МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Циклова комісія *комп'ютерних систем та мереж*
(повна назва циклової комісії)

Допустити до захисту
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
_____ *Ірина КРАВЧУК*
(підпис) (ім'я, ПРІЗВИЩЕ)
« 16 » 06 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-ПРОФЕСІЙНОГО СТУПЕНЯ ФАХОВИЙ МОЛОДШИЙ БАКАЛАВР

Тема: *Створення бази даних для управління бібліотекою з використанням SQL*

Група: *3-013* Спеціальність: *123 «Комп'ютерна інженерія»*

Здобувач освіти _____ *Владислав ПИСЬМЕННИЙ*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Олександр ГРИНЧЕНКО*
(підпис) (ім'я, ПРІЗВИЩЕ)

Консультант з оформлення пояснювальної записки _____ *Оксана ОСАДЧА*
(підпис) (ім'я, ПРІЗВИЩЕ)

Кривий Ріг 2025 р.

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Відділення *комп'ютерної та програмної інженерії*
Циклова комісія *комп'ютерних систем та мереж*
Освітньо-професійний ступінь *фаховий молодший бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
_____ *Ірина КРАВЧУК*
(підпис) (ім'я, ПРІЗВИЩЕ)
« *01* » *03* 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

Письменний Владислав Сергійович
(прізвище, ім'я, по батькові)

1.Тема роботи *Створення бази даних для управління бібліотекою з використанням SQL*

Керівник роботи *Гринченко Олександр Сергійович, викладач вищої категорії*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по коледжу від « *04* » *04* 2025 року № *50-ст*

2. Строк подання здобувачем освіти роботи з _____ по _____

3. Вихідні дані до роботи *Об'єкти бази даних, СУБД Microsoft SQL, SQL запити, адміністрація бази даних*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз предметної області, проектування бази даних, реалізація бази даних у СУБД, тестування і оптимізація

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
ER-діагрма бази даних, SQL-запити для створення та тестування бази даних

6. Консультанти розділів роботи (проекту)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Узгодження технічного завдання*	*02.05 – 07.05*	*Виконано*
2	*Огляд літератури по темі кваліфікаційної роботи*	*08.05 – 13.05*	*Виконано*
3	*Аналіз предметної області*	*14.05 – 19.05*	*Виконано*
4	*Проектування бази даних*	*20.05 – 25.05*	*Виконано*
5	*Реалізація бази даних у СУБД*	*26.05 – 31.05*	*Виконано*
6	*Тестування і оптимізація*	*01.06 – 06.06*	*Виконано*
7	*Оформлення пояснювальної записки*	*07.06 – 12.06*	*Виконано*
8	*Захист кваліфікаційної роботи*	*20.06 – 21.06*	*Виконано*

Здобувач освіти _____ *Владислав ПИСЬМЕННИЙ*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Олександр ГРИНЧЕНКО*
(підпис) (ім'я, ПРІЗВИЩЕ)

Звіт подібності

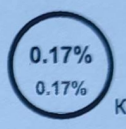
метадані

Назва організації
Ukrainian national aviation university
Заголовок
Письмений В_3-013_2025_КПІ
Автор Науковий керівник / Експерт
ПисьменийГринченко О
підрозділ
Криворізький Фаховий коледж

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1



КЦ

25	10281	78449
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки		0
Парафрази (SmartMarks)	a	17

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Аль-фітурі Асія фб-31 курсова робота по БД [redacted] 12/21/2024 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute course papers (ФТІ, К-ра інформаційної безпеки)	18 0.18 %
2	Аль-фітурі Асія фб-31 курсова робота по БД [redacted] 12/21/2024 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute course papers (ФТІ, К-ра інформаційної безпеки)	16 0.16 %

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих систем управління бібліотеками

Автоматичні бібліотечні системи відіграють ключову роль у сучасному бібліотечному менеджменті, значно спрощуючи та підвищуючи ефективність роботи з бібліотечними фондами. Ці системи забезпечують інтегрований інструмент для обліку літератури, читачів та пов’язаних даних, що дозволяє бібліотекам функціонувати в режимі, який відповідає вимогам сучасного інформаційного суспільства. З розвитком цифрових технологій, автоматизація бібліотечних процесів стає не просто зручністю, а необхідністю для забезпечення конкурентоспроможності бібліотек.

Основна мета автоматизації бібліотечних процесів полягає в оптимізації всіх ключових аспектів бібліотечної діяльності. Це охоплює процеси видачі та повернення книг, управління наявністю фонду, а також підготовку звітності про діяльність бібліотеки. Одним з найпоширеніших засобів для досягнення цих цілей є різноманітні програмні продукти, які включають в себе такі важливі функції:

Облік книг: Автоматичні системи надають можливість зберігати та управляти інформацією про літературні видання. Це не лише назви книг, але й імена авторів, дані видавництв, роки публікації, тематику та інші важливі дані. Завдяки цій інформації бібліотекарі можуть ефективно використовувати дані про фонди, швидко знаходити необхідні видання та аналізувати популярність тих чи інших книг. У результаті, бібліотеки отримують можливість впроваджувати більш цілеспрямовані стратегії щодо формування своїх фондів, що, безумовно, впливає на якість обслуговування читачів.

Адміністративне управління користувачами: Сучасні автоматизовані системи дозволяють бібліотекам реєструвати нових користувачів, таких як учні, викладачі та інші, збирати їхні дані, а також встановлювати права

доступу до бібліотечних ресурсів. Це сприяє створенню більш персоналізованого та зручного обслуговування для читачів, адже кожен користувач отримує можливість отримати доступ до необхідних йому матеріалів. Завдяки автоматизації цього процесу, бібліотеки можуть підтримувати актуальність інформації про користувачів, що в свою чергу впливає на якість надання послуг.

