**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

**ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ**

**Кафедра цифрової економіки та бізнес-аналітики**

**КУРСОВА РОБОТА**

з прогнозування та адміністрування СД та БД

на тему:

**Інформаційна система книжкового магазину**

**спеціальність:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**051 «Економіка»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                              (код та найменування спеціальності)

**спеціалізація:** \_\_\_\_\_\_\_\_\_«Інформаційні технології в бізнесі» \_\_\_\_\_\_\_\_\_\_\_

                             (найменування спеціалізації)

**освітній ступінь: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**бакалавр**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

              (бакалавр/магістр)

|  |  |  |
| --- | --- | --- |
| **Науковий керівник:**  к.е..доцент кафедри **Депутат Б.Я**.    (науковий ступінь, посада, прізвище, ініціали)  \_\_\_\_\_\_\_\_\_\_  **“\_\_\_\_”** \_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.       (підпис) |  | **Виконавець:**  студент(ка) групи \_УФЕ-21с\_Гринда Т.М.\_                         (прізвище, ініціали)  \_\_\_\_\_\_\_\_\_\_\_\_  **“\_\_\_\_”** \_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.      (підпис) |

**Загальна**

**кількість балів**  \_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                                            (підпис, ПІП членів комісії)

**ЛЬВІВ 2023**

**ВСТУП**

**РОЗДІЛ 1. АНАЛІЗ ВИМОГ**

* 1. **Постановка завдання**
  2. **Розробка моделі варіантів використання веб-сайту**

Інформаційна система інтернет-магазину книг має три рівні доступу:

1. Будь-який користувач мережі
2. Зареєстрований користувач
3. Адміністратор.

На кожному з тих рівнів надається певний набір можливостей, які можуть зробити користувачі сайту. Відвідувачі інтернет-магазину з першим рівнем доступу мають найбільше обмежень. Вони можуть переглядати головну сторінку сайту, переглядати інформацію про робітників магазину, ознайомитись з асортиментом книг в магазині, а також можуть зареєструватись в магазині. Проте їм заборонено додавати товари в корзину та оформляти замовлення. Після того, коли користувач зареєструється або авторизується на сайті він матиме третій рівень доступу на сайті. В нього з’являться дозволи, які були відсутні на попередньому рівні. Зареєстрований відвідувач сайту може додавати товари в корзину і оформляти замовлення. Третій рівень доступу має лише адміністратор. Він можить перегляди усі замовлення, які були здійсненні в магазині, переглядати інформацію про, книги, клієнтів та працівників (рис.1.1.)

Використання такої системи надасть користувачам такі переваги:

1. Користувачі можуть купувати книги з будь-якого місця де є доступ до інтернету.
2. Також в користувачів не має географічних обмежень і вони можуть купляти книги не зважаючи на робочі години магазину.
3. Це дозволяє зекономити час та зусилля, які були б витрачені на пошук книжкових магазинів і фізичне відвідування.
4. Вони можуть швидко та легко за допомогою пошуки перевірити чи певна книга в асортименті.
5. Також користувачі мають можливість користуватись корзиною. Це дозволить клієнту зберегти книгу, яка їм сподобалась і подумати чи справді вони хочуть її купити.

Для адміністраторів інтернет магазину така інформаційна система також має свої переваги:

1. Вони можуть переглядати статистику інтернет-магазину.
2. Також адміністратори можуть переглядати усі замовлення, які були здійсненні в інтернет-магазині.
   1. **Аналіз засобів реалізації (техніко-економічне обґрунтування**

**вибору)**

Для створення інформаційної системи інтернет-магазину книг було використано наступні технології:

1. HTML, CSS, JS. Дані інструменти були використанні для створення front-end.
2. PHP, SQL. Ці компоненти використовувались для створення beck-end.

HTML - стандартизована мова розмітки документів. За допомогою HTML було розміщено елементи на веб-сторінках.

Використання CSS дозволило застосовувати стилі для розміщених елементів на сайті. CSS – це мова, яка використовується для опису зовнішнього виду сторінки, написаною мовою розмітки HTML.

JavaScript – вважається однією з найпопулярніших мов програмування в світі. JavaScript- це динамічна, об’єктно-орієнтована мова програмування. За допомогою цього інструменту додається інтерактивність та функціональність на сайт, що дозволяє йому працювати коректно. За допомогою цією мови можна здійснювати перехід між сторінками або додавати обробник подій для кнопок. JS підтримується всіма браузерами і її легко можна інтегрувати у верстку HTML. Використання JavaScript при створенні веб-сайту надає ряд переваг:

1. Розширення можливостей HTML та CSS: JavaScript дозволяє додавати динамічність та взаємодію до HTML-сторінок. Він може маніпулювати елементами сторінки, змінювати їх стилі, додавати або видаляти елементи, змінювати контент і багато іншого. Це дозволяє створювати більш складні та інтерактивні веб-інтерфейси.
2. Розширення функціональності веб-додатків: JavaScript має велику кількість сторонніх бібліотек і фреймворків, які дозволяють легко розширювати функціональність веб-додатків.
3. JavaScript є широко підтримуваною мовою програмування, що працює в різних браузерах. Більшість сучасних браузерів підтримують JavaScript, що дозволяє розробникам створювати веб-додатки, які працюють у багатьох середовищах без необхідності писати окремий код для кожного браузера.

Для проектування Beck-end було використано PHP. Це скриптова мова програмування, яка було створена для генерації HTML-сторінок на стороні вебсервера. Особливістю цієї мови є те що в неї безпосередньо можна вбудувати код HTML-сторінки.

Використання PHP надає наступні перев аги:

* Можна підключатись до всіх баз даних в яких є драйвер.
* Дозволяє обробляти сценарії з високою швидкістю.
* За рахунок того що PHP виконується на стороні сервере, зникає потреба у встановленні додаткового програмного забезпечення на пристрої користувачів.
* Ця мова легка у вивченні.
* Багато хостингів підтримують PHP.
* Велика кількість функцій та бібліотек: PHP має велику кількість вбудованих функцій та бібліотек, які значно спрощують розробку веб-додатків.

