МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Музична пошукова система

КУРСОВА РОБОТА

з дисципліни «Бази даних»

|  |  |
| --- | --- |
| **Керівник**:  к.т.н., доц. Завгородній В.В.  «Допущено до захисту»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (особистий підпис керівника)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ р.  Захищено з оцінкою  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка)  Члени комісії:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (особистий підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (особистий підпис) | **Виконавець**:  Гацан С.Ю.  студент ІІ курсу  групи ІК-01  signзалікова книжка  № ІК-0106  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (особистий підпис виконавця)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ р.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (розшифровка підпису)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (розшифровка підпису) |

**Київ 2021**

**Міністерство освіти та науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Кафедра інформаційних систем та технологій

Дисципліна Бази даних

Спеціальність Інформаційні системи та технології

Курс II Група IК-01 Семестр 3

**ЗАВДАННЯ**

**на курсову роботу студента**

Гацана Сергія Юрійовича

(прізвище, ім’я, по батькові)

**1.** **Тема роботи** Музична пошукова система\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2.** **Термін здачі студентом закінченої роботи** « 23 » грудня 2021 р.

**3.** **Вхідні дані до роботи** система управління базами даних MySQL\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4. Зміст розрахунково-пояснювальної записки** (перелік питань, які підлягають розробці)

зміст, вступ, теоретична частина (аналіз та опис предметної області, проектування бази даних), практична частина (створення бази даних, організація вибірки інформації з бази даних, заповнення таблиць реляційної бази даних, створення збережених процедур, створення тригерів, створення представлень), висновки, список використаної літератури, додатки

**5. Перелік графічного матеріалу** (з точним зазначенням обов’язкових креслень)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6**. **Дата видачі завдання** « 23 » вересня 2021 р.

**КАЛЕНДАРНИЙ ПЛАН-ГРАФІК**

**виконання курсової роботи**

**студентом**  Гацан С.Ю

(прізвище, ініціали)

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів роботи та питань, які мають бути розроблені відповідно до завдання | Термін  виконання | Примітки керівника про виконання завдань |
| 1 | Видача завдання | 23.09.21 |  |
| 2 | Аналіз та опис предметної області | 07.10.21 |  |
| 3 | Проектування бази даних | 28.10.21 |  |
| 4 | Створення бази даних | 18.11.21 |  |
| 5 | Організація вибірки інформації з бази даних | 09.12.21 |  |
| 6 | Оформлення пояснювальної записки | 16.12.21 |  |
| 7 | Здача курсової роботи | 23.12.21 |  |

**Дата видачі завдання** «23» вересня 2021 р.



**Завдання прийняв до виконання** \_\_\_\_\_\_\_\_\_\_\_ С.Ю. Гацан

(підпис)

**Керівник**\_\_\_\_\_\_\_\_\_\_ В.В. Завгородній

(підпис)

**ЗМІСТ**

**(ПЕРЕГЕНЕРИРОВАТЬ, ЧТОБЫ БЫЛО АВТОМАТОМ, А НЕ ВРУЧНУЮ)**

[ВСТУП 6](#_Toc833122209)

[1 ТЕОРЕТИЧНА ЧАСТИНА 8](#_Toc87891385)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ВСТУП** | | | | | 5 |
| **1 ТЕОРЕТИЧНА ЧАСТИНА** | | | | | 7 |
|  | 1.1 Аналіз та опис предметної області | | | | 7 |
|  |  | | 1.1.1 Опис предметної області | | 7 |
|  |  | | 1.1.2 Опис вхідних даних | | 9 |
|  |  | | 1.1.3 Опис вихідних даних | | 10 |
|  | 1.2 Проектування бази даних | | | | 10 |
|  |  | | 1.2.1 Інфологічна модель бази даних | | 10 |
|  |  | |  | 1.2.1.1 Опис сутностей | 12 |
|  |  | |  | 1.2.1.2. Опис атрибутів | 12 |
|  |  | |  | 1.2.1.3 Опис зв'язків | 14 |
|  |  | |  | 1.2.1.4 Діаграма "сутність-зв'язок" | 16 |
|  |  | | 1.2.2 Нормалізація таблиць при проектуванні бази даних | | 17 |
|  |  | | 1.2.3 Даталогічна модель бази даних | | 19 |
| **2 ПРАКТИЧНА ЧАСТИНА** | | | | | 23 |
|  | 2.1 Створення бази даних за допомогою MySQL | | | | 23 |
|  |  | 2.1.1 Створення бази даних | | | 23 |
|  |  | 2.1.2 Створення таблиць бази даних | | | 25 |
|  |  | 2.1.3 Створення діаграми бази даних | | | 27 |
|  |  | 2.1.4 Заповнення таблиць бази даних даними | | | 29 |
|  |  | 2.1.5 Створення збережених процедур | | | 30 |
|  |  | 2.1.6 Створення тригерів | | | 32 |
|  |  | 2.1.7 Створення представлень | | | 33 |
|  | 2.2 Організація вибірки інформації з бази даних | | | | 34 |
|  |  | 2.2.1 Проста вибірка даних | | | 35 |
|  |  | 2.2.2 Вибірка обчислюваних значень | | | 38 |
|  |  | 2.2.3 Вибірка значень з діапазону | | | 39 |
|  |  | 2.2.4 Вибірка з використанням шаблонів | | | 40 |
|  |  | 2.2.5 Угруповання даних при організації запитів | | | 41 |
|  |  | 2.2.6 Об’єднання таблиць | | | 41 |
| **ВИСНОВКИ** | | | | | 43 |
| **СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ** | | | | | 44 |
| **ДОДАТКИ** | | | | |  |
|  | **Додаток А.** Заповнення таблиць реляційної бази даних | | | | 46 |
|  | **Додаток Б.** Створення збережених процедур | | | | 52 |
|  | **Додаток В.** Створення тригерів | | | | 53 |
|  | **Додаток Г.** Створення представлень | | | | 55 |

# ВСТУП

*База даних* *-* це сукупність даних, яка об’єднана та організована відповідно до концепції, яка описує характеристику цих даних, взаємозв’язки між їх елементами. В загальному випадку базою даних вважають будь-який впорядкований набір даних.

Свою історію бази розпочали ще з 1960-х років, коли було розроблено першу БД - CODASYL. Пізніше, тобто у 1970-х рр. з’явились перші основи для реляційних моделей баз даних, які зацікавили лише наукові кола. 1980-ті рр. славляться появою перших комерційних реляційних БД Oracle та DB2. Ці бази були дуже успішними, адже вони дуже швидко почали витісняти з ринку попередні, мережеві та ієрархічні. В 1990-х рр. науковці звернули увагу на об’єктно-орієнтовні БД, які були потрібні в тих сферах, де використовувались комплексні дані. Починаючи з 2000-х рр. і до сьогодення починають з’являтись безкоштовні версії баз даних, які може навчитись використовувати кожен.

*Реляційна БД -* такий вид баз даних, який заснований на реляційній моделі, тобто на зв’язках; це сукупність відношень, що містять усю інформацію, яка повина зберігатися у БД.

*MySQL -* це вільна система керування базами даних, яка була розроблена компанією “TcX” для підвищення швидкодії обробки баз даних. На сьогоднішній день MySQL - одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різних мов програмування.

Можливості сервера MySQL:

* Простота у встановленні та використанні;
* Підтримується необмежена кількість користувачів, що одночасно працюють з БД;
* Кількість рядків у таблицях може досягати 50 млн;
* Висока швидкість виконання команд;
* Наявність простої та ефективної систем и безпеки.

*MySQL Workbench -* це графічний інтерфейс для проектування баз даних. Серед можливостей програми можна виділити наступні:

* Дозволяє наочно представити модель бази даних у графічному вигляді;
* Наочний і функціональний механізм установки зв’язків між таблицями, в тому числі “багато до багатьох” із створенням таблиці зв’язків;
* Reverse Engineering - відновлення структури таблиць з вже існуючої на сервері БД;
* Зручний редактор SQL запитів, що дозволяє відразу ж відправляти їх серверові і отримати відповідь у вигляді таблиці;
* Можливість редагування даних у таблиці в візуальному режимі.

В даній курсовій роботі була розроблена реляційна база даних “Музична пошукова система” за допомогою MySQL 8.0.27, в якій необхідно вирішити наступні завдання:

* Провести аналіз предметної області з описом вхідних та вихідних даних;
* Побудувати інфологічну моель бази даних (опис сутностей, атрибутів та зв’язків між сутностями);
* Провести нормалізацію таблиць при проектуванні бази даних;
* Побудувати даталогічну можель баз даних;
* За допомогою СУБД MySQL створити базу даних, таблиці, діаграму бази даних, збережені процедури, тригери та представлення;
* Заповнити таблиці необхідними даними;
* Виконати вибірку інформації з бази даних (просту, вибірку обчислювальних значень, з діапазону, з використанням шаблонів, при об’єднанні таблиць).