Процеси видачі та повернення книг: Автоматизація цих процесів дозволяє знизити навантаження на бібліотечний персонал, зменшуючи конкуренцію в процесі видачі книг та контролюючи повернення літератури. Автоматизовані системи забезпечують простоту у відстеженні термінів повернення, а також нарахування штрафів за порушення термінів, що підвищує дисципліну серед читачів. Таким чином, бібліотеки мають можливість зменшити кількість затримок у поверненні книжок, що дозволяє підтримувати високу якість обслуговування.

Ведення звітів і статистичних даних: Збір різноманітних типів звітів, які включають аналіз популярності книг, діяльності користувачів, а також обробку запитів на бронювання літератури, надає бібліотекам можливість краще розуміти потреби своїх читачів. Це дозволяє оптимізувати свою діяльність, плануючи закупівлю нових видань відповідно до запитів користувачів, а також адаптувати інформаційні послуги до актуальних тенденцій. Регулярний аналіз статистичних даних також сприяє виявленню нових напрямків для розвитку бібліотечних послуг.

Завдяки новітнім інформаційним системам, які використовуються в бібліотеках, значно підвищується ефективність та якість їх роботи. Аналіз даних дозволяє оптимізувати діяльність, зменшує навантаження на працівників бібліотек, забезпечує швидкий доступ до необхідної інформації та надає можливість працювати в режимі реального часу. Це, в свою чергу, сприяє підвищенню конкурентоспроможності бібліотек у сучасному інформаційному просторі.

Серед програмних засобів, які активно використовуються для автоматизації бібліотечної діяльності, можна виділити такі системи:

– ***Aleph*** – це потужна та розповсюджена система для автоматизації бібліотечних операцій. Вона підтримує всі основні функції для обліку книг, управління інформацією про користувачів та оптимізації процесів роботи бібліотеки. Система Aleph дозволяє створювати електронні каталоги, здійснювати точний пошук книг, автоматизувати формування цін на бібліотечні матеріали та готувати різноманітні аналітичні звіти [1].

– ***Koha*** – вільне програмне забезпечення для автоматизації бібліотечних процесів, яке дозволяє ефективно здійснювати облік бібліотечних матеріалів, управляти картками користувачів, а також організовувати процеси видачі та повернення книг. Koha інтегрується з іншими програмними засобами та інтернет-каталогами, що надає їй високий рівень гнучкості та адаптивності до змін. Ця система також підтримує різноманітні мовні інтерфейси, що робить її доступною для користувачів з різними мовними уподобаннями [2].

– ***OpenBiblio*** – це система з відкритим кодом, яка призначена для автоматизації бібліотечної діяльності середнього та малого рівня. Її простий та інтуїтивно зрозумілий інтерфейс значно полегшує навчання бібліотекарів, дозволяючи швидко освоїти функції обліку бібліотечних фондів та ефективного управління ними. OpenBiblio також дозволяє інтегрувати бібліотечні дані з веб-сервісами, що розширює можливості доступу до інформації [3].

– ***LIBRARY 2.0*** – сучасна інноваційна система, яка активно застосовується в Україні, зокрема в навчальних закладах та шкільних бібліотеках. Вона автоматизує процеси обліку бібліотечних матеріалів та створює електронні каталоги, забезпечуючи високий рівень ефективності збору та аналізу статистичних даних. Ця система сприяє розвитку електронних ресурсів, що є особливо важливим в умовах зростаючої потреби в цифровій інформації.

Використання цих автоматизованих систем значно полегшує роботу бібліотекарів, оптимізує управлінські процеси та забезпечує зручний доступ до необхідної інформації. Це, в свою чергу, сприяє підвищенню загального рівня обслуговування користувачів бібліотек. Завдяки технологічним інноваціям, бібліотеки стають більш відкритими та доступними для всіх верств населення, що сприяє розвитку культури читання та освіти в суспільстві. В умовах стрімкого розвитку технологій, автоматизація бібліотечної діяльності стає запорукою успіху і надійності бібліотек у виконанні їх місії щодо забезпечення доступу до знань та інформації. Зокрема, вона відкриває нові можливості для інтерактивних послуг, таких як віртуальні консультації, електронні ресурси та інтеграція з соціальними мережами, що лише підвищує значимість бібліотек у суспільстві.

1.2 Особливості та вимоги до шкільної бібліотеки

Шкільна бібліотека є важливим структурним підрозділом навчального закладу, що виконує ключову роль у забезпеченні учнів та вчителів необхідними знаннями і ресурсами для навчального процесу. Вона не тільки надає доступ до різноманітної літератури, але й відіграє важливу роль у формуванні читацької культури серед учнів, сприяючи їхньому розвитку як самостійних мислячих особистостей. У сучасних умовах, коли інформаційні технології активно впроваджуються в усі сфери життя, автоматизація бібліотечних процесів стає невід’ємною складовою функціонування шкільної бібліотеки.

Розробка бази даних для автоматизації бібліотеки має враховувати ряд специфічних характеристик, які суттєво впливають на її ефективність. Відмінності від загальних бібліотечних систем можна розглядати через кілька основних аспектів, зокрема, типи користувачів, типи літератури, обмежений бюджет, інтеграцію з навчальним процесом, систему обліку та моніторинг, доступність для учнів і можливість звітності та аналізу.

Типи користувачів: У шкільній бібліотеці основними користувачами є учні та вчителі. Ця особливість вимагає створення багаторівневої системи доступу, яка враховує різні потреби кожної групи користувачів. Наприклад, учні можуть мати обмеження на кількість книг, які вони можуть взяти на певний термін, тоді як вчителі повинні мати можливість доступу до більшого обсягу ресурсів з метою підготовки уроків і професійного розвитку. Це вимагає впровадження функцій, які б автоматично масштабували доступ на основі ролі користувача. Таке розмежування дозволить створити більш організовану і продуктивну бібліотечну систему, що зможе задовольнити потреби всіх користувачів.