Ще в процесі створення інформаційної системи було використано SQL. Це стандартна мова для зберігання, обробки та отримання даних з бази даних. Ця мова використовується в майже всіх додатках та сайтах для взаємодії з базами даних. Її функцією є створення запитів до БД. Дана мова виступає посередником між сайтом і базою даних. За допомогою SQL можна обробляти великі масиви даних і в результаті отримувати структуровану інформацію.

Використання SQL надає такі переваги:

* SQL має простий та логічний синтаксис, що дозволяє легко вивчати та використовувати цю мову. Запити SQL виражаються у формі зрозумілих команд, таких як SELECT, INSERT, UPDATE та DELETE, що робить їх легко зрозумілими та зручними для роботи з базами даних.
* Це є стандартна мова для керування базами даних і підтримується більшістю реляційних баз даних. Це означає, що код SQL можна використовувати на різних платформах та з різними базами даних без необхідності значних модифікацій.
* SQL надає потужні засоби для аналізу даних. За допомогою складних запитів можна витягувати, фільтрувати, сортувати та групувати дані відповідно до потреб користувача. SQL також надає можливість виконувати операції об'єднання, перетину та різниці між таблицями, що дозволяє здійснювати складні операції з об'єднаними даними.
* SQL бази даних використовують оптимізатор запитів, який автоматично вибирає найбільш ефективний спосіб виконання запиту. Це дозволяє досягти швидкості та оптимальної продуктивності при обробці великого обсягу даних.
* Ця мова дозволяє виконувати складні операції з базами даних, такі як об'єднання, підзапити, агрегатні функції, створення тригерів та збережених процедур. Це дає розробникам гнучкість та можливість реалізувати складні бізнес-логіки в базі даних.

**РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ**

**2.1. Опис моделі даних**

Модель даних – це абстрактна структура, яка визначає спосіб організації та представлення даних в БД.

В процесі виконання цієї роботи було створено модель даних, яка зображена на рис.2.1.

Зображення, що містить текст, знімок екрана, схема, ряд

Автоматично згенерований опис

Рис.2.1

З моделі даних можна побачити, що база даних складається з восьми таблиць: CATEGORY, PUBLICATION, AUTHOR, BOOK, OFFICE EMPLOYEE, CUSTOMER, ORDERS. Таблиця CATEGORY складається з наступних полів: ID, TYPE\_OF\_BOOK, LANGUAGE\_BOOK, FORMAT\_BOOK. Таблиця PUBLICATION має такі поля: ID, NAME, LOCATION, EMAIL, PHONE\_NUMBERS. У таблиці AUTHOR присутні такі поля: ID, FIRSTNAME, LASTNAME, BATE\_BIRTH. Таблиця BOOK складається з таких полів: ID, NAME, CATEGORY, YEAR\_PUBLICATION, DESCRIPTION\_BOOK, PRICE, AUTHOR\_ID, PUBLICATION\_ID. Таблиця OFFICE включає поля: ID, NAME, CITY, STREET, PHONE\_NUMBERS, EMAIL. У таблиці EMPLOYEE містяться такі поля: ID, FIRSTNAME, LASTNAME, POSITION, DATE\_EMPLOYMENT, RATE, OFFICE\_ID. Таблиця CUSTOMER складається з полів: ID, FIRSTNAME, LASTNAME,EMAIL, PASSWORD, GENDER, DATE\_BIRTH, CITY, PHONE\_NUMBER. Остатня таблиця, яка зображена в моделі даних це ORDERS. Вона має сім полів: ID, BOOK\_ID, EMPLOYEE\_ID, CUSTOMER\_ID, NUMBERS, DATE\_ORDER, AMOUNT.

Усі таблиці, окрім ORDERS мають первині ключі ID. Це ми можемо побачити за допомогою символу, який зображає ключ і знаходиться біля поля ID.

З моделі даних яка зображена на рис.2.1. можна побачити наступне:

* В таблиці "EMPLOYEE" є зовнішній ключ "OFFICE\_ID", який посилається на первинний ключ "ID" в таблиці "OFFICE". Це дозволяє зв'язати працівників з конкретним офісом, в якому вони працюють.
* В таблиці "BOOK" є зовнішній ключ "CATEGORY\_ID", який посилається на первинний ключ "ID" в таблиці "CATEGORY". Це дозволяє прив'язати книги до конкретної категорії.
* В таблиці "BOOK" також є зовнішній ключ "AUTHOR\_ID", який посилається на первинний ключ "ID" в таблиці "AUTHOR". Це дозволяє прив'язати книги до конкретного автора.
* В таблиці "BOOK" також присутній зовнішній ключ "PUBLICATION\_ID", який посилається на первинний ключ "ID" в таблиці "PUBLICATION". Це дозволяє прив'язати книги до конкретного видавництва.
* В таблиці "ORDERS" є зовнішній ключ "CUSTOMER\_ID", який посилається на первинний ключ "ID" в таблиці "CUSTOMER". Це дозволяє зв'язати замовлення з конкретним клієнтом.
* В таблиці "ORDERS" є зовнішній ключ "EMPLOYEE\_ID", який посилається на первинний ключ "ID" в таблиці "EMPLOYEE". Це дозволяє зв'язати замовлення з конкретним працівником.
* В таблиці "ORDERS" є зовнішній ключ "BOOK\_ID", який посилається на первинний ключ "ID" в таблиці "BOOK". Це дозволяє зв'язати замовлення з конкретною книгою.

Зовнішні ключі допомагають управляти залежностями між таблицями і забезпечують цілісність даних в базі даних. Вони дозволяють встановлювати зв'язки між записами у різних таблицях.