# 1 ТЕОРЕТИЧНА ЧАСТИНА

## Аналіх предметної області

## Опис предметної області

Основним призначенням інформаційної системи (ІС) є оперативне забезпечення користувача інформацією про зовнішній світ шляхом реалізації питально-вудповідного відношення.

Питально-відповідні відношення дозволяють виділити для ІС певний її фрагмент - *предметну область (ПрО)*, - який буде втілений в автоматизованій ІС.

Інформація про зовнішній світ подається в ІС у формі даних, що обмежує можливості змістовної інтерпретації інформації й конкретизує семантику її подання в ІС. Сукупність циї виділених для ІС даних, зв’язків між ними й операцій над ними утворить інформаційну й функціональну моделі предметної області, що описують її стан з певною точністю. Інформаційна й функціональна моделі ПрО є вхідними даними для процесу проектування БД.

Будь-яка організація потребує своєчасного доступу до інформації. Цінність інформацї в даний час дуже висока. Роль розпорядників інформації в сучасному світі найчастіше виконують бази даних. Бази даних забезпечують надійне зберігання інформації, у структурованому вигляді і своєчасний доступ до неї.

Практично будь-яка сучасно організація потребує базу даних, що задовольняє ті чи інші потреби зі зберігання, управління та адміністрування даних.

Тому ефективне функціонування сучасного підприємства неможливо без застосування інформаційних систем. Дана проблема актуальна як для великих підприємств, так і для підприємств середнього і навіть малого бізнесу.

Інформаційні системи мають ряд істотних відмінностей від стандартних прикладних програм. Залежно від предметної області інформаційні системи модуть сильно відрізнятися за своєю архітектурою і функцією.

При розробці реляційної бази даних “Музична пошукова система” було проведено дослідження предметної області. Основними операціями в досліджуваній предметній області є складання списку треків, їх авторів, належність пісень до альбомів, моніторинг користувачів та їхніх прав.

Треки є різних жанрів та належать до різних альбомів. Так як це пошукова система, то потрібно вести список користувачів та їх прав доступу до бази. Основною метою даної курсової роботи є автоматизація всього цього процесу, щоб скоротити час оператора на обробку даних та пошук потрібної інформації. Також задля спрощення взаємодії з базою даних, було створено графічний інтерфейс бази даних.

## Опис вхідних даних

Для кожного виду вхідних даних вказується форма подання, періодичніть надходження, джерело виникнення. Якщо вхідні дані зафіксовані у первинних документах, то вказується найменування документа, періодичність його надходження на обробку. Якщо вхідні дані є результатом розв’язання інших задач і зберігаються на машинних носіях, то вони описуються із зазнаенням найменування задачі (підпрограми), під час вирішення якої були одержані.

При розробці реляційної бази даних “Музична пошукова система” були виділені наступні вхідні дані:

* Інформація про пісні;
* Інформація про композиторів;
* Інформація про жанри;
* Інформація про користувачів;
* Інформація про альбоми;
* Інформація про тексти пісень.

## Опис вихідних даних

Для кожного виду вихідних даних вказується форма подання, періодичність видання, користувач інформації. Вихідні повідомлення залежно від конкретної задачі можуть бути подані у вигляжі набору даних на машинному носії - як проміжні дані для нагромадження, зберігання та використання під час роботи інших програм або розв’язання інших задач. У цьому випадку вони описуються із зазначенням найенування задачі (підпрограми), під час розв’язання якої будуть використовуватися.

Для вихідної інформації, що подається у формі документа, необхідно привести зразок форми документа та його опис, охарактеризувати порядок його оформлення.

Вихідними даними для бази даних “Музична пошукова система” є вихідні запити.

## Проектування бази даних

## Інфологічна модель бази даних

*Мета інфологічного проектування -*  забезпеення найбільш природних для людини способів збору і представлення тієї інформації, яку передбачається зберігати в створеній БД. Тому, інфологічну модель намагаються будувати за аналогією з природною мовою. Основними конструктивними елементами інфологічних можелей є сутності, зв’язки між ними та їх властивості [2-5].

Інфологічна можель створюється за результатими проведення досліджень ПрО.

*Інфологічна модель -* це опис майбутньої бази даних, поданий за допомогою природної мови, формул, графіків, діаграм, таблиць та інших засобів, зрозумілих як розробникам БД, так і звичайним користувачам. Призначення такої моделі полягає в адекватному описі процесів, інформаційних потоків, функцій система за допомогою загольнодоступної предметної області, консультантів, користувачів для обговорення моделі і внесення виправлень.

Створення інфологічної моделі є природним продовденням досліджень предметної області, але, на відміну від нього, є представленням БД з точки зору проектувальника (розробника). Наочність представлення такої моделі дозволяє експертам ПрО оцінити її точність і внести виправлення. Від правильності моделі залежить успіх подальшої розробки.

Важливо відзначити, зо створювана на цьому етапі модель повністю не залежить від фізичної реалізації майбутньої системи. У випадку з БД це означає, що зовсім неважливо, де фізично будуть зберігатися дані: на папері, в пам’яті комп’ютера і хто або що ці дані буде обробляти. У цьому випадку, коли структури даних не залежать від їх фізичної реалізації, а моделюються виходячи х їх смислового призначення, моделювання називається *семантичним*.

Існує кілька способів опису інфологічної моделі, проте на сьогодні одним з найбільш поширених підходів, що застосовуються в інфологічному моделюванні, є підхід, заснований на застосуванні діаграм “сутність-зв’язок” (ER - Entity Relationship).

Відносна простота і наочність опису предметної області дозволяє використовувати її в процесі діалогу з потенційними користувачами із самого початку інфологічного проектування. Побудова інфологічної моделі, як і будь-якої іншої моделі, є творчим процесом, тому єдиної методики її створення немає. Однак за будь-якого підходу до побудови моделі використовують три основних конструктивні елементи: сутність, атрибут та зв’язок.

## Опис сутностей

Базовими елементами моделі “сутність-зв’язок” є сутності. *Сутність (Entity) -*  це множина екземплярів реальних або абстрактних об’єктів (людей, подій, станів, ідей, предметів тощо), які володіють загальними атрибутами або характеристиками. Будь-який об’єкт системи може бути представлений тільки однією сутністю, яка повина відображати тип або клас об’єкта, а не його конкретний екземпляр.

Сутність можна визначити як об’єкт, подію або концепцію, інформація про яких повинна зберігатися. Сутності повинні мати найменування з чітким смисловим значенням, іменуватисяя іменником в однині, не мати “технічних” найменувань. Іменування сутності в однині полегшує надалі читання моделі. Кожна сутність може мати будь-яку кількість зв’язків з іншими сутностями моделі.

У відповідності з описом предметної області були отримані такі сутності:

* “Автор” - зберігається інформація про композиторів;
* “Альбом” - зберігається інформація про музичні альбоми;
* “Пісня” - зберігається інформація про музичні треки;
* “Жанр” - зберігається інформація про музичні жанри;
* “Пісня в альбомі” - зберігається інформація про належність пісні до альбому;
* “Тексти” - зберігається інформація про тексти пісень;
* “Користувач” - зберігається інформація про усіх користувачів пошукової системи;
* “Права доступу” - зберігається інформація про можливі права доступу користувачів.

## Опис атрибутів

*Атрибут (Attribute) -*  будь-яка характеристика сутності, значуща для розглянутої предметної області і призначена для кваліфікації, ідентифікації, класифікації, кількісної характеристики або вираження стану сутності. Атрибут є типом характеристик або властивостей, асоційованих із безліччю реальних або абстрактних об’єктів.

*Екземпляр атрибута -*  це певна зарактеристика окремого елемента множини. Екземпляр атрибута визначається типом характеристики і її значенням, так званим значенням атрибута. На діаграмі “сутність-зв’язок” атрибути асоціюються з конкретними сутностями. Таким чином, приклад сутності повинен мати єдине встановлене значення для асоційованого атрибута.

Реляційні таблиці можуть бути пов’язані одна з одною, отже, дані можуть отримуватись одночасно з декількох таблиць. Таблиці зв’язуються мід собою для зменшення обсягу БД. Зв’язок кожної пари таблиць забезпечується за допомогою полів, що мають однаковий формат та назву.