Типи літератури: Шкільна бібліотека повинна акцентувати увагу не лише на художній літературі, а й на навчальних, методичних матеріалах, підручниках і довідкових виданнях. База даних повинна мати можливість класифікувати книги за предметами, класами і видами літератури, а також включати облік навчальних посібників і методичних матеріалів, специфічних для вчителів. Це забезпечить зручність пошуку і оптимізує процес роботи з літературою. Важливо, щоб дані про нові надходження були доступні в режимі реального часу, що дозволить вчителям і учням швидко знаходити необхідні ресурси для навчання.

Обмежений бюджет: Бібліотеки у навчальних закладах зазвичай стикаються з обмеженим бюджетом на закупівлю нових книг. Отже, важливо, щоб база даних включала інформацію про книги, що знаходяться в аварійному або застарілому стані, а також підтримувала функцію резервування книг. Це дозволить бібліотекарям ефективніше управляти ресурсами і оптимізувати процес оновлення фонду бібліотеки. Система також повинна дозволяти відслідковувати запити на нові надходження, що допоможе в плануванні закупівель.

Інтеграція з навчальним процесом: Шкільна бібліотека безпосередньо пов’язана з навчальним процесом. Вчителі часто використовують бібліотеку для підготовки до уроків або для проведення позакласних заходів. Система

повинна забезпечувати можливість швидкого і зручного пошуку літератури, яка може бути корисною для конкретних предметів або тем уроків. Це може включати інтеграцію з навчальними планами і програмами, а також вбудовані рекомендаційні функції, що пропонують ресурси на основі запитів вчителів і учнів.

Система обліку та моніторинг: Моніторинг термінів повернення книг та нарахування штрафів за прострочення є важливим аспектом бібліотечного обліку. База даних повинна автоматично обчислювати терміни повернення, відстежувати прострочені книги та формувати відповідні звіти для адміністрації школи. Це дозволить вести контроль за книгами і своєчасно реагувати на ситуації, що потребують уваги. Реалізація системи автоматичного нагадування для користувачів про терміни повернення може також зменшити кількість прострочень.

Доступність для учнів: Бібліотечна система повинна бути простою і зручною у використанні, щоб учні різного віку могли легко знаходити потрібні ресурси. Інтерфейс користувача повинен бути інтуїтивно зрозумілим, а функції пошуку — швидкими та ефективними. Це сприятиме більш активному використанню бібліотеки учнями та покращить загальний рівень читацької активності. Запровадження мобільних додатків або онлайн-платформ може також розширити доступ до бібліотечних ресурсів, дозволяючи учням шукати та бронювати книги з будь-якої точки.

Можливість звітності та аналізу: Шкільна бібліотека повинна мати можливість формувати звіти про використання книг, популярність літератури серед учнів, статистику нарахування штрафів, а також оцінку бібліотечних ресурсів для подальшого поповнення фонду. Це дозволить адміністрації бібліотеки аналізувати потреби користувачів, адаптувати фонд та вживати заходів для покращення обслуговування. Регулярний аналіз даних про використання ресурсів допоможе виявити популярні тенденції та вчасно реагувати на зміни в інтересах учнів.

Таким чином, унікальні аспекти функціонування шкільної бібліотеки

вимагають розробки спеціалізованої бази даних, яка зможе ефективно впоратися з цими викликами та забезпечити максимальну підтримку навчального процесу.

Сучасні вимоги до інформаційних систем вимагають ретельного підходу до проектування бази даних, оскільки правильна організація даних забезпечить зручність і ефективність роботи як бібліотекарів, так і учнів. У цьому контексті важливо врахувати кілька ключових аспектів, що формують основу для функціонування бібліотечної системи а також аналіз ефективності використання бібліотечних фондів. Це дає можливість адміністрації школи отримувати оперативну інформацію про стан бібліотеки, визначати потребу в оновленні фонду, планувати закупівлю нових видань та приймати обґрунтовані рішення щодо подальшого розвитку бібліотечних послуг. Регулярне ведення такої звітності допомагає не лише покращити якість обслуговування користувачів, а й забезпечити прозорість у використанні матеріальних та інформаційних ресурсів навчального закладу.

Таким чином, система автоматизації шкільної бібліотеки повинна враховувати специфіку навчального середовища, бути інтегрованою з освітнім процесом, мати простий і доступний інтерфейс, гнучкі можливості обліку та контролю, а також функції для формування статистичних звітів і аналізу. Усі ці вимоги необхідно враховувати під час проєктування бази даних, яка стане основою для автоматизованої бібліотечної системи у навчальному закладі.

1. Простота користування та доступність інтерфейсу

Однією з найважливіших вимог до бази даних є простота користування. Інтерфейс повинен бути зручним і інтуїтивно зрозумілим для всіх категорій користувачів, незалежно від їхнього рівня підготовки. Це передбачає наявність зрозумілих меню та навігаційних елементів, що дозволяють швидко знаходити потрібну інформацію. Користувачі повинні мати можливість оперативно здійснювати пошук книг за різними критеріями, такими як назва, автор або жанр. Інтерфейс також повинен бути адаптований під потреби різних груп користувачів: учнів, вчителів, бібліотекарів та адміністрації.

2. Функціональність обліку книг

База даних повинна містити велику кількість інформації про всі книги, які є в бібліотеці. Ключові характеристики, які слід врахувати, включають назву книги, автора, рік видання, жанр або категорію, кількість доступних примірників та стан книги (нове, пошкоджене, відсутнє). Крім того, важливо забезпечити можливість оновлення даних у разі поповнення фонду або заміни книг, що дозволить підтримувати базу в актуальному стані.