Існує декілька видів моделей даних:

* Ієрархічна модель
* Мережева модель
* Реляційна модель

Ієрархічна модель даних – це модель, у якій база даних представляється у вигляді деревоподібної (ієрархічної) структури, яка складається з об’єктів різних рівнів. Кожен об’єкт такої моделі може містити декілька об’єктів нижчого рівня. В такій моделі реалізується відношення предка до нащадку. При цьому може бути ситуація що предок моє декілька нащадків або не має жодного. Кожен нащадок може мати лише одного предка.

Ієрархічне дерево може мати лише одну вершину. Ця вершина не підпорядковується ніякій іншій і має найвищий рівень. До кожного записи в базі даних існує лише один можливий шлях від кореневого запису.

Структура даних в ієрархічній моделі вимагає необхідного зв'язку між батьківськими і дочірніми об'єктами даних. Ці об'єкти даних включають такі елементи структури:

* Атрибут – найменша одиниця елемента даних. Він визначає функціональну характеристику об’єкта
* Запис – це група атрибутів, яка представляє конкретний екземпляр описуваного об'єкта.
* Групове відношення – це зв’язок між записами, які відображають зв’язок між батьківськими і дочірніми записами
* Ключовий елемент – це атрибут, який містить лише унікальні значення для кожного запису.

Використання ієрархічної моделі має наступні переваги:

* Легко можна отримати дані, за рахунок зв’язку між таблицями
* Дані внесені в батьківській таблиці автоматично оновлюються в дочірній, за рахунок цього забезпечується цілісність даних.
* Ієрархічна модель є простою для розуміння і реалізації, за рахунок батьківського-дочірнього відношення.
* Через те, що дані в цій моделі організовуються у вигляді дерева забезпечується швидкий доступ до записів. Запити у такій структурі ефективно виконуються.
* Забезпечується безпека даних.
* Така модель даних ефективна у відношеннях 1 до ∞.

Графічно ця модель зображена на рис.2.2.

Зображення, що містить ескіз, схема, білий, ряд

Автоматично згенерований опис

Рис.2.2.

Мережева модель даних – це модель, яка складається з записів, елементів даних (поля з назвами) та зв'язків типу "один до багатьох" або "багато до багатьох", які з'єднують записи між собою. Зв'язок "один-до-багатьох" означає, що значенню елемента Х відповідають багато значень пов'язаних з ним елементів У. Зв’язок багато до багатьох означає, що багато записів зв’язані з багатьма записами.

Головна відмінність між мережевою моделлю даних і ієрархічною моделлю даних полягає в обмеженнях, які стосуються зв'язків між записами. В ієрархічній моделі нащадок мав лише одного предка, а в мережевій предків може бути декілька. Така модель складається з набору екземплярів певного типу запису і набору екземплярів певного типу зв’язків між записами. Тип зв’язку визначається для предка і нащадка.

Перевагою використання такої моделі даних є можливість ефективної реалізації за показниками витрат пам’яті та оперативності.

Проте така модель має свої недоліки. Дана модель є складна у використані. Така модель не є повністю незалежна від програми, тому що процедура вибірки даних залежить від фізичної організації цих даних.

Графічне зображення мережевої моделі даних представлено на рис.2.3.

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

Рис.2.3.

Реляційна модель даних – це модель, в якій текстова або числова інформація подається у вигляді таблиць. Таблиці складаються з рядків та стовпців, які називаються атрибутами.

Назва реляційна походить від слові відносний. Така модель базується на відношеннях між записами в таблиці.

Кожна таблиця в такій моделі даних має задовільняти наступні вимоги:

* Значення в таблицях мають бути одиничними
* Таблиця представляється, як один об’єкт, який складається з рядків та стовпців.
* Стовпчики в базі даних мають мати унікальні значення
* Таблиця має мати первинний ключ, або поле яке ідентифікує поле таблиці.
* Таблиця не може містити однакові рядки.

Така модель має наступні переваги:

* Простота і зрозумілість: Дані організовані у вигляді таблиць, що легко сприймається та аналізується.
* Гнучкість і розширюваність: Реляційна модель дозволяє легко додавати нові таблиці, змінювати структуру бази даних та виконувати складні операції зв'язків між таблицями.
* Ефективність: Реляційні бази даних можуть бути оптимізовані для швидкого доступу до даних.
* 4. Цілісність даних: Реляційна модель включає правила цілісності даних, які дозволяють контролювати додавання, оновлення та видалення даних у базі даних.
* 5. Широке використання: Реляційна модель є стандартом для багатьох систем управління базами даних (СУБД) і має велику підтримку та інструментарій.
* Реляційна модель включає правила цілісності даних, які дозволяють контролювати додавання, оновлення та видалення даних у базі даних. Ці правила забезпечують коректність даних.

Графічне зображення реляційної моделі даних зображено на рис.2.4.

![Зображення, що містить ряд, схема, годинник, дизайн

Автоматично згенерований опис]()

Рис.2.5.

В процесі реалізацію даного завдання було використано реляційну базу даних.

**2.2. Нормалізація відношень**

Нормалізація відношень – це процес декомпозиції складних відношення, які є БД на простіші. Цей процес забезпечує перехід схеми відношень бази даних в нормальні форми. Кожна з таких форм має свої обмеження. У структурі, яка має форму вищого порядку, видаляється з таблиць надмірна описова інформація.

Нормалізація баз даних дозволяє згрупувати пов’язані дані в одну таблицю. Атрибутивні бо пов’язані дані розміщуються в різних таблицях. Пізніше ці таблиці зв’язуються за допомогою взаємозв’язків між дочірньою і батьківською таблицями.

Існує така послідовність нормальних форм:

* Перша звичайна форма (1NF)
* Друга нормальна форма (2NF)
* Третя нормальна форма (3NF)
* Звичайна форма Бойса-Кодда (3.5NF)
* Четверта нормальна форма (4NF)
* П'ята нормальна форма (5NF)

Перша звичайна форма (1NF) дозволяє ефективну організацію та обробку даних у таблицях, забезпечуючи однозначність, структурованість та безпеку даних. Ця форма вимагає, щоб дані були представлені у вигляді таблиць, де кожне поле містить лише атомарне значення, а кожен рядок таблиці має унікальний ідентифікатор.