У відповідності з описом предметної області були виділені наступні атрибути у кожній сутності:

1. Таблиця *Автор (Author)* містить:

ID - унікальний код автора

Name - ім’я автора

1. Таблиця *Альбом (Album)* містить:

ID - унікальний код альбому

ID\_Author - код автора альбому

Name - назва альбому

Release\_year - рік випуску альбому

1. Таблиця *Пісня (Song)* містить:

ID - унікальний код пісні

Name - назва пісні

ID\_Author - унікальний код автора

ID\_Lyrics - код тексту пісні

ID\_Genre - унікальний код жанру

Release\_year - рік випуску

1. Таблиця *Жанр (Genre)* містить:

ID - унікальний код жанру

Name - назва жанру

1. Таблиця *Пісня в альбомі (Song in album)* містить:

ID - унікальний код зв’язку

ID\_Song - код пісні

ID\_Album - код альбому

1. Таблиця *Тексти (Lyrics)* містить:

ID - код тексту

URL - посилання на текст

1. Таблиця *Користувач (User)* містить:

ID - унікальний код користувача

Nickname - нік користувача

Email - пошта користувача

Password - пароль користувача

Phone - номер телефону користувача

Last\_Login - час останнього входу користувача

1. Таблиця *Права доступу (Access Rights)* містить:

ID - унікальний код пошуку

Name - назва права доступу

## Опис зв’язків

*Зв’язок* - це асоціювання двох і більше сутностей [10,11]. Якби призначенням реляційної бази даних було тільки збереження окремих, не пов’язаних між собою даних, то її структура могла бути дуже проста. Проте одна з основних вимог до органіхації бази даних - це забезпечення можливості відшукання одних сутностей за призначенням інших, для чого необхідно встановити між ними певні зв’язки.

*Зв’язок* - це асоціація між сутностями, за якої кожен екземпляр однієї сутності асоційований іх довільною (у тому числі нульовою) кількістю екземплярів другої сутності, і навпаки.

Зв’язок може додатково визначатися за допомогою зазначення ступеня або потужності (кількості екземплярів дочірньої сутності, які може породжувати кожен екземпляр батьківської сутності).

Зв’язок з логічної точки зору становить співвідношення мід сутностями, яке нерідко може бути виражене звичайними фразами, наприклад: “Клієнт розміщує вклад”, де іменниками є назви пов’язаних між собою сутностей.

Переважна більшість засобів проектування даних дозволяє створювати ER-діаграми візуально, зображаючи сутності і поєднуючи їх зв’язками за допомогою миші.

Інтерфейс таких засобів нерідко настільки простий, що дозволяє освоїти логічне проектування даних не тільки розробнику, але й звичнайному користувачеві, якщо такий бере участь у проектуванні даних як експерт в предметній області.

Слід зауважити, що на етапі логічного проектування можна лписаи поведінку СУБД у разі порушення правил цілісності, що визначаються даним зв’язком.

Взаємозв’язки між таблицями реляційної бази жаних можуть бути типізовані за такими основнии видами [5,6]:

* Відношення “один до одного” (1:1) означає, що кожному запису однієї таблиці відповідає тільки один запис в іншій таблиці;
* Відношення “один до багатьох” (1:Б) виникає, коли один запис взаємопов’язаний з багатьма іншими;
* Відношення “багато до одного” (Б:1) означає, що багато записів пов’язані з однією;
* Відношення “багато до багатьох” (Б:Б) виникає між двома таблицями в тих випадках, коли:
* Один запис із першої таблиці може бути пов’язаний більш ніж з одним записом із другої таблиці;
* Один запис з другої таблиці може бути пов’язаний більш ніж з одним записов з першої таблиці.

В курсовій роботі були використані наступні типи зв’язків між таблицям реляційної бази даних “Музична пошукова система” (табл. 1.1).

Таблиця 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер зв’язку** | **Головна таблиця** | **Дочірня таблиця** | **Тип зв’язку** |
| 1 | Автор | Пісня | 1:Б |
| 2 | Жанр | Пісня | 1:Б |
| 3 | Пісня | Пісня в альбомі | 1:1 |

Продовження таблиці 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | Альбом | Пісня в альбомі | 1:Б |
| 5 | Автор | Альбом | 1:Б |
| 6 | Право доступу | Користувач | 1:Б |
| 7 | Текст | Пісня | 1:Б |

Перший зв’язок “Автор - Пісня” вказує на те, що один автор може написати багато пісень.

Другий зв’язок “Жанр - Пісня” вказує на те, що до одного жанру може належати багато пісень.

Третій зв’язок “Пісня - Пісня в альбомі” вказує на те, що пісня може належати лише до одного альбому.

Четвертий зв’язок “Альбом - Пісня в альбомі” вказує на те, що в одному альбомі може бути багато пісень.

П’ятий зв’язок “Автор - Альбом” вказує на те, що у одного автора може бути багато альбомів.

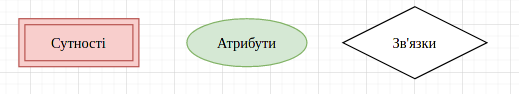
Шостий зв’язок “Право доступу - Користувач” вказує на те, що одне право доступу може бути у декількох користувачів.

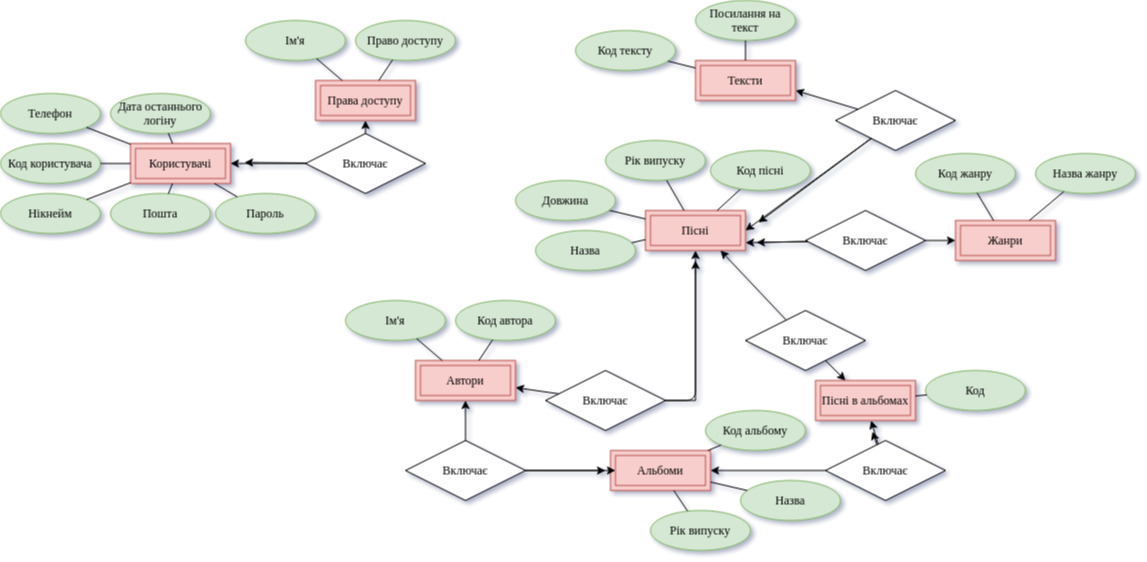
Сьомий зв’язок “Текст - Пісня” вказує на те, що один текст може бути в декількох записах пісень (у випадку декількох авторів у пісні буде декілька записів про цю пісню, і у кожного з них один текст).

* + - 1. Діаграма “сутність-зв’язок”

На рисунку 2.1 представлена ER-діаграма бази даних “Музична пошукова система”, на якій відображені всі сутності, їх атрибути та зв’язки між сутностями.

*Позначення:*



Рисунок 2.1 - ER-діаграма бази даних “Музична пошукова система”

## Нормалізація таблиць при проектуванні бази даних

*Нормалізація -* це розбиття таблиці на дві або більше, що володіють кращими властивостями при включенні, зміні і видаленні даних.

У теорії нормалізації існує п’ять нормальних форм (скорочено НФ) таблиць. Ці форми призначені для зменшення надлишкової інформації від першої до п’ятох нормальної форми. Тому кожна наступна НФ повинна задовольняти вимогами попередньої форми і деяким додатковим умовам.

Проведемо нормалізацію наявних сутностей.

Таблиця у першій НФ вимагає, щоб всі значення всіх атрибутів були атомарними, тобто неділимими. Іншими словами, кожен атрибут відношення повинен зберігати одне-єдине значення і не бути ні списком, ні набором певних значень.

Таблиця знаходиться в другій НФ, якщо кожен не ключовий атрибут повністю функціонально залежить від ключа. Також потрібно щоб таблиця вже належала до першої НФ.

