to postgres;
```

	id [PK] bigint	name character varying (40)	is_explicit boolean	publication_date date
1	1	He исправлюсь	false	2020-09-10
2	2	Oh My My	false	2016-10-07
3	3	dont smile at me	false	2017-08-11
4	4	Origins	false	2018-11-09
5	5	Don't	true	2014-08-24
6	6	Axerao	false	2020-07-30
7	7	Аскарбинка	true	2019-03-04

## Author:

```
1 -- Table: public.Author
2
3 -- DROP TABLE public."Author";
4
5 CREATE TABLE public."Author"
6 (
7     id bigint NOT NULL DEFAULT nextval('"Author_id_seq"'::regclass),
8     name character varying(15) COLLATE pg_catalog."default" NOT NULL,
9     age integer,
10    CONSTRAINT "Author_pkey" PRIMARY KEY (id)
11 )
12
13 TABLESPACE pg_default;
14
15 ALTER TABLE public."Author"
16     OWNER to postgres;
```

	id [PK] bigint	name character varying (15)	age integer
1	1	Billie Eilish	18
2	2	Imagine Dragons	[null]
3	3	Ed Seeran	29
4	4	дора	20
5	5	OneRepublic	[null]
6	6	МЭЙБИ БЭЙБИ	24

## Bank\_account:

```
1 -- Table: public.Bank_account
2
3 -- DROP TABLE public."Bank_account";
4
5 CREATE TABLE public."Bank_account"
6 (
7     "number" character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     exp_date date NOT NULL,
9     user_id bigint NOT NULL,
10    CONSTRAINT "Bank_account_pkey" PRIMARY KEY ("number"),
11    CONSTRAINT "Bank_account_user_id_key" UNIQUE (user_id),
12    CONSTRAINT "Bank_account_user_id_fkey" FOREIGN KEY (user_id)
13        REFERENCES public."User" (id) MATCH FULL
14        ON UPDATE CASCADE
15        ON DELETE CASCADE
16 )
17
18 TABLESPACE pg_default;
19
20 ALTER TABLE public."Bank_account"
21     OWNER to postgres;
```

	number [PK] character varying (10)	exp_date date	user_id bigint
1	2354675463	2021-10-10	1
2	3456234375	2020-11-12	4
3	8473648574	2022-07-06	2
4	8574003364	2021-08-16	3
5	7466347586	2020-12-05	5
6	9688574633	2023-04-19	6

## user\_listens\_song:

```
1 -- Table: public.user_listens_song
2
3 -- DROP TABLE public.user_listens_song;
4
5 CREATE TABLE public.user_listens_song
6 (
7     user_id bigint NOT NULL,
8     song_id bigint NOT NULL,
9     id bigint NOT NULL DEFAULT nextval('user_listens_song_id_seq'::regclass),
10    times_listened bigint DEFAULT 0,
11    CONSTRAINT user_listens_song_pkey PRIMARY KEY (id),
12    CONSTRAINT user_listens_song_song_id_fkey FOREIGN KEY (song_id)
13        REFERENCES public."Song" (id) MATCH FULL
14        ON UPDATE CASCADE
15        ON DELETE CASCADE,
16    CONSTRAINT user_listens_song_user_id_fkey FOREIGN KEY (user_id)
17        REFERENCES public."User" (id) MATCH FULL
18        ON UPDATE CASCADE
19        ON DELETE CASCADE
20 )
21
22 TABLESPACE pg_default;
23
24 ALTER TABLE public.user_listens_song
25     OWNER to postgres;
```

	user_id bigint	song_id bigint	id [PK] bigint	times_listened bigint
1	1	2	17	0
2	3	4	18	3
3	2	7	19	4
4	1	6	20	2
5	3	1	21	6
6	4	10	22	8
7	2	5	23	3
8	5	11	24	1
9	5	14	25	4

user\_uses\_playlist:

```
1 -- Table: public.user_uses_playlist
2
3 -- DROP TABLE public.user_uses_playlist;
4
5 CREATE TABLE public.user_uses_playlist
6 (
7     id bigint NOT NULL DEFAULT nextval('user_uses_playlist_id_seq'::regclass),
8     user_id bigint NOT NULL,
9     playlist_id bigint,
10    CONSTRAINT user_uses_playlist_pkey PRIMARY KEY (id),
11    CONSTRAINT user_uses_playlist_playlist_id_fkey FOREIGN KEY (playlist_id)
12        REFERENCES public."Playlist" (id) MATCH FULL
13        ON UPDATE CASCADE
14        ON DELETE CASCADE,
15    CONSTRAINT user_uses_playlist_user_id_fkey FOREIGN KEY (user_id)
16        REFERENCES public."User" (id) MATCH FULL
17        ON UPDATE CASCADE
18        ON DELETE CASCADE
19 )
20
21 TABLESPACE pg_default;
22
23 ALTER TABLE public.user_uses_playlist
24     OWNER to postgres;
```

	id [PK] bigint	user_id bigint	playlist_id bigint
1	1	2	1
2	2	4	2
3	3	4	[null]

playlist\_contains\_song:

```
1 -- Table: public.playlist_contains_song
2
3 -- DROP TABLE public.playlist_contains_song;
4
5 CREATE TABLE public.playlist_contains_song
6 (
7     id bigint NOT NULL DEFAULT nextval('playlist_contains_song_id_seq'::regclass),
8     playlist_id bigint NOT NULL,
9     song_id bigint NOT NULL,
10    CONSTRAINT playlist_contains_song_pkey PRIMARY KEY (id),
11    CONSTRAINT playlist_contains_song_playlist_id_fkey FOREIGN KEY (playlist_id)
12        REFERENCES public."Playlist" (id) MATCH FULL
13        ON UPDATE CASCADE
14        ON DELETE CASCADE,
15    CONSTRAINT playlist_contains_song_song_id_fkey FOREIGN KEY (song_id)
16        REFERENCES public."Song" (id) MATCH FULL
17        ON UPDATE CASCADE
18        ON DELETE CASCADE
19 )
20
21 TABLESPACE pg_default;
22
23 ALTER TABLE public.playlist_contains_song
24     OWNER to postgres;
```

	id [PK] bigint	playlist_id bigint	song_id bigint
1	1	1	1
2	2	1	4
3	3	1	13
4	4	1	2
5	5	2	1
6	6	2	3
7	7	2	4
8	8	2	11
9	9	2	6

author\_publishes\_album:

```
1 -- Table: public.author_publishes_album
2
3 -- DROP TABLE public.author_publishes_album;
4
5 CREATE TABLE public.author_publishes_album
6 (
7     id bigint NOT NULL DEFAULT nextval('author_publishes_album_id_seq'::regclass),
8     author_id bigint NOT NULL,
9     album_id bigint NOT NULL,
10    CONSTRAINT author_publishes_album_pkey PRIMARY KEY (id),
11    CONSTRAINT author_publishes_album_album_id_fkey FOREIGN KEY (album_id)
12        REFERENCES public."Album" (id) MATCH FULL
13        ON UPDATE CASCADE
14        ON DELETE CASCADE,
15    CONSTRAINT author_publishes_album_author_id_fkey FOREIGN KEY (author_id)
16        REFERENCES public."Author" (id) MATCH FULL
17        ON UPDATE CASCADE
18        ON DELETE CASCADE
19 )
20
21 TABLESPACE pg_default;
22
23 ALTER TABLE public.author_publishes_album
24     OWNER to postgres;
```

	id [PK] bigint	author_id bigint	album_id bigint
1	1	1	3
2	2	2	4
3	3	3	5
4	4	4	1
5	5	6	1
6	6	5	2
7	7	6	6
8	8	6	7