Друга нормальна форма (2NF): У другій нормальній формі вимагається, щоб кожний атрибут в таблиці пов'язаний із всіма атрибутами, які є частиною їх первинного ключа. Значення, що пов'язані із тим же первинним ключем, витягуються в окрему таблицю.

Третя нормальна форма (3NF): У третій нормальній формі вимагається, щоб кожен атрибут у таблиці був залежний від первинного ключа, а не від інших атрибутів. Це допомагає уникнути транзитивних залежностей між атрибутами.

Звичайна форма Бойса-Кодда (3.5NF): Ця форма вводиться для усунення аномалій, що виникають при наявності залежностей між неключовими атрибутами. Вона розбиває такі атрибути на окремі таблиці, залежні від їх первинних ключів.

Четверта нормальна форма (4NF): Четверта нормальна форма вимагає усунення мультимодальних залежностей, коли один атрибут залежить від групи атрибутів, які не є первинним ключем. Це досягається шляхом розбиття таких груп атрибутів на окремі таблиці.

П'ята нормальна форма (5NF): П'ята нормальна форма, також відома як проектна нормальна форма (PNF), спрямована на усунення залежностей природного зв'язку між атрибутами, які не можна усунути за допомогою попередніх нормальних форм. Вона дозволяє представити такі залежності як окремі таблиці з посиланнями на них.

На основі моделі даних, яка зображена на рис.2.1. можна побачити, що на таблиці знаходяться у 1, 2 та 3 нормальних формах. Кожна таблиця має первиний ключ, а всі неключові атрибути функціонально залежать від первиного ключа, що дозволяє досягти нормалізації даних та зменшити ризик втрати інформації та аномалій при роботі з даними.

Таблиці бази даних книжкового магазину в таких нормальних формах:

1NF: Таблиці знаходяться в 1 нормальній формі, оскільки атрибути містять атомарні значення.

2NF: Зважаючи на структуру таблиць, можна сказати, що вони знаходиться в 2 нормальній формі, оскільки всі атрибути залежать від первинного ключа ID.

3NF: Таблиці також знаходиться в 3 нормальній формі, оскільки всі неключові атрибути функціонально залежать від первинного ключа ID.

**2.3. Визначення типів даних**

Типи даних – це характеристика об’єкту. Тип даних визначає множину припустимих значень, формат у якому зберігаються дані, розмір виділеної пам’яті.

У таблиці 2.1. представлено назву текстових типів даних та їх опис.

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| CHAR() | Містить рядок фіксованої довжини. Довжина задається в дужках |
| VARCHAR | Містить рядок змінної довжини. В дужках задається максимальна довжина |
| TINYTEXT | Рядок з найбільшою довжиною 255 символів |
| TEXT | Рядок з найбільшою довжиною 65,535 символів |
| BLOB | Великий двійковий об’єкт |
| MEDIUMTEXT | Рядок з максимальною довжиною в 16,777,215 символів. |
| MEDIUMBLOB | Рядок, який зберігає 16 Мегабайтів даних |
| LONGTEXT | Рядок, який має максимальну довжину 4,294,967,295 символів. |
| LONGBLOB | Великий двійковий об’єкт, який зберігає 4 Гігабайти даних |
| ENUM | Дозволяє вести список значень |
| SET | Містить 64 значень списку, і не зберігає більше одного вибору |

Текстові типи даних в SQL використовуються для зберігання різноманітної текстової інформації. Вони дозволяють зберігати рядки символів різної довжини і надають гнучкість для роботи з текстовими даними. Використання конкретного типу залежить від потреб бази даних і вимог до обробки текстової інформації. Наприклад, CHAR може бути корисним для полів з фіксованою довжиною, тоді як VARCHAR і TEXT надають гнучкість для зберігання різних розмірів текстових даних.

Окрім текстових типів даних існують ще числові типи. Вони представлені в таблиці 2.2.

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| INT | Цілі числа без десяткової частини. |
| BIGINT | Великі цілі числа без десяткової частини. |
| FLOAT | Числа з плаваючою комою (одинарна точність). |
| DOUBLE | Числа з плаваючою комою подвійної точності. |
| DECIMAL | Числа з фіксованою точністю та масштабом. |
| TINYINT | Малий цілий тип з діапазоном від -128 до 127. |
| SMALLINT | Малий цілий тип з діапазоном від -32768 до 32767. |
| REAL | Числа з плаваючою комою (одинарна точність). |
| BOOLEAN | Логічний тип, який може приймати значення TRUE або FALSE. |

Числові типи даних в SQL використовуються для зберігання та обробки числової інформації. Вони надають можливість зберігати цілі числа, числа з плаваючою комою, числа з фіксованою точністю та інші числові значення.

Окрім текстових та числових даних ще існують тип даних дати та часу. Ці типи даних представлено в таблиці 2.3.

|  |  |
| --- | --- |
| **Назва** | **Опис** |
| DATE | Зберігає дату в форматі 'РРРР-ММ-ДД'. |
| TIME | Зберігає час в форматі 'ГГ:ХХ:СС'. |
| DATETIME | Зберігає дату та час в форматі 'РРРР-ММ-ДД ГГ:ХХ:СС'. |
| TIMESTAMP | Зберігає дату та час у форматі TIMESTAMP (часто використовується для відстеження змін у записах). |
| YEAR | Зберігає рік у форматі 'РРРР'. |

Ці типи даних дозволяють зберігати і обробляти інформацію про дату та час у базі даних. Кожен тип має свої особливості та формати зберігання.

У базі даних книжкового магазину використовуються наступні типи даних: ID, VARCHAR, DATE. Найбільший розмір (100 символів)для типу VARCHAR вказано для рядка DESCRIPTION\_BOOK в таблиці BOOK. Найменший розмір 1 символів має поле GENDER у таблиці CUSTOMER.