Таблиця знаходиться в третій НФ, якщо вона задовольняє умовам другої НФ та кожне не ключове поле не мало транзитивних залежностей.

Іншими словами, щоб привести відношення до 3НФ, необхідно усунути функціональні залежності між не ключовими атрибутами відношення. Тобто дані, що зберігаються в таблиці, повинні залежати тільки від ключа.

При вирішенні практичних завдань в більшості випадків 3НФ є достатньою. Процес проектування реляційної бази даних, як правило, закінчується приведенням її до третьої нормальної форми.

Дана модель не потребує подальшого приведення до четвертої форми нормалізації.

Згідно до наведеної структури бази даних, наведеної вище, можна побачити, що вона вже знаходиться в першій нормальній формі, бо в ній немає полів, що не були б атомарними. Також жодна з таблиць не містить груп, що повторюються.

Таблиця вже знаходиться в другій нормальній формі, бо всі не ключові поля таблиці залежать від первинного ключа, тобто, первинний ключ однозначно визначає кожен запис.

Таблиця вже знаходиться в третій нормальній формі, бо в таблиці немає транзитивних залежностей між не ключовими полями. Тобто значення будь-якого поля, що не входить у первинний ключ, не залежить від значення іншого поля, що також не входить у первинний ключ.

Згідно з цим, ми маємо базу даних, приведену до третьої нормальної форми.

## Даталогічна модель бази даних

На етапі даталогічного проектування здійснюється перехід від інфологічної моделі ПрО до логічної (даталогічної) моделі, яка підтримується засобами конкретної СУБД. Процес переходу від інфологічної до даталогічної моделі називається *відображенням*.

Даталогічна модель являє собою базу даних, структуровану на логічному рівні й орієнтовану на конкретну систему управління базами даних. Перш ніж виконати даталогічне проектування, необхідно вибрати СУБД.

Кожна конкретна система управління базами даних накладає ряд обмежень на побудову логічної моделі даних, тому насамперед необхідно вивчити специфіку і особливості СУБД, виявити всі фактори, які можуть вплинути на логічну модель БД.

Основними факторами, що впливаються на даталогічне проектування з боку СУБД, є такі:

* Тип логічної моделі, що його підтримує вибрана СУБД. Зараз найпоширенішими на ринку програмних засобів і в практиці автоматизації економічних розрахунків є реляційні СУБД. Крім реляційних моделей існують ієрархічні й сіткові моделі баз даних;
* Особливості фізичної організації даних у середовищі вибраної СУБД. Наприклад, у системі управління базами даних Paradox чи dBASE-системах база даних організована у вигляді набору взаємопов’язаних файлів форматів DT і DBF, усі інші об’єкти, такі як форми та звіти, також зберігаються в окремих файлах. У середовищі СУБД MySQL усі дані та інструментальні засоби роботи з ними зберігаються в IBD файлах, кожен з яких відповідає за окрему таблицю. Розширення файлу може бути іншим, якщо в якості драйверу для бази даних обрано іншу систему, яка відрізняється від стандартної, тобто InnoDB. Тому при проектуванні бази даних потрібно знати не лише правила побудови логічної, а й особливості фізичної організації БД;
* Кількісні обмеження, які накладає СУБД (наприклад, кількість рівнів ієрархії в ієрархічних моделях, можлива кількість полів, записів, файлів тощо).

Усе ще не знайдено формалізованих методів, які б давали змогу однозначно виконати даталогічне проектування. Тому його результат багато в чому залежить від уміння та рівня кваліфікації спеціалістів, які здійснюють проектні розробки.

В результаті даталогічного проектування можна отримати кілька варіантів побудови логічної моделі даних. Тому важливим моментом є оцінка отриманих моделей і вибір найбільш оптимального варіанта. Отриманий результат передусім потрібно оцінити з точки зору відповідності наявним машинним ресурсам. У разі невідповідності цим обмеженням потрібно здійснити перепроектування БД.

Крім того, на отриманій моделі необхідно перевірити умови виконання всіх запитів користувачів і вимог прикладних програм, тобто умову адекватності логічної моделі інформаційній моделі предметної області.

Для кожного поля таблиці реляційної бази даних “Музична пошукова система” (табл. 1.2 - 1.8) вказується розмір поля (кількість символів), тип даних та обмеження для кожного поля, якщо необхідно.

Для первинних ключів необхідно ввести заборону невизначених значень. Для інших полів - можливість заборони невизначених значень визначається семантикою предметної області [1].

Таблиця 1.2 - Таблиця Пісні (Songs)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Найменування атрибутів** | | **Тип поля** | **Розмір поля** | **Обмеження** |
| Унікальний код пісні | ID | int | 4 | Not null |
| Назва пісні | Name | varchar | 100 |  |
| Унікальний код автора | ID\_Author | int | 4 |  |
| Час пісні | Duration | varchar | 12 |  |
| Код тексту пісні | ID\_Lyrics | int | 4 |  |
| Унікальний код жанру | ID\_Genre | int | 4 |  |

Продовження таблиці 1.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік випуску пісні | Release\_year | Year |  |  |

Таблиця 1.3 - Таблиця Автори (Authors)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Найменування атрибутів** | | **Тип поля** | **Розмір поля** | **Обмеження** |
| Унікальний код автора | ID | int | 4 | Not null |
| Ім’я автора | Name | varchar | 100 |  |

Таблиця 1.4 - Таблиця Користувачі (Users)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Найменування атрибутів** | | **Тип поля** | **Розмір поля** | **Обмеження** |
| Унікальний код користувача | ID | int | 4 | Not null |
| Нікнейм користувача | Nickname | varchar | 100 |  |
| Електронна пошта користувача | Email | varchar | 30 |  |
| Пароль користувача | Password | varchar | 30 |  |
| Режим роботи користувача | User\_Mode | int | 4 |  |
| Номер телефону користувача | Phone | bigint |  |  |
| Час останнього входу користувача до системи | Last\_login | DateTime |  |  |

Таблиця 1.5 - Таблиця Жанри (Genres)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Найменування атрибутів** | | **Тип поля** | **Розмір поля** | **Обмеження** |
| Унікальний код жанру | ID | int | 4 | Not null |
| Назва жанру | Genre\_name | varchar | 15 |  |

Таблиця 1.6 - Таблиця Альбоми (Albums)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Найменування атрибутів** | | **Тип поля** | **Розмір поля** | **Обмеження** |
| Унікальний код альбому | ID | int | 4 | Not null |
| Код автора альбому | ID\_Author | int | 4 |  |
| Назва альбому | Name | varchar | 30 |  |
| Рік випуску альбому | Release\_year | Year |  |  |

Таблиця 1.7 - Таблиця Пісні в альбомах (Songs\_in\_albums)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Найменування атрибутів** | | **Тип поля** | **Розмір поля** | **Обмеження** |
| Унікальний код зв’язку | ID | int | 4 | Not null |
| Унікальний код пісні | ID\_Song | int | 4 |  |
| Унікальний код альбому | ID\_Author | int | 4 |  |

Таблиця 1.8 - Таблиця Права доступу (Access\_Rights)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Найменування атрибутів** | | **Тип поля** | **Розмір поля** | **Обмеження** |
| Унікальний код права | ID | int | 4 | Not null |
| Право доступу | Name | varchar | 20 |  |

Таблиця 1.9 - Таблиця Тексти (Lyrics)

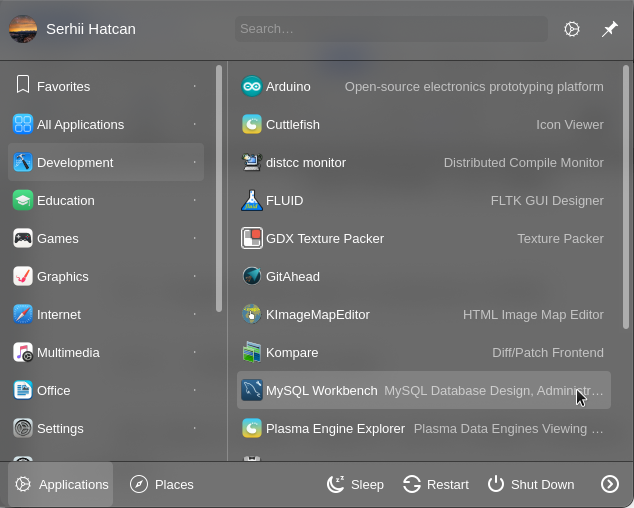
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Найменування атрибутів** | | **Тип поля** | **Розмір поля** | **Обмеження** |
| Унікальний код тексту | ID | int | 4 | Not null |
| Посилання на текст | URL | varchar | 200 |  |