3. Управління користувачами

Ефективне управління користувачами є основою для успішної роботи бібліотеки. База даних повинна дозволяти реєстрацію різних типів користувачів, таких як учні, вчителі та бібліотекарі, з чітко визначеними рівнями доступу. Учні повинні мати можливість шукати та замовляти книги, вчителі – додавати книги до своїх списків вимог для уроків і отримувати пріоритет у видачі навчальних матеріалів, а бібліотекарі – редагувати записи про книги, управляти обліком та нараховувати штрафи за прострочення.

4. Механізм видачі та повернення книг

Автоматизація процесу видачі та повернення книг є критично важливим аспектом. База даних повинна фіксувати дату видачі книги користувачу, автоматично обчислювати терміни повернення та нараховувати штрафи за прострочення. Також важливо забезпечити можливість резервування книг, які тимчасово недоступні, що дозволить користувачам планувати свої дії.

5. Формування звітів та статистики

Для аналізу діяльності бібліотеки база повинна підтримувати функцію генерації звітів. Це дозволить отримувати дані про кількість виданих книг за певний період, прострочені книги та нараховані штрафи, а також статистику популярності книг та авторів серед користувачів. Аналіз цих даних допоможе бібліотекарям приймати обґрунтовані рішення щодо оновлення фонду.

6. Безпека та захист даних

Безпека даних є невід’ємною частиною роботи сучасної бібліотеки. База даних повинна забезпечувати надійний захист від несанкціонованого доступу.

Це включає визначення прав доступу для різних категорій користувачів, забезпечення конфіденційності даних користувачів та книг, а також регулярне резервне копіювання даних для уникнення їх втрати.

7. Масштабованість та адаптивність системи

Важливим аспектом є масштабованість та адаптивність системи до майбутніх змін. Система повинна бути гнучкою, щоб підтримувати можливість додавання нових функцій, таких як інтеграція з іншими шкільними інформаційними системами або платформами для електронного резервування книг, що дозволить постійно вдосконалювати бібліотечний сервіс.

8. Підтримка багатомовності

З огляду на різноманітність мовних груп, які можуть користуватися послугами бібліотеки, підтримка багатомовності інтерфейсу є важливою вимогою. Це забезпечить доступність бази даних для всіх учнів, незалежно від їхньої рідної мови.

Таким чином, розробка бази даних для шкільної бібліотеки є комплексним завданням, що вимагає врахування багатьох аспектів. Задовольняючи ці вимоги, можна створити ефективну та зручну бібліотечну систему, яка задовольнить потреби всіх її користувачів.

З урахуванням зазначених характеристик, бібліотечна система стане потужним інструментом у формуванні читацької культури та розвитку знань серед учнів та вчителів, сприяючи їхньому успіху в навчанні і розвитку.

РОЗДІЛ 2
ПРОЄКТУВАННЯ БАЗИ ДАНИХ

2.1 Формування технічних та функціональних вимог

На основі аналізу предметної області та існуючих бібліотечних систем було сформовано перелік технічних та функціональних вимог до бази даних для шкільної бібліотеки.

Функціональні вимоги:

- реєстрація та збереження інформації про книги (назва, автор, рік видання, видавництво, кількість примірників);
- ведення обліку читачів (ПІБ, клас, дата народження, контактні дані);
- облік операцій видачі та повернення книг із зазначенням дат і відповідальних осіб;
- можливість пошуку книг за різними параметрами (назва, автор, рік видання, предмет, клас);
- формування звітів про видані книги, залишок на складі, популярність літератури;
- резервне копіювання та відновлення бази даних [4].

Технічні вимоги:

- база даних має бути створена на основі реляційної СУБД;
- підтримка мови *SQL* для роботи із запитами;
- можливість роботи у локальній мережі школи без доступу до Інтернету;
- мінімальні системні вимоги до клієнтських ПК (*Windows* 10, 4 ГБ ОЗП, процесор *Intel i3* та вище);
- забезпечення захисту даних шляхом надання різних рівнів доступу користувачам (адміністратор, бібліотекар, читач) ;

- можливість розширення структури бази даних у разі збільшення книжкового фонду або появи нових категорій користувачів.

Таким чином, база даних має бути простою в адмініструванні, адаптованою для використання в умовах шкільної бібліотеки та мати достатній функціонал для виконання основних облікових операцій.

2.2 Визанчення сутностей

У сучасному світі інформаційних технологій проєктування бази даних для шкільної бібліотеки є важливим етапом, який потребує детального планування та врахування всіх аспектів, пов’язаних із зберіганням та обліком бібліотечних ресурсів. Це не лише сприяє ефективному управлінню фондами книг, але й забезпечує зручність для користувачів, які звертаються до бібліотеки за необхідними матеріалами. У цьому підрозділі проаналізуємо основні об’єкти бази даних, їх характеристики та атрибути, що повинні бути враховані під час проєктування.

2.2.1 Книга

Основний об’єкт — таблиця ***Books***, яка містить дані про книги. Вона пов’язана з таблицями ***Authors***, ***Categories*** та ***BookCopies*** для зберігання додаткової інформації. Основні атрибути:

- *BookID* — унікальний ідентифікатор книги.
- *Title* — офіційна назва книги.
- *AuthorID* — посилання на таблицю **Authors**.
- *CategoryID* — посилання на таблицю **Categories**.
- *Year* — рік видання.
- *Publisher* — назва видавництва.

Додаткова інформація про кількість примірників та їх статус зберігається в таблиці ***BookCopies***.

2.2.2 Автор

Таблиця ***Authors*** містить інформацію про авторів книг. Основні атрибути:

- *AuthorID* — унікальний ідентифікатор автора.
- *FullName* — повне ім’я автора.

Це дозволяє системі зручно виконувати пошук літератури за авторством.