Зображення, що містить текст, знімок екрана, число, програмне забезпечення

Автоматично згенерований опис

Рис.2.6

На рис.2.6. можна побачити, що при створенні таблиці CUSTOMER було використано декілька типів даних. Для всіх полів з типом VARCHAR, було задано максимальну кількість символів. Для числового типу INT не було задано обмежень.

**2.4. Обмеження цілісності даних**

Обмеження цілісності – це правила і засоби, що забезпечують надійну реалізацію встановлених зв’язків між даними що містяться в базі. Ці обмеження допомагають забезпечити цілісність даних, що означає, що дані в базі даних залишаються у валідному та очікуваному стані.

Обмеження цілісності даних є важливим етапом при роботі з базами даних. Такі обмеження забезпечують унікальність, цілісність, точність та надійність даних.

Ці обмеження накладаються на дані, що зберігаються в базі даних, і контролюють правильність введення та змінення даних, а також забезпечують унікальність, цілісність зв'язків між таблицями та інші логічні правила.

Основні типи обмежень цілісності даних включають:

* Обмеження первинного ключа (Primary Key Constraint): Це обмеження вимагає, щоб значення поля або комбінації полів у таблиці були унікальними і не могли містити значення NULL. Воно ідентифікує унікальний запис в таблиці і забезпечує посилання на нього з інших таблиць.
* Обмеження зовнішнього ключа (Foreign Key Constraint): Це обмеження встановлює зв'язок між полями двох таблиць. Воно вимагає, щоб значення поля або комбінації полів у таблиці були посиланнями на значення первинного ключа іншої таблиці.
* Обмеження унікальності (Unique Constraint): Це обмеження вимагає, щоб значення поля або комбінації полів у таблиці були унікальними. Воно не дозволяє дублікатів значень у визначеному полі або комбінації полів.
* Обмеження перевірки (Check Constraint): Це обмеження встановлює правила перевірки для значень в певному полі. Воно дозволяє вказати логічні вирази або функції, які обмежують діапазон допустимих значень у полі.
* Обмеження значення NULL (Null Constraint): Це обмеження визначає, чи допускається значення NULL у певному полі. Воно вказує, чи є поле обов'язковим для введення значення, або чи може містити значення NULL.
* Обмеження автоматичного значення (Default Constraint): Це обмеження встановлює значення за замовчуванням для поля, якщо значення не вказано при вставці нового запису.

Для того, що забезпечити введенню коректних і правильних даних, а також зробити зв’язок між таблицями та дотримання логічних правил в базі даних для інформаційної системи також було використано обмеження цілісності даних. В базі даних були використанні наступні обмеження:

* Первиний ключ
* Зовнішній ключ
* NOT NULL

**2.5. Реалізація SQL-скрипту**

SQL-скрипт - це послідовність команд або запитів, які використовують мову запитів SQL (Structured Query Language) для взаємодії з базою даних. SQL-скрипти використовуються для створення, модифікації та керування базами даних.

SQL-скрипти дозволяють виконувати різноманітні операції з базою даних, такі як створення таблиць, вставка, оновлення або видалення даних, визначення обмежень цілісності, створення індексів, зміна схеми бази даних та багато іншого. Вони є важливим інструментом для розробників та адміністраторів баз даних.

Для реалізації інформаційної системи книжкового магазину було використано такі види запитів: «SELECT», «DELETE», «UPDATE», «INSERT».

Для перегляд всіх книг, які належать до певного виду літратири можна за допомогою наступного запиту:

SELECT NAME FROM BOOK, CATEGORY WHERE CATEGORY\_ID=CATEGORY.ID AND TYPY\_OF\_BOOK IN ('Художня література').

Переглянути, скільки було продано кожну книгу в асортименті магазину в певному році за допомогою запиту:

SELECT NAME, COUNT(\*) AS COUNT\_BOOK FROM BOOK JOIN ORDERS ON BOOK\_ID=BOOK.ID JOIN CUCTOMER ON CATEGORY\_ID=CUCTOMER.ID WHERE YEAR(DATE\_ORDER)=2020 GROUP BY NAME ORDER BY COUNT\_BOOK DESC.

Для того щоб подивитись які книги купив покупець і скільки він заплати, було написано наступний запит до бази даних:

SELECT FIRSTNAME, LASTNAME, NAME, SUM(AMOUNT) AS TOTAL\_PRICE FROM CUCTOMER JOIN ORDERS ON CUSTOMER\_ID=CUCTOMER.ID JOIN BOOK ON BOOK\_ID=BOOK.ID WHERE CUCTOMER.ID=5 GROUP BY FIRSTNAME, LASTNAME, NAME.

Для перегляду рейтингу категорій потрібно виконати наступний запит:  
SELECT TYPY\_OF\_BOOK, SUM(NUMBERS) AS TOTAL\_NUMBERS FROM BOOK JOIN CATEGORY ON CATEGORY\_ID=CATEGORY.ID JOIN ORDERS ON BOOK.ID=BOOK\_ID GROUP BY TYPY\_OF\_BOOK ORDER BY TOTAL\_NUMBERS DESC

Для того, щоб перевірити, які клієнти купляли книги більше 5 разів було написано наступний запит:

SELECT FIRSTNAME, LASTNAME, COUNT(\*) AS ORDERS\_COUNTFROM CUCTOMER JOIN ORDERSON CUSTOMER\_ID=CUCTOMER.ID GROUP BY FIRSTNAME, LASTNAME HAVING COUNT(\*) > 5.

Також було написано запит для того щоб побачити скільки книг певного автора було продано в заданому році.

SELECT NAME, COUNT(\*) AS NUMBER\_ORDERS FROM AUTHOR JOIN BOOK ON AUTHOR\_ID=AUTHOR.ID JOIN ORDERS ON BOOK\_ID=BOOK.ID WHERE LASTNAME ='Костенко' AND YEAR(DATE\_ORDER)=2021 GROUP BY NAME.