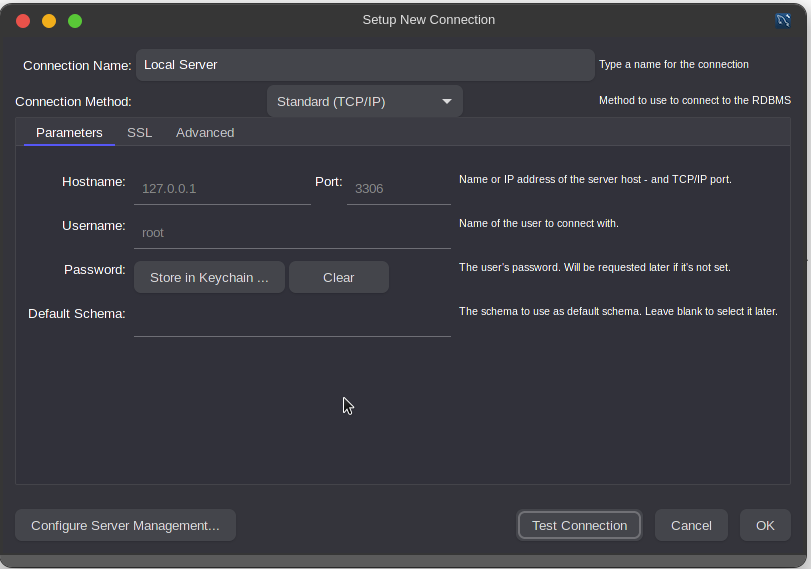
# 2 ПРАКТИЧНА ЧАСТИНА

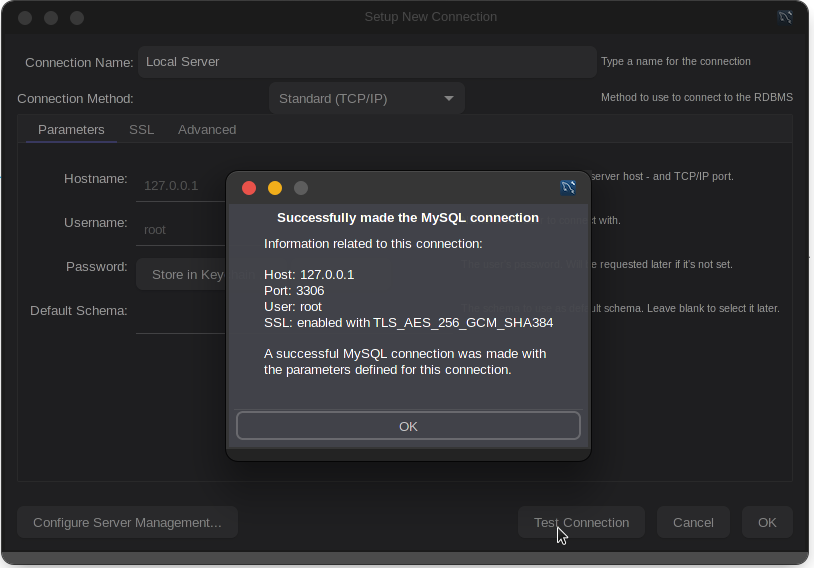
Створення бази даних за допомогою MySQL

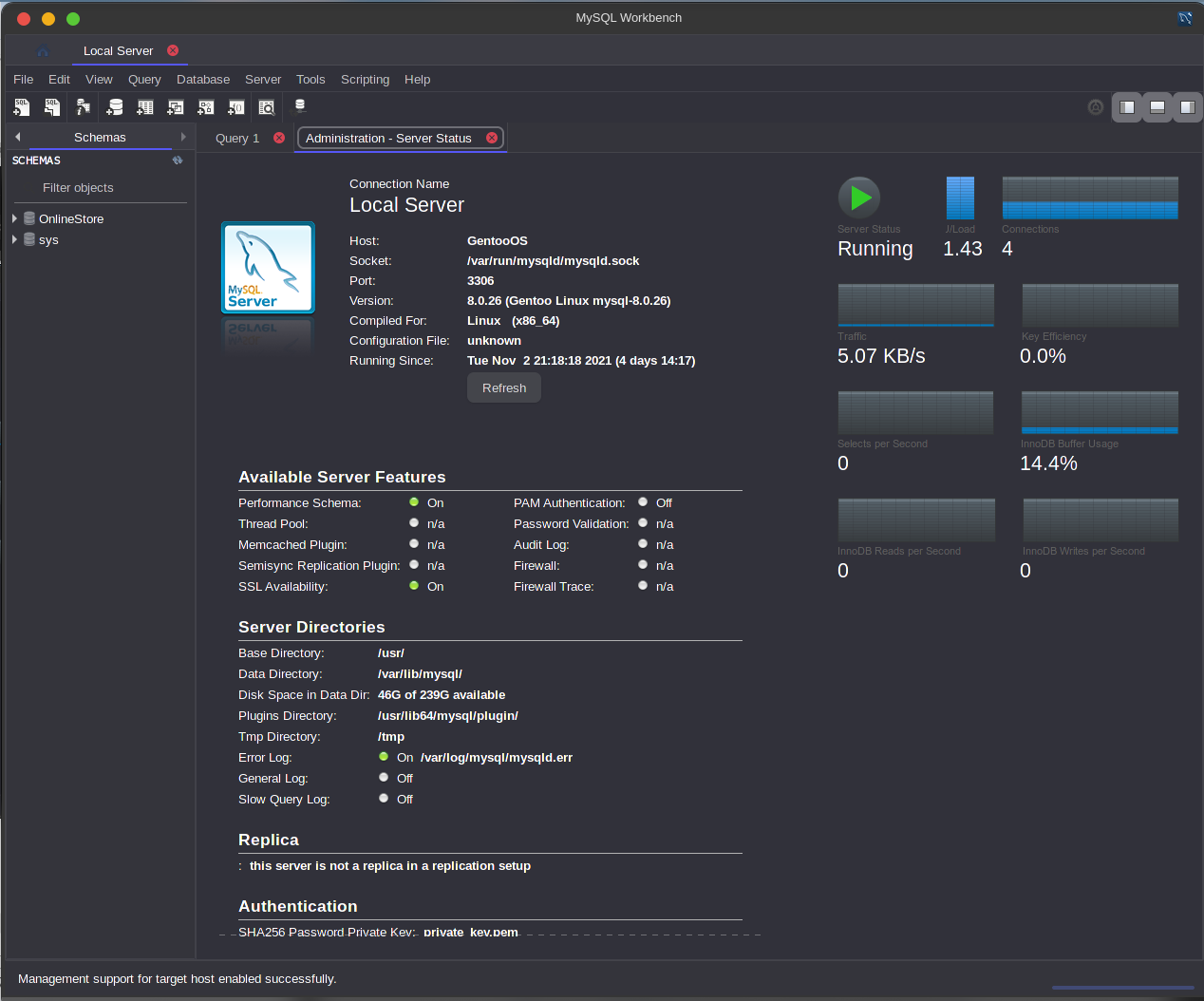
Створення бази даних

Для запуску MySQL обираємо утиліту MySQL WorkBech та підключаємося до серверу:









Далі за допомогою *sql*-команд створюємо реляційну базу даних із назвою Music\_Search\_System [6]:

create database Music\_Search\_System;

Результат зображено на рисунку 2.1.

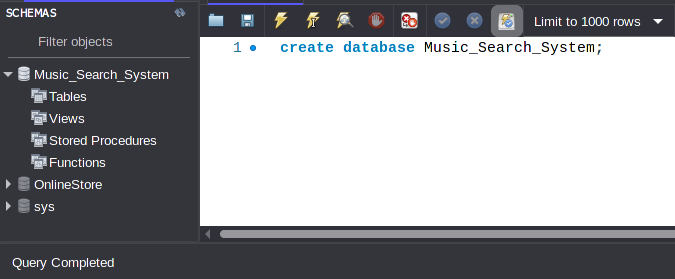


Рисунок 2.1 - Створення бази даних “Музична пошукова система”

Створення таблиць бази даних

В базі даних Music\_Search\_System створюємо наступні таблиці:

* пісні (Songs),
* автори (Authors),
* жанри (Genres),
* альбоми (Albums),
* тексти (Lyrics),
* пісні в альбомах (Songs\_in\_Albums),
* користувачі (Users),
* права доступу (Access\_Rights)

за допомогою наступних команд (рис. 2.2) [12,13]:

use Music\_Search\_System;   
  
-- Видалення всіх таблиць перед їх створенням, якщо вони були  
drop table if exists Access\_Rights;   
drop table if exists Songs\_in\_albums;   
drop table if exists Songs;   
drop table if exists Lyrics;   
drop table if exists Albums;   
drop table if exists Authors;   
drop table if exists Genres;   
drop table if exists Users;