2.2.3 Категорія книги

Таблиця **Categories** визначає жанр чи напрямок літератури. Атрибути:

- *CategoryID* — унікальний ідентифікатор категорії.
- *Name* — назва категорії.
- *Description* — опис.

Це забезпечує структурування фонду за типами літератури.

2.2.4 Примірник книги

Таблиця ***BookCopies*** зберігає інформацію про кожен конкретний екземпляр книги. Атрибути:

- *CopyID* — унікальний ідентифікатор примірника.
- *BookID* — посилання на книгу з ***Books***.
- *InventoryNumber* — інвентарний номер.
- *Status* — стан книги: в наявності, видано, втрачено.

Це дозволяє бібліотеці ефективно керувати фізичними примірниками.

2.2.5 Користувач

Таблиця ***Users*** містить інформацію про відвідувачів бібліотеки. Атрибути:

- *UserID* — унікальний ідентифікатор користувача.
- *FullName* — повне ім’я.
- *BirthDate* — дата народження.
- *Phone* — телефон.

- *Role* — категорія користувача: учень, учитель, бібліотекар.
- *ClassID* — посилання на таблицю ***Classes*** (для учнів).

Це дає змогу відслідковувати, хто бере книги та керувати правами доступу.

2.2.6 Клас

Таблиця **Classes** використовується для обліку шкільних класів. Атрибути:

- *ClassID* — унікальний ідентифікатор.
- *ClassName* — назва класу (наприклад, «7-А»).

Використовується для зв’язку з учнями.

2.2.7 Видача книги

Таблиця ***IssuedBooks*** містить записи про видачу книг користувачам. Атрибути:

- *IssueID* — унікальний ідентифікатор видачі.
- *CopyID* — посилання на примірник книги.
- *UserID* — посилання на користувача.
- *IssueDate* — дата видачі.
- *ReturnDate* — запланована дата повернення.
- *ActualReturnDate* — фактична дата повернення (якщо вже повернуто).

Зв’язки між сутностями:

- Один-до-багатьох (1:M) між таблицею *Книга* та *Видача книги*, оскільки одна книга може видаватися багато разів.
- Один-до-багатьох (1:M) між таблицею *Читач* та *Видача книги*, оскільки один читач може брати багато книг [5].

На цьому етапі визначення сутностей та їх атрибутів дозволяє перейти до побудови ER-діаграми, яка наочно відобразить структуру та взаємозв’язки елементів бази даних.

2.3 *ER*-діаграма

На основі визначених сутностей та їх атрибутів було створено ***ER*-діаграму**, яка наочно відображає структуру взаємозв'язків між основними об'єктами бази даних шкільної бібліотеки. Ця діаграма дозволяє візуалізувати логічну модель системи та показати зв'язки між таблицями.

У системі передбачені такі основні сутності:

- ***Users*** — користувачі бібліотеки (учні, вчителі)
- ***Books*** — інформація про книги
- ***Categories*** — категорії літератури
- ***Authors*** — автори книг
- ***IssuedBooks*** — журнал видачі книг
- ***Classes*** — шкільні класи (для прив'язки користувачів-учнів)
- ***BookCopies*** — екземпляри книг

Зв'язки між сутностями:

- ***Users*** пов'язується з ***Classes*** через поле *class_id*, оскільки учень належить до певного класу.
- ***Books*** пов'язується з ***Authors*** через поле *author_id* та з ***Categories*** через *category_id*.
- ***Books*** має зв'язок «один до багатьох» з ***BookCopies***, оскільки одна книга може мати декілька екземплярів.
- ***IssuedBooks*** пов'язується з ***Users*** через *user_id* та з ***BookCopies*** через *copy_id*, що дозволяє фіксувати видачу конкретного екземпляра книги певному користувачу.
- ***IssuedBooks*** також містить інформацію про дати видачі, повернення та можливі штрафи.

Опис зв'язків:

- ***One-to-Many*** **(1:N)** — одна категорія може містити багато книг, один автор може написати багато книг, одна книга може мати багато примірників.

– ***Many-to-One*** **(N:1)** — багато користувачів можуть належати до одного класу.

– ***Many-to-Many*** **(N:M)** — реалізується через зв’язок видачі книг, де один користувач може отримувати багато книг, а одна книга може бути видана багатьом користувачам (через різні екземпляри).

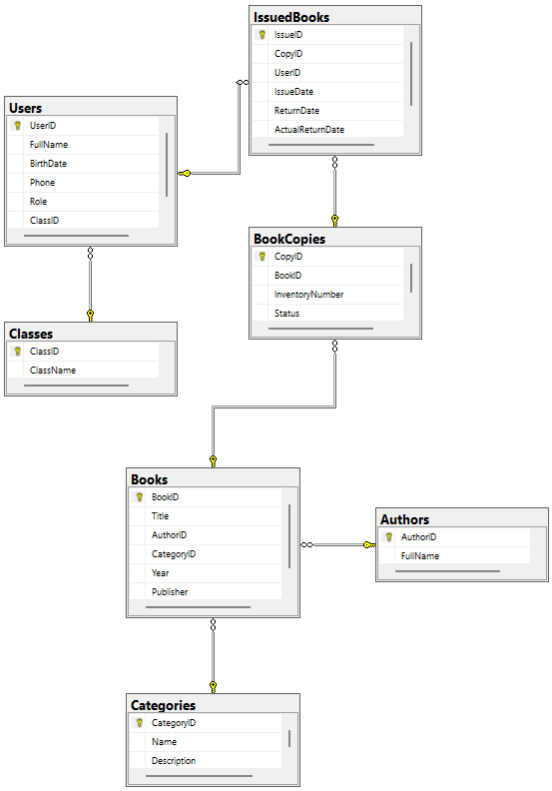


Рисунок 2.1 – *ER*-модель

2.4 Нормалізація бази даних

Процес нормалізації бази даних є необхідним етапом проєктування, який дозволяє усунути надлишковість даних, запобігти аномаліям при оновленні, вставці та видаленні записів, а також забезпечити логічну узгодженість структури даних. Нормалізація бази даних здійснюється шляхом послідовного приведення її до нормальних форм.