Для того, щоб побачити скільки книжок було продано в кожне місто було написано наступний запит до бази даних:

SELECT CITY, COUNT(\*) AS ORDERS\_COUNT FROM BOOK JOIN ORDERS ON BOOK\_ID=ORDERS.ID JOIN CUCTOMER ON CUSTOMER\_ID=ORDERS.ID GROUP BY CITY.

Для того, щоб побачити книги які ніхто не купува потрібно зробити наступний запит:

SELECT NAME FROM BOOK WHERE NOT EXISTS( SELECT \* FROM ORDERS WHERE BOOK\_ID=BOOK.ID).

Вище наведенні запити здійсненні за допомогою використання JOIN та використання функцій SUM() та COUNT().

Також в процесі роботи з базою даних книжкового магазину було використано підзапити. Їх приклади наведено нижче.

Наступний запит дозволяє побачити список клієнтів, сума покупки який перевищує 1000 грн.

SELECT FIRSTNAME, LASTNAME, RESULT FROM

(SELECT FIRSTNAME, LASTNAME, SUM(AMOUNT) AS RESULT FROM CUCTOMER JOIN ORDERS ON CUSTOMER\_ID=CUCTOMER.ID GROUP BY FIRSTNAME, LASTNAME) AS RESULT WHERE RESULT>1000.

Щоб побачити статистику продажу книг з різних видавництв по рохах потрібно виконати такий запит.

SELECT YEAR\_ORDER, NAME, COUNT\_ORDERS, SUM\_ORDERS

FROM(SELECT YEAR(DATE\_ORDER) AS YEAR\_ORDER, PUBLICATION.NAME AS NAME, SUM(NUMBERS) AS [COUNT\_ORDERS], SUM(AMOUNT) AS [SUM\_ORDERS] FROM PUBLICATION JOIN BOOK ON PUBLICATION\_ID=PUBLICATION.ID JOIN ORDERS ON BOOK\_ID=BOOK.ID GROUP BY PUBLICATION.NAME, YEAR(DATE\_ORDER)) AS RESULT/

Також було написано запит до бази даних, щоб побачити скільки книг продав кожен продавець.

SELECT FIRSTNAME, LASTNAME, TOTAL FROM(SELECT FIRSTNAME, LASTNAME, SUM(NUMBERS) AS TOTAL FROM EMPLOYEE JOIN ORDERS ON EMPLOYEE\_ID=EMPLOYEE.ID GROUP BY FIRSTNAME, LASTNAME) AS RESULT.

Для того, щоб додати нового користувача зі сайту було здійснено підключення до бази даних і виконано запит.

$mysql = new mysqli('localhost','root','','website');

$mysql->query("INSERT INTO `customer`(`FIRSTNAME`,`LASTNAME`,`EMAIL`,`PASS\_WORD`,`GENDER`,`DATE\_BIRTH`,`CITY`,`PHONE\_NUMBERS`) VALUES ('$name', '$surname', '$email', '$password', '$gender','$dateOfBirth','$city','$phoneNumber')");

$mysql->close();

Для того щоб було виведено список усіх книг, що є асортименті і інформацію про них було виконано наступний запит до бази даних.

$query = "SELECT `book`.`ID`, `book`.`NAME`, `category`.`TYPE\_OF\_BOOK`, `category`.`LANGUAGE\_BOOK`, `category`.`FORMAT\_BOOK`, `author`.`FIRSTNAME`, `author`.`LASTNAME`, `publication`.`NAME` AS `PUBLICATION\_NAME`, `book`.`YEAR\_PUBLICATION`, `book`.`DESCRIPTION\_BOOK`, `book`.`PRICE` FROM `book` JOIN `category` ON `book`.`CATEGORY\_ID` = `category`.`ID` JOIN `author` ON `book`.`AUTHOR\_ID` = `author`.`ID` JOIN `publication` ON `book`.`PUBLICATION\_ID` = `publication`.`ID` ORDER BY `book`.`ID`"; $stmt = $pdo->prepare($query); $stmt->execute(); $customers = $stmt->fetchAll();

Також було написано запит, який дозволяє відобразити інформацію про всіх клієнтів магазину.

$query = "SELECT `ID`,`FIRSTNAME`, `LASTNAME`, `EMAIL`,`PASS\_WORD`, `GENDER`, `DATE\_BIRTH`, `CITY`, `PHONE\_NUMBERS` FROM `customer`"; $stmt = $pdo->prepare($query); $stmt->execute(); $customers = $stmt->fetchAll();

Для того щоб побачити інформацію про всіх працівників магазину виконується наступний запит:

$query = "SELECT `employee`.`ID`, `employee`.`FIRSTNAME`, `employee`. `LASTNAME`, `employee`.`POSITION`, `employee`.`DATE\_EMPLOYMENT`, `employee`.`RATE`, `office`.`CITY`, `office`.`STREET` FROM `employee` JOIN `office` ON `employee`.`OFFICE\_ID` = `office`.`ID` ORDER BY `employee`.`ID`";

$stmt = $pdo->prepare($query); $stmt->execute(); $customers = $stmt->fetchAll();

Запит наведений нижче дозволяє побачити інформацію про всі замовлення.

$query = "SELECT `orders`.`ID`, `book`.`NAME`, `employee`.`LASTNAME`, `customer`.`FIRSTNAME`, `customer`.`LASTNAME`, `orders`.`NUMBERS`, `orders`.`DATE\_ORDER`, `orders`.`AMOUNT` FROM `orders` JOIN `book` ON `orders`.`BOOK\_ID` = `book`.`ID` JOIN `employee` ON `orders`.`EMPLOYEE\_ID` = `employee`.`ID` JOIN `customer` ON `orders`.`CUSTOMER\_ID` = `customer`.`ID` ORDER BY `orders`.`ID`"; $stmt = $pdo->prepare($query); $stmt->execute(); $customers = $stmt->fetchAll();

**РОЗДІЛ 3. РОЗРОБКА ВЕБ-ДОДАТКУ**

* 1. **Структура веб-сайту та макет сторінок веб-сайту**

Структура сайту – це схема на якій прослідковується зв’язок між сторінками. Розрізняють внутрішню і зовнішню структуру сайту.