-- Створення таблиці Автори  
create table Authors   
(   
       ID int not null auto\_increment,   
       Name varchar(100),   
       primary key(ID)   
);   
  
-- Створення таблиці Альбоми  
create table Albums   
(   
       ID int not null auto\_increment,   
       ID\_Author int,   
       Name varchar(100),   
       Release\_date Date,   
       primary key(ID),   
       foreign key(ID\_Author) references Authors(ID)   
);   
  
-- Створення таблиці Жанри  
create table Genres   
(   
       ID int not null auto\_increment,   
       Genre\_name varchar(15),   
       primary key(ID)   
);   
  
-- Створення таблиці Тексти  
create table Lyrics   
(   
       ID int not null auto\_increment,   
       URL varchar(200),   
       primary key(ID)   
);

-- Створення таблиці Пісні  
create table Songs   
(   
       ID int not null auto\_increment,   
       Name varchar(100),   
       ID\_Author int,   
       Duration varchar(12),   
       ID\_Lyrics int,   
       ID\_Genre int,   
       Release\_year year,   
       primary key(ID),   
       foreign key(ID\_Author) references Authors(ID) on delete cascade,   
       foreign key(ID\_Genre) references Genres(ID) on delete cascade,   
       foreign key(ID\_Lyrics) references Lyrics(ID) on delete cascade    
);   
  
-- Створення таблиці Права доступу  
create table Access\_Rights   
(   
       ID int not null auto\_increment,   
       Name varchar(20),   
       primary key(ID)   
);

-- Створення таблиці Користувачі  
create table Users   
(   
       ID int not null auto\_increment,   
       Nickname varchar(30),   
       Email varchar(30),   
       Password varchar(30),   
       User\_Mode int,   
       Phone BigInt,   
       Last\_login datetime,   
       primary key(ID)   
       foreign key (User\_Mode) references Access\_Rights(ID)   
);   
  
-- Створення таблиці пісні в альбомах  
create table Songs\_in\_albums   
(   
       ID int not null auto\_increment,   
       ID\_Song int unique,   
       ID\_Album int,   
       primary key(ID),   
       foreign key(ID\_Song) references Songs(ID) on delete cascade,   
       foreign key(ID\_Album) references Albums(ID) on delete cascade   
);

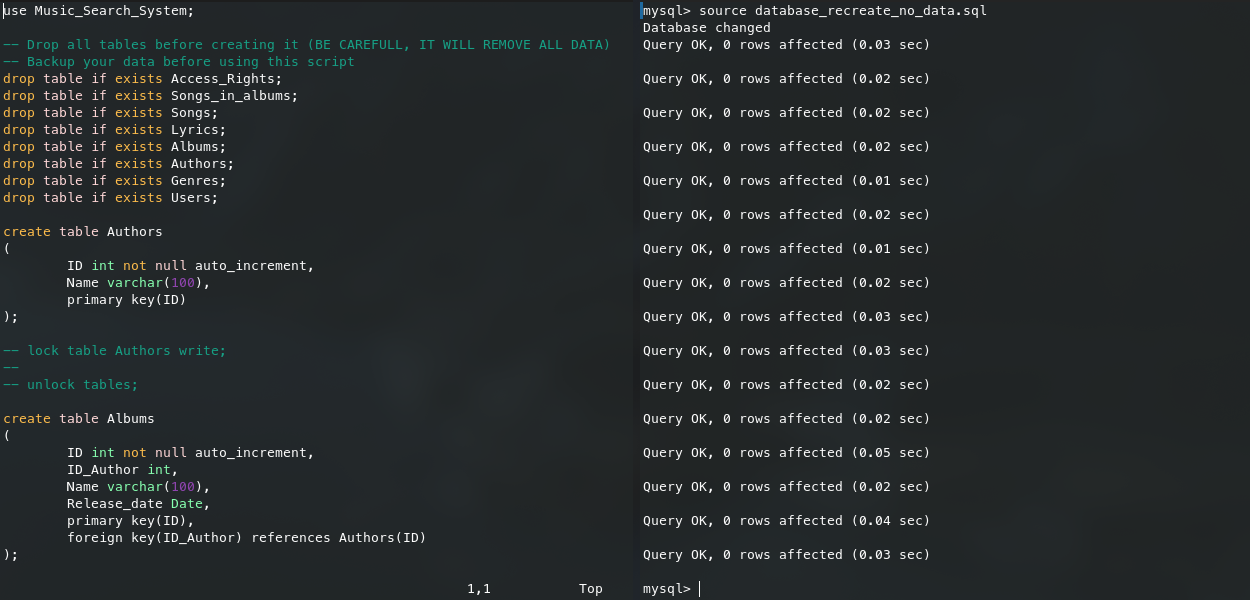
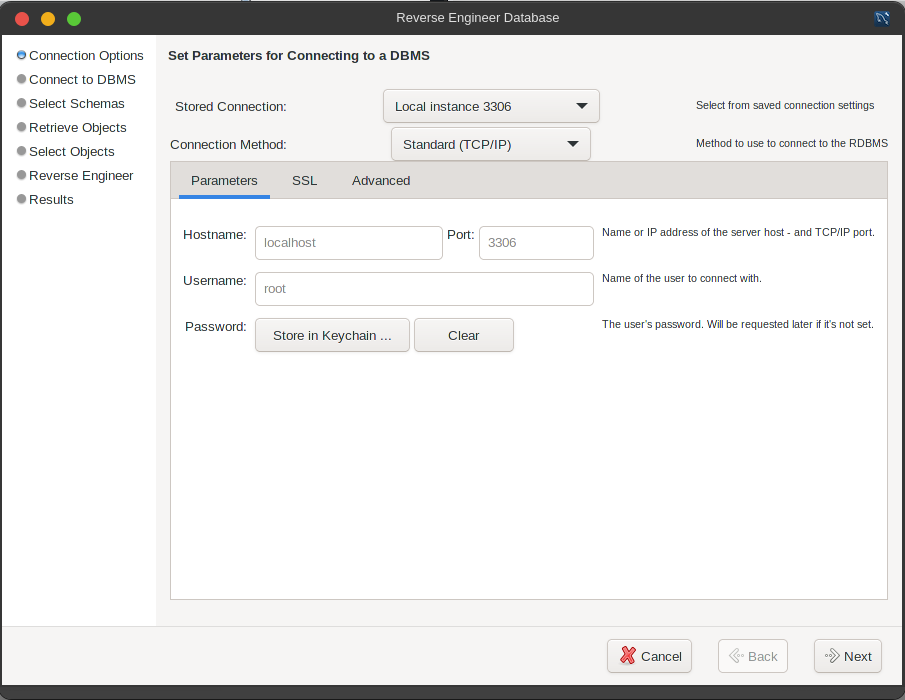


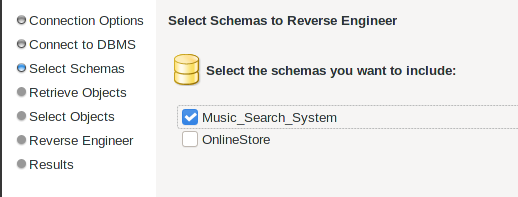
Рисунок 2.2 - Створення таблиць бази даних “Музична пошукова система”