У даному проєкті для шкільної бібліотеки нормалізація виконувалася до третьої нормальної форми , що забезпечує оптимальне структурування даних без зайвої дубляції.

Перша нормальна форма

Перша нормальна форма вимагає, щоб:

- усі таблиці мали унікальні назви стовпців;
- усі атрибути містили лише атомарні (подільні) значення;
- кожен рядок у таблиці був унікальним.

У запропонованій структурі:

- кожен запис у таблицях має унікальний ідентифікатор (первинний ключ);
- усі значення в атрибутах є неподільними (наприклад, повне ім’я розбите на окремі поля: прізвище, ім’я, по батькові);
- відсутні повторювані групи даних.

Таким чином, база даних відповідає вимогам 1НФ.

Друга нормальна форма

Друга нормальна форма передбачає:

- виконання вимог 1НФ;
- усунення часткової залежності атрибутів від складеного первинного ключа.

У базі даних усі таблиці або мають простий первинний ключ, або атрибути залежать лише від повного складеного ключа (якщо він є). Наприклад:

- у таблиці ***IssuedBooks*** всі атрибути залежать або від *issue_id*, або від обох *copy_id* та *user_id* через зв’язок.
- інформацію про класи винесено в окрему таблицю ***Classes***, а не зберігається безпосередньо в таблиці ***Users***.

Таким чином, часткових залежностей немає — 2НФ виконано.

Третя нормальна форма

Третя нормальна форма вимагає:

- виконання вимог 2НФ;
- усунення транзитивних залежностей, тобто жоден неключовий атрибут не повинен залежати від іншого неключового атрибута.

В даній базі даних:

- наприклад, автор книги (автор_ім’я) винесений в окрему таблицю ***Authors***, щоб уникнути дублювання і залежності від назви книги;
- категорії літератури винесено в таблицю ***Categories***;
- місце зберігання книги та її статус містяться у таблиці ***BookCopies***, а не безпосередньо у ***Books***, що дозволяє обліковувати кілька примірників однієї книги незалежно;
- класи учнів зберігаються в таблиці ***Classes***, а в таблиці ***Users*** зберігається тільки зовнішній ключ *class_id*.

У результаті база даних приведена до 3НФ, що дозволяє забезпечити її логічну цілісність, уникнути надмірності даних та аномалій при оновленні інформації.

2.5 Опис логічної та фізичної моделі

Проєктування бази даних передбачає створення двох основних моделей: логічної та фізичної. Логічна модель відображає структуру даних, зв’язки між об’єктами та їх властивості без урахування особливостей конкретної системи керування базами даних . Фізична модель є деталізованим втіленням логічної моделі у вигляді конкретної реалізації в обраній СУБД з урахуванням технічних обмежень та особливостей.

2.5.1 Логічна модель бази даних

Логічна модель бази даних для системи управління шкільною бібліотекою включає основні сутності (таблиці) та зв’язки між ними. Вона

відображає структуру даних без конкретного прив’язування до фізичного зберігання, забезпечуючи цілісність та несуперечливість інформації.

У моделі визначено такі основні сутності:

- ***Users*** — містить інформацію про користувачів бібліотеки (учнів, вчителів, бібліотекарів). Основні атрибути: *UserID* (первинний ключ), *FullName*, *UserType*, *ClassID* (зовнішній ключ на таблицю *Classes* для учнів), PhoneNumber.
- ***Books*** — зберігає дані про книги. Основні атрибути: *BookID* (первинний ключ), *Title*, *AuthorID* (зовнішній ключ на таблицю *Authors*), *CategoryID* (зовнішній ключ на таблицю *Categories*), *YearPublished*, *ISBN*.
- ***Authors*** — містить відомості про авторів книг. Основні атрибути: *AuthorID* (первинний ключ), *FullName*, *BirthYear*.
- ***Categories*** — описує категорії літератури. Основні атрибути: *CategoryID* (первинний ключ), *CategoryName*.
- ***BookCopies*** — описує фізичні екземпляри книг. Основні атрибути: *CopyID* (первинний ключ), *BookID* (зовнішній ключ на таблицю *Books*), *InventoryNumber*, *Condition*.
- ***IssuedBooks*** — реєструє факти видачі книг користувачам. Основні атрибути: *IssueID* (первинний ключ), *CopyID* (зовнішній ключ на таблицю *BookCopies*), *UserID* (зовнішній ключ на таблицю *Users*), *IssueDate*, *ReturnDate*.
- ***Classes*** — містить інформацію про шкільні класи. Основні атрибути: *ClassID* (первинний ключ), *ClassName*, *ClassTeacher*.

Основні зв’язки між таблицями:

- ***Books*** має зв’язок із ***Authors*** (один автор може написати багато книг) — зв’язок *один-до-багатьох*.
- ***Books*** має зв’язок із ***Categories*** (одна категорія може містити багато книг) — зв’язок *один-до-багатьох*.
- ***BookCopies*** пов’язується з ***Books*** (одна книга може мати кілька екземплярів) — зв’язок *один-до-багатьох*.

– ***IssuedBooks*** пов’язана з ***BookCopies*** (одна видача — один примірник) — зв’язок *багато-до-одного*, та з ***Users*** (один користувач може отримати багато книг) — зв’язок *багато-до-одного*.

– ***Users*** пов’язана з ***Classes*** (учень може належати до одного класу, вчителі та бібліотекарі — ні) — зв’язок *багато-до-одного*.