Зовнішня структура сайту включає розташування та організацію різних елементів і компонентів на веб-сторінках.

Основні елементи зовнішньої структури сайту включають:

* **Навігація**: Меню або навігаційна панель, яка допомагає користувачам переходити між різними сторінками сайту. Навігація може бути горизонтальною або вертикальною, залежно від дизайну.
* **Заголовок**: Заголовок сторінки, який зазвичай розміщується вверху сторінки і містить назву сайту або підзаголовок конкретної сторінки.
* **Банер або шапка:** Це область, яка зазвичай розміщується вгорі сторінки і містить важливу інформацію, таку як логотип, контактні дані або пошукову форму.
* **Контентна область**: Це основна частина сторінки, де розміщується вміст, такий як текст, зображення, відео, таблиці тощо. Контентна область може бути поділена на секції або блоки для кращої організації вмісту.
* **Бічна панель**: Це додаткова область, яка зазвичай розміщується на бічній стороні сторінки і містить додаткову інформацію, посилання, рекламу або інші елементи. Бічна панель може бути зліва або справа від контентної області.
* **Підвал**: Це область, яка зазвичай розміщується внизу сторінки і містить додаткову інформацію, посилання на соціальні мережі, копірайти або інші дії, такі як підписка на розсилку або зворотній зв'язок.

Зовнішня структура сайту може варіюватися залежно від його типу, цілей та вимог.

Структура головної сторінки сайту книжкового магазину зображена на рис. 3.1.

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис

Рис3.1

Головна сторінка сайту відкривається за адресою <http://localhost/index.html>. Зверху на сторінці є навігація та шапка профілю в якій розміщено логотип магазину. Нижче на сторінці можна побачити назву інтернет-магазину і запрошення його відвідати. Також на сторінці є інформації про магазин для відвідувачів. Прогорнувши сторінку до низи відвідувачі сайту можуть ознайомитись зі статистикою магазину і подивитись відео про книги. Сторінка закінчується футером. В ньому було розміщено посилання на соціальні мережі.

В навігацію є посилання на сторінку про нас. Перейшовши на неї користувач може подивитись інформацію про продавців, які працюють в магазині.

На сторінці книги відвідувачі сайту побачать пошук, в якому можуть ввести назву книги, яку шукають. Також на цій сторінці відображаються карточки товарів та корзина.

Перейшовши на сторінку контакти, користувачі побачать форму зворотного зв’язку. Прокрутивши в низ користувачі можуть ознайомитись з наступною інформацією: адресу магазинів, номера телефонів, та електроні адреси. Внизу сторінки знаходяться карти по яких можна перейти, щоб зрозуміти де розташовані магазини.

Також на сайті є сторінки входу та авторизації. Перейшовши на них користувачі побачать форми, які зможуть заповнити.

Адміністратор магазину також може перейти на сторінку авторизації. Після правильного вводу логіна та пароля відкриється нова сторінка на якій є змога вибрати, яку інформацію подивитись а саме: інформацію про книги, замовлення, клієнтів, працівників.

* 1. **Програмування серверної частини**

Програмування серверної частини – це створення ядра сайту, розробка платформи, основного функціоналу та адміністративної частини. Back-end є невидимою стороною для користувача, який пов’язаний з написання скриптів для сервера.

Основні аспекти програмування серверної частини включають:

* Вибір технології для розробки серверної частини залежить від ваших вимог, навичок та уподобань. Серед популярних технологій та мов програмування для розробки серверної частини веб-додатків також варто згадати PHP. PHP є популярною мовою програмування з великою спільнотою розробників та багатим екосистемою фреймворків та бібліотек. Вибір PHP для розробки залежить від вашого досвіду, вимог проекту та доступності ресурсів для розробки та підтримки.
* Роутинг та маршрутизація: Цей аспект включає визначення маршрутів для обробки різних запитів веб-додатку. Наприклад, ви можете визначити маршрути для обробки запитів GET, POST, PUT або DELETE на певному URL-адресі.
* Обробка запитів: Цей аспект включає обробку запитів, що надходять від клієнта. Можна отримувати дані з запиту, перевіряти їх, виконувати дії, які вимагаються (наприклад, зберігання в базі даних), та генерувати відповідь для клієнта.
* Взаємодія з базою даних: Багато серверних додатків вимагають зберігання та отримання даних з бази даних. Ви можете використовувати SQL або ORM (Object-Relational Mapping) для взаємодії з базою даних та виконання операцій, таких як створення, читання, оновлення та видалення записів.
* Бізнес-логіка: Бізнес-логіка веб-додатку визначає логіку та правила, які виконуються на сервері. Це може включати перевірку прав доступу, обробку даних, валідацію, обчислення та інші операції, які відповідають за функціональність додатку.
* API та взаємодія з клієнтом: Якщо ваш веб-додаток надає API для взаємодії з клієнтами, ви можете розробити API ендпоінти, які надають доступ до даних та функціональності сервера. Це може включати формати відповідей, такі як JSON або XML, та обробку запитів REST або інших протоколів.

Для створення серверної частини було використано мову PHP. Нижче наведений приклад використання цієї мови для створення форми зворотного зв’язку.