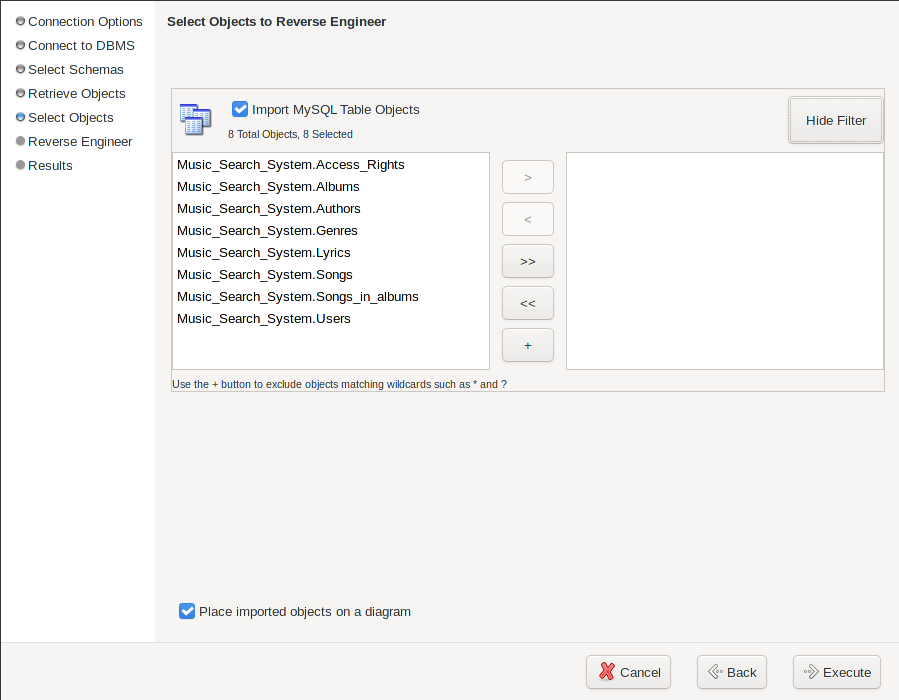
Створення діаграми бази даних

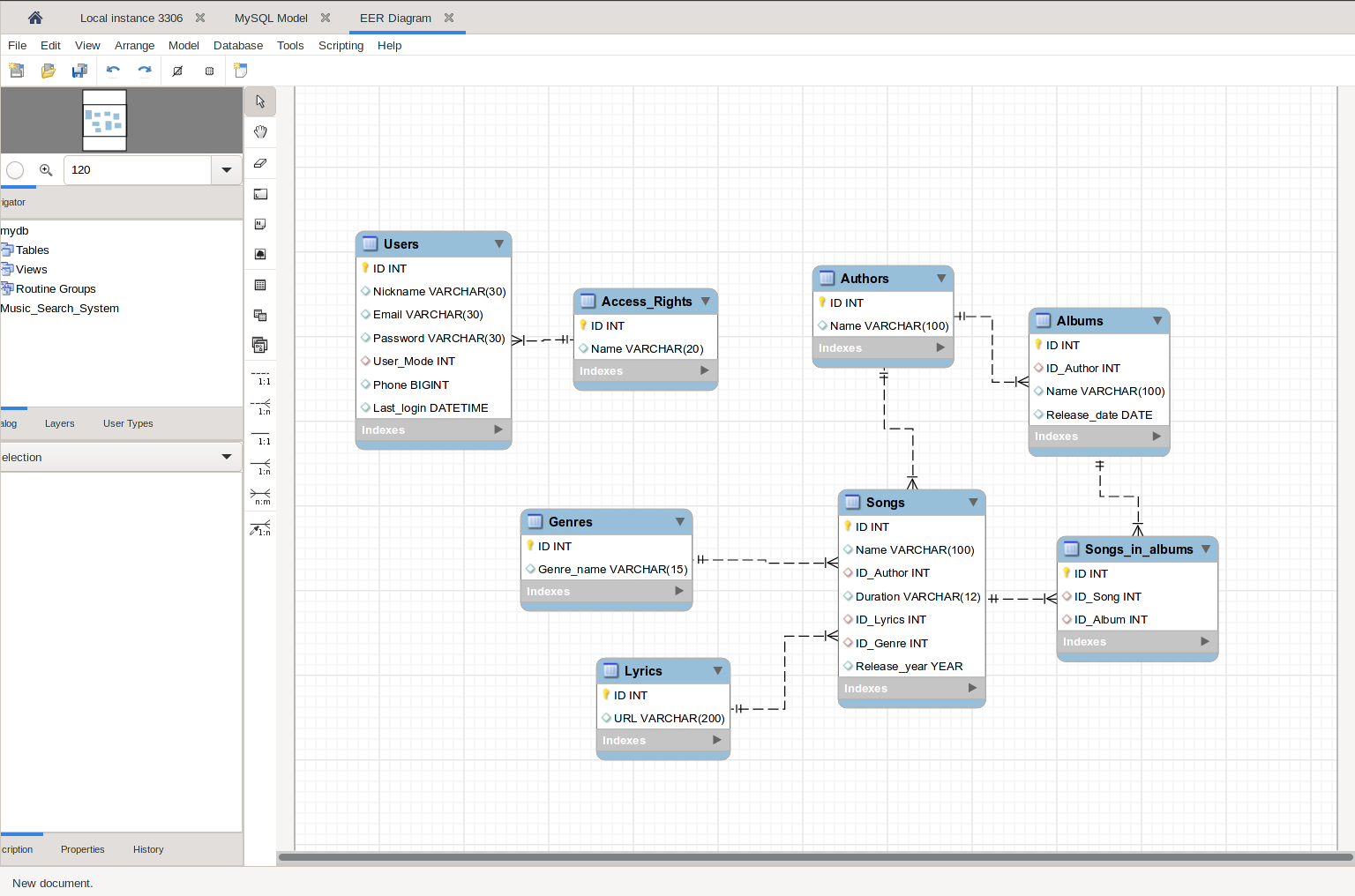
У розділі з маніпулювання базою данних в утиліті MySQL WorkBench методом зворотньої інженерії створюємо діаграму бази даних та перевіряємо зв’язки між таблицями.

Результат створення діаграми зображено на рисунку 2.3.









**Рисунок 2.3 - Діаграма реляційної бази даних “Музична пошукова система”**

Заповнення таблиць бази даних даними

Для зручнішого заповнення усіх таблиць бази даних використаємо мікропрограму search\_genius.py, код якої знаходиться в додатку А.



Рисунок 2.4 - Заповнення таблиць бази даних “Музична пошукова система”

Переглянемо результат в таблиці *Альбоми (Albums)*:

SELECT \* FROM `Albums`

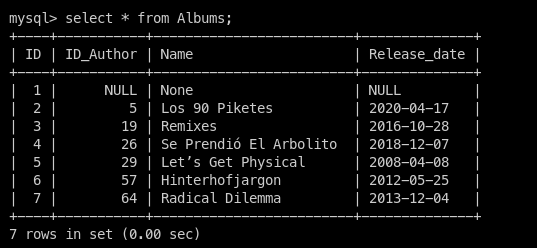


Рисунок 2.5 - Заповнена таблиця Альбоми (Albums)

Більше скриншотів з даними таблиці можна переглянути в додатку Б - Заповненні таблиці реляційної бази даних.

Створення збережених процедур

Збережена процедура *(stored procedure)* – це іменований набір команд мови SQL, що зберігається безпосередньо на сервері і представляє собою самостійний об'єкт – базу даних. Вона існує незалежно від таблиць або яких- небудь інших об'єктів баз даних. Збережена процедура може бути викликана клієнтською програмою, іншою збереженою процедурою або тригером [12,13]..

Розробимо процедуру для пошуку пісень в базі даних за заданими альбомом, назвою, роком випуску, автором або жанром (рис. 2.6):

drop procedure if exists search\_song;

delimiter //

create procedure search\_song(

in \_Author varchar(100),

in \_Album varchar(100),

in \_Song varchar(100),

in \_Release\_date date,

in \_Genre varchar(15)

)

begin

/\*Adding double backslash before brackets for regexp\*/

set \_Album = Replace(\_Album, '\(', '\\\\(');

set \_Album = Replace(\_Album, '\)', '\\\\)');

set \_Author = Replace(\_Author, '\(', '\\\\(');

set \_Author = Replace(\_Author, '\)', '\\\\)');

set \_Song = Replace(\_Song, '\(', '\\\\(');

set \_Song = Replace(\_Song, '\)', '\\\\)');

/\*Starter pack\*/

set @input = 'select s.\* from Songs s';

/\*Join requests\*/

set @a = " join Authors a on s.ID\_Author = a.ID";

set @b = " join Songs\_in\_albums sia on sia.ID\_Song = s.ID join Albums al on sia.ID\_Album = al.ID";

set @c = " join Genres g on g.ID = s.ID\_Genre";

/\*For detection, if I need to write `where` or `and`\*/

set @first\_param = 0;

/\*`Join` zone\*/

if \_Author is not NULL then

set @input = concat(@input, @a);

/\*select 'Author detected' as '';\*/

end if;

if \_Album is not NULL then

set @input = concat(@input, @b);

/\*select 'Album detected' as '';\*/

end if;

if \_Genre is not NULL then

set @input = concat(@input, @c);

/\*select 'Genre detected' as '';\*/

end if;