Логічна модель забезпечує коректну організацію даних, дозволяє уникнути дублювання інформації, підтримує цілісність та уніфікацію даних. Завдяки продуманій структурі таблиць і зв’язків досягається ефективне управління бібліотечним фондом, користувачами та обліком видач книг.

Розроблена логічна модель є основою для створення фізичної моделі бази даних, реалізації системи у середовищі СУБД та подальшого написання *SQL*-запитів для роботи з даними.

2.5.2 Фізична модель бази даних

Фізична модель реалізована у вигляді *SQL*-структури на базі *Microsoft SQL Server*. У фізичній моделі вказані назви таблиць, стовпців, типи даних, первинні та зовнішні ключі, обмеження цілісності.

Основні особливості фізичної моделі:

– Використано типи даних *INT*, *VARCHAR*, *DATE*, *DECIMAL*, *BIT* відповідно до характеристик атрибутів.

– Для кожної таблиці визначено *PRIMARY KEY* для унікальної ідентифікації записів.

– Встановлено *FOREIGN KEY* для забезпечення зв’язків між таблицями.

– Передбачено використання *NOT NULL* для обов’язкових полів.

– Введено обмеження на унікальність там, де це необхідно (наприклад, унікальні ідентифікатори).

Приклад фізичної реалізації зв’язків:

– У таблиці *Books* поле *author_id* є зовнішнім ключем, що посилається на *author_id* з таблиці *Authors*.

- У таблиці *IssuedBooks* поле *user_id* посилається на *user_id* у таблиці ***Users*, а *copy_id*— на *copy_id*` у *BookCopies*.
- Таблиця *Users* містить поле *class_id*, що є зовнішнім ключем до таблиці *Classes*.

Таке проєктування забезпечує цілісність даних та дозволяє уникнути помилок при взаємодії між таблицями.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ БАЗИ ДАНИХ У СУБД

3.1 Вибір СУБД

Вибір системи управління базами даних (СУБД) є ключовим етапом у процесі розробки бази даних для шкільної бібліотеки. Це рішення має величезний вплив не лише на ефективність роботи з даними, але й на масштабованість, безпеку, зручність адміністрування та інтеграцію з іншими системами. У цьому контексті було обрано *Microsoft SQL Server* як оптимальну СУБД, що відповідає специфічним вимогам і умовам діяльності шкільної бібліотеки.

3.1.1 Ключові критерії для вибору СУБД

При виборі СУБД для створення бази даних шкільної бібліотеки було враховано кілька важливих факторів, що можуть істотно вплинути на ефективність впровадження:

Продуктивність і масштабованість: *Microsoft SQL Server* забезпечує високу продуктивність навіть при обробці великих обсягів даних. Це особливо важливо для шкільної бібліотеки, оскільки кількість користувачів та кількість книг можуть зрости, вимагаючи від системи адаптації до нових реалій без втрати продуктивності. Завдяки механізмам горизонтальної та вертикальної масштабованості, бібліотека може безперешкодно адаптуватися до змін у технологічному середовищі та зростаючих вимог користувачів. Це забезпечує не лише стабільну роботу системи, але й дозволяє в майбутньому впроваджувати нові функції та можливості без значних витрат і зусиль.

Безпека: Безпека є одним із найважливіших аспектів при виборі СУБД, особливо коли йдеться про зберігання конфіденційної інформації, такої як персональні дані користувачів бібліотеки. *Microsoft SQL Server* пропонує високий рівень захисту через шифрування даних, управління доступом на

рівні користувача та груп, а також можливість інтеграції з іншими системами безпеки. Додаткові можливості моніторингу доступу та ведення журналів дій користувачів підвищують загальний рівень безпеки системи, що є важливим для забезпечення довіри користувачів.

Підтримка транзакцій: Критично важливим аспектом для роботи бібліотеки є надійність процесів, пов'язаних із видачею, поверненням та резервуванням книг. *Microsoft SQL Server* надає потужні механізми підтримки транзакцій, що дозволяє забезпечити консистентність даних під час їх змін. Завдяки реалізації *ACID*-принципів, усі транзакції виконуються в повному обсязі або ж не виконуються зовсім, що виключає ймовірність виникнення ситуацій, коли дані можуть бути в непослідовному стані внаслідок одночасних операцій **Безпека.**

Однією з основних переваг *Microsoft SQL Server* є високий рівень безпеки, яка реалізується через шифрування даних, управління доступом на рівні користувача та груп, а також інтеграцію з іншими системами безпеки. Це дає змогу захистити конфіденційну інформацію, таку як особисті дані користувачів бібліотеки, від несанкціонованого доступу. Надані можливості для моніторингу доступу та ведення журналів дій користувачів ще більше підвищують рівень безпеки.

Підтримка складних запитів та процедур: *Microsoft SQL Server* підтримує розширені можливості для виконання *SQL*-запитів, процедур та тригерів, що дозволяє ефективно обробляти дані та автоматизувати різноманітні процеси. Це, зокрема, стосується нарахування штрафів за прострочені книги або перевірки наявності зарезервованих видань. Завдяки потужному механізму оптимізації запитів, система може справлятися навіть із найскладнішими запитами без істотних затримок, що значно підвищує зручність користування системою.

Інтеграція з іншими системами *Microsoft SQL Server* дозволяє безперешкодно інтегруватися з іншими частинами шкільного інформаційного середовища, такими як електронні журнали, системи управління навчальним

процесом та інші програмні рішення, що використовуються в школі. Це забезпечує єдність інформаційного потоку, що, в свою чергу, спрощує доступ до даних для всіх учасників навчального процесу, дозволяючи вчителям, учням і адміністраторам швидко отримувати необхідну інформацію.