<?php

if ($\_SERVER["REQUEST\_METHOD"] === "POST") {

// Отримати дані з форми

$first\_name = $\_POST["first\_name"];

$last\_name = $\_POST["last\_name"];

$email = $\_POST["email"];

$message = $\_POST["message"];

$to = "hrundatania@gmail.com";

$subject = "Нове повідомлення з форми зворотнього зв'язку";

$body = "Ім'я: $first\_name\nПрізвище: $last\_name\nЕлектрона адреса: $email\nПовідомлення: $message";

if (mail($to, $subject, $body)) {

echo "Ваше повідомлення надіслано успішно.";

} else {

echo "Під час надсилання повідомлення сталася помилка.";

}

}

?>

За допомогою наведено вище наведеного коду реалізується відправлення форми зворотнього зв’язку на електрону пошту. Якщо все пройшло успішно видається повідомлення про відправлення листа. У випадку якщо повідомлення не надіслано буде відображвтися повідомлення про помилку.

На початку цього коду перевіряється чи насправді метод запиту є POST. Пізніше використовуючи змінні $ = $\_POST[""] отримуються дані з форми. Також відбуваються налаштування електронної пошти. Вказується отримувач ($to), тема листа ($subject) та тіло повідомлення ($body), яке містить дані з форми.

Також за допомогою PHP відбувається з’єднання з базою даних. Приклад коду наведено нижче.

<?php

$host = 'localhost';

$db = 'website';

$user = 'root';

$pass = 'root';

try {

$dsn = "mysql:host=$host;dbname=$db";

$pdo = new PDO($dsn, $user, $pass);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$pdo->setAttribute(PDO::ATTR\_DEFAULT\_FETCH\_MODE, PDO::FETCH\_ASSOC);

// Ваш код для роботи з базою даних

} catch (PDOException $e) {

echo "Помилка підключення до бази даних: " . $e->getMessage();

}

На початку коду вказуються дані для з’єднання з базою даних. Пізніше налаштовуються додаткові параметри PDO. Якщо підключення не відбулось виводиться повідомлення про помилку.

**3.3. Програмування клієнтської частини**

Програмування клієнтської частини включає розробку коду, який виконується на браузері або на клієнтському пристрої і взаємодіє з користувачем та серверною частиною. При програмуванні клієнтської частини важливо враховувати зручність використання, швидкість виконання, кросс-браузерну сумісність та безпеку. Також слід враховувати особливості цільових платформ, наприклад мобільні пристрої, та оптимізувати код для покращення продуктивності та користувацького досвіду.

В даній роботі було написано багато коду для створення клієнтської частини. Для прикладу розглянемо код для створення карточки товару.

<div class="card" data-id="2">

<img class="product\_img" src="img/book2.jpg" alt="Не озирайся і мовчи" />

<div class="content">

<div class="row">

<div class="card2">

<span>Не озирайся і мовчи</span>

</div>

<div class="details-wrapper">

<div class="items">

<div class="items\_\_control" data-action="minus">-</div>

<div class="items\_\_current" data-counter>1</div>

<div class="items\_\_control" data-action="plus">+</div>

</div>

<div class="price">465 ₴</div>

</div>

</div>

<button data-cart type="button" class="buttons">В корзину</button>

</div>

</div>

Даний код відповідає за розташування елементів на сторінці. В ньому використовуються декілька елементів для розташування елементів на сайті: <div>, <span>, < img > .

Використання HTML-коді класів, дозволило встановити спільні стилі до однакових об’єктів та маніпулювати ними за допомогою JavaScript. Приклад CSS-коду наведено нижче.

.btn {

width: 180px;

height: 50px;

margin-left: auto;

margin-right: 70px;

margin-top: 90px;

background-color: rgb(215, 230, 5);

border: none;

border-radius: 40px;

text-transform: uppercase;

font-size: 25px;

}

.btn:hover {

background-color: transparent;

color: rgb(215, 230, 5);

background-color: #0f0e0e;

}

За допомогою цього коду налаштовуються стилі для кнопки «Замовити». В ньому вказується колір кнопки, розміри, стиль тексту всередині кнопки. Також в наведеному коді прописано, як буде змінюватись кнопки при наведенні на неї.

Для того щоб сайт працював правильно, для його написання було використано JavaScript. Для прикладу наведено реалізацію пошуку на сайті за допомогою цієї мови.

<script type="text/javascript">

const elastic = document.querySelector("#elastic");

elastic.addEventListener("input", (ev) => {

const value = ev.target.value.trim();

const elasticItems = document.querySelectorAll(".card");

const searchRegExp = new RegExp(value, "gi");

if (value === "") {

elasticItems.forEach((el) => {

el.classList.remove("hide");

});

return;

}

elasticItems.forEach((el) => {

const innerCard = el.querySelector(".card2");

const elementText = innerCard.textContent;

const isContainsSearchRequest = searchRegExp.test(elementText);

if (!isContainsSearchRequest) {

el.classList.add("hide");

} else {

el.classList.remove("hide");

}

});

});

</script>

Також за допопомогою JavaScript було реалізовано перехід на інші сторінки. Приклад наведено нижче.

const button = document.querySelector(".btn\_nav");

button.addEventListener("click", function () {

// Перейти на іншу сторінку

window.location.href = "sign.html";

});

**3.4 Розміщення веб-сайту на локальному сервері**

Для розміщення веб-сайту було використано локальне середовище XMAPP.

Зображення, що містить текст, знімок екрана, програмне забезпечення, Комп’ютерна піктограма

Автоматично згенерований опис

Рис.3.2.

XAMPP є безкоштовним пакетом веб-сервера, який включає в себе необхідне програмне забезпечення для розробки та розгортання веб-сайтів на локальному сервері.

1. Apache: Веб-сервер, який обробляє запити від клієнтів та надає веб-сторінки.

2. MySQL: Система управління базами даних, яка дозволяє зберігати та керувати даними для вашого веб-сайту.

3. phpMyAdmin: Веб-інтерфейс для керування базами даних MySQL.

XAMPP надає зручну установку та налаштування цих компонентів і дозволяє швидко розпочати роботу з розробкою та тестуванням веб-сайтів на локальному сервері.

XAMPP є популярним вибором для розробників, які працюють над веб-проектами на локальному сервері, оскільки він простий у використанні та має широкий функціонал для розробки, тестування та налагодження веб-додатків.