/\*`Where` zone\*/

if \_Author is not NULL then

if @first\_param = 0 then

set @input = concat(@input, " where regexp\_like(a.Name, '.\*(", \_Author, ").\*')");

set @first\_param = 1;

else

set @input = concat(@input, " and regexp\_like(a.Name, '.\*(", \_Author, ").\*')");

end if;

end if;

if \_Album is not NULL then

if @first\_param = 0 then

set @input = concat(@input, " where regexp\_like(al.Name, '.\*(", \_Album, ").\*')");

set @first\_param = 1;

else

set @input = concat(@input, " and regexp\_like(al.Name, '.\*(", \_Album, ").\*')");

end if;

end if;

if \_Genre is not NULL then

if @first\_param = 0 then

set @input = concat(@input, " where regexp\_like(g.Name, '.\*(", \_Genre, ").\*')");

set @first\_param = 1;

else

set @input = concat(@input, " and regexp\_like(g.Name, '.\*(", \_Genre, ").\*')");

end if;

end if;

if \_Song is not NULL then

if @first\_param = 0 then

set @input = concat(@input, " where regexp\_like(s.Name, '.\*(", \_Song, ").\*')");

set @first\_param = 1;

else

set @input = concat(@input, " and regexp\_like(s.Name, '.\*(", \_Song, ").\*')");

end if;

/\*select 'Song name detected' as '';\*/

end if;

if \_Release\_date is not NULL then

if @first\_param = 0 then

set @input = concat(@input, " where regexp\_like(s.Release\_date, '.\*(", \_Release\_date, ").\*')");

set @first\_param = 1;

else

set @input = concat(@input, " and regexp\_like(s.Release\_date, '.\*(", \_Release\_date, ").\*')");

end if;

/\*select 'Release date detected' as '';\*/

end if;

/\*select @input as '';\*/

prepare stmt from @input;

execute stmt;

deallocate prepare stmt;

end//

delimiter ;

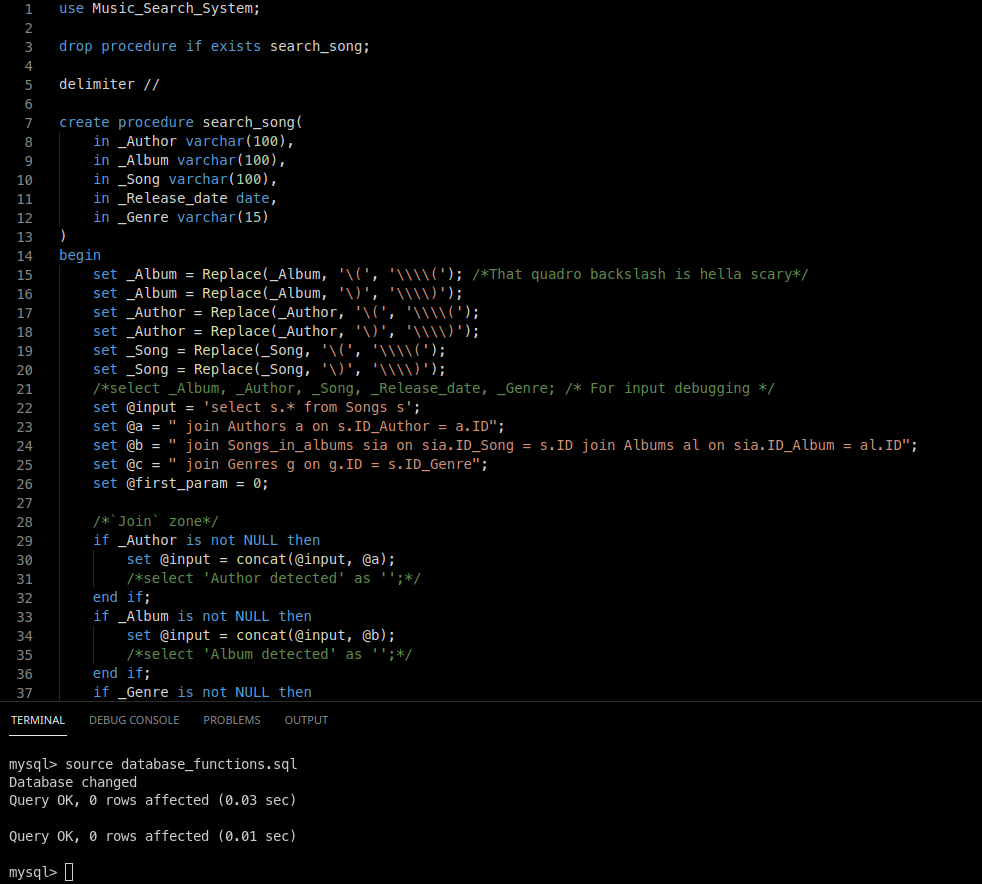
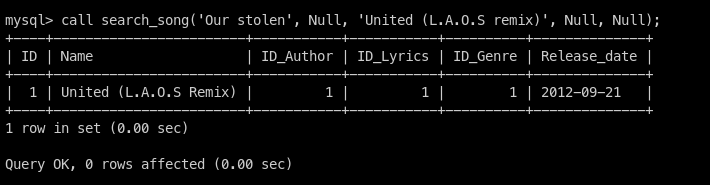


Рисунок 2.6 - Збережена процедура для пошуку пісні за одним із параметрів, заданих користувачем

Для звернення до процедури можна використовувати наступну команду:

call search\_song(\*parameters\*)



Далі створюємо збережені процедури для інших таблиць реляційної бази даних “Музична пошукова система” (Додаток В. Створення збережених процедур).

Створення тригерів

Тригер *(trigger)* – це спеціальний тип збережених процедур, що запускаються сервером автоматично при виконанні тих чи інших дій з даними таблиці. Кожен тригер прив'язується до конкретної таблиці. Коли користувач намагається, наприклад, змінити дані в таблиці, сервер автоматично запускає тригер і, тільки якщо він завершується успішно, дозволяється виконання змін. Всі здійснені тригером модифікації даних розглядаються як одна транзакція. У разі виявлення помилки або порушення цілісності даних відбувається відкат цієї транзакції [6,13].

Тим самим внесення змін буде заборонено. Будуть також відмінені всі зміни, вже зроблені тригером.

Створимо для таблиці “Права користувачів” (Access\_rights) тригер, який заборонятиме вставку нових рядків у таблицю, видаючи при цьому повідомлення “Changes inside this table are disabled.” (рис. 2.7):

use Music\_Search\_System;

drop trigger if exists block\_access\_rights;

delimiter //

create trigger block\_access\_rights before insert on Access\_Rights

for each row

begin

SIGNAL SQLSTATE '45000'

set MESSAGE\_TEXT = 'Changes inside this table are disabled.';

end//

delimiter ;

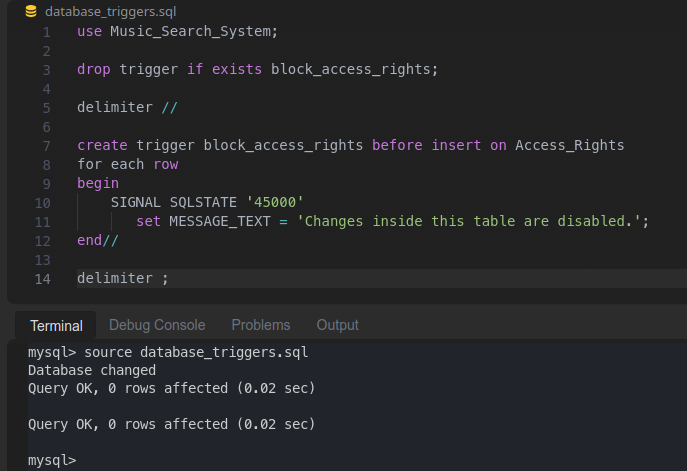


Рисунок 2.7 - Тригер, який заборонятиме вставку нових рядків у таблицю “Права користувачів” (Access\_rights)

Перевіримо працездатність тригера:

Insert into Access\_rights value (3, ‘test’);



Далі створюємо тригери для інших таблиць бази даних “Музична пошукова система” (Додаток Г - Створення тригерів).

**ДЖЕРЕЛА**

1. Атре, Ш. Структурный подход к организации баз данных / Ш. Атре. - М.: Финансы и статистика, 2010. - 317 с.
2. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Третье издание. Москва-Сакт-Петербург-Киев: Вильямс, 2003 - 1440 с.
3. Марков А.М., Лисовский К.Ю. Базы данных. Введение в теорию и методологию: Учебник. - М.: Финансы и статистика, 2006. - 512 с.
4. Хомоненко А.Д., Цыганков В.М., Мальцкв М.Г. Базы данных: Учебник для высших учебных заведений. - 4-е изд., доп. и перераб. - СПб.: КОРОНА принт, 2004. - 736 с.
5. Роб П., Коронел К. Системы баз данных: проектирование, реализация и управление. - 5-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2004. - 1040 с.
6. <https://uk.wikipedia.org/wiki/База_даних>
7. <https://uk.wikipedia.org/wiki/MySQL>
8. <https://uk.wikipedia.org/wiki/Реляційна_база_даних>
9. <https://uk.wikipedia.org/wiki/MySQL_Workbench>