Доступність інструментів для адміністрування *Microsoft SQL Server* надає широкий набір потужних інструментів для адміністрування, зокрема *SQL Server Management Studio*, які суттєво спрощують налаштування, моніторинг і управління базою даних. Завдяки цим інструментам, адміністраторам стає легше реагувати на запити користувачів та забезпечувати безперервну роботу системи. Наявність автоматизованих систем моніторингу дозволяє оперативно виявляти та усувати проблеми, що виникають у процесі експлуатації, що знижує ризики та підвищує загальну стабільність роботи бібліотечної системи.

3.1.2 Огляд можливих СУБД

Для створення бази даних для шкільної бібліотеки розглядалися кілька популярних систем управління базами даних (СУБД). Кожна з них має свої унікальні особливості, які можуть бути важливими в контексті специфічних потреб шкільної бібліотеки.

MySQL. *MySQL* є однією з найбільш популярних СУБД з відкритим кодом, яка реалізує *SQL*. Її основною перевагою є простота використання та висока продуктивність, що робить її ідеальною для невеликих та середніх проєктів. Однак, незважаючи на свою популярність, *MySQL* має певні обмеження в контексті роботи зі складними транзакціями, тригерами та індексацією. Ці обмеження можуть стати перепоною для бібліотеки, що планує розширювати свої функції або обсяг інформації. Також важливо враховувати, що для забезпечення надійного резервного копіювання та безпеки даних часто необхідно використовувати сторонні рішення, що може ускладнити загальну архітектуру системи.

PostgreSQL. *PostgreSQL* — це потужна об'єктно-реляційна СУБД з відкритим кодом, що підтримує складні запити, транзакції, а також різноманітні розширення. Однією з основних переваг *PostgreSQL* є її здатність обробляти великі обсяги даних та виконувати складні аналітичні запити, що може бути корисним для бібліотек з розширеною структурою даних. Проте варто зазначити, що налаштування та адміністрування *PostgreSQL* є більш складними в порівнянні з іншими СУБД. Це може виявитися проблемою в умовах шкільної бібліотеки, де часто немає фахівців з високими кваліфікаціями для управління такими системами. Тому, незважаючи на свою гнучкість та потужність, *PostgreSQL* може не бути оптимальним вибором для даного контексту.

SQLite. *SQLite* представляє собою легку та компактну СУБД, що зберігає всі дані в одному файлі, що робить її простим у використанні рішенням. Відсутність необхідності в окремому сервері та простота налаштування є значними перевагами. Проте, варто зазначити, що *SQLite* має значні обмеження в багатокористувацькому середовищі. Вона не призначена для одночасного доступу великої кількості користувачів, що може призвести до проблем із продуктивністю та цілісністю даних. Ці недоліки можуть стати критичними в контексті шкільної бібліотеки, де система може використовуватися кількома користувачами одночасно.

Microsoft SQL Server. На відміну від інших розглянутих СУБД, Microsoft SQL Server пропонує широкий набір інструментів для розробки, адміністрування, резервного копіювання, моніторингу та безпеки «з коробки». Перевагами цього рішення є підтримка складних транзакцій, розширених запитів та інтеграція з іншими системами. Додатково, безкоштовна версія Microsoft SQL Server Express забезпечує доступ до потужних функцій без додаткових фінансових витрат, що робить цю СУБД оптимальним вибором для шкільної бібліотеки. Інтуїтивно зрозумілий графічний інтерфейс *SQL Server Management Studio* спрощує процес налаштування, моніторингу та

адміністрування системи, що є важливим аспектом для бібліотек, які можуть не мати висококваліфікованого *IT*-персоналу.

Таблиця 3.1 – Порівняння СУБД

Критерій	***MySQL***	***PostgreSQL***	***Microsoft SQL Server***	***SQLite***
Тип ліцензії	*Open Source*	*Open Source*	Пропрієтарна (безкоштовна *Express*)	*Open Source*
Платформа	*Windows, Linux, MacOS*	*Windows, Linux, MacOS*	Переважно *Windows (Linux також*)	*Windows, Linux, MacOS*
Інструменти адміністрування	*MySQL Workbench, CLI*	*pgAdmin, CLI*	*SQL Server Management Studio (SSMS)*	*CLI,* сторонні *GUI*
Підтримка транзакцій	Так	Так	Так	Так
Підтримка збережених процедур і тригерів	Так	Так	Так	Обмежено
Безпека	Висока	Висока	Дуже висока (*Active Directory* інтеграція, ролі, політики)	Середня
Рівень продуктивності	Високий для невеликих систем	Високий для складних систем	Високий (оптимізований для великих та середніх систем)	Високий для невеликих проєктів
Можливість масштабування	Обмежена	Висока	Висока	Обмежена
Додаткові можливості	Реплікація	Розширені аналітичні функції, *JSON*	Інтеграція з *.NET, Reporting Services,* аналітичні сервіси	Немає серверної частини

3.1.3 Обґрунтування вибору *Microsoft SQL Server*

Порівняння розглянутих СУБД дозволяє зробити висновок, що для умов шкільної бібліотеки ***Microsoft SQL Server*** є найбільш збалансованим рішенням.

Основні переваги такого вибору:

1) **Масштабованість і надійність**: *Microsoft SQL Server* підтримує масштабованість, що дозволяє йому справлятися з поточними вимогами, а також адаптуватися до зростаючих обсягів даних у майбутньому. Це важливо для забезпечення безперервності роботи бібліотеки, навіть під час збільшення кількості користувачів та фондів. Завдяки цьому, бібліотека зможе без труднощів реагувати на зміни у запитах та потребах своїх користувачів. На відміну від *MySQL* та *SQLite*, що обмежені у функціональності при зростанні обсягів даних, SQL Server легко адаптується до змін.

2) **Підтримка транзакцій та цілісності даних**: Система забезпечує гарантію того, що всі операції з базою даних, такі як видача книг, їх повернення або накладення штрафів, будуть виконані без порушень цілісності даних, чого бракує у *SQLite*, а *MySQL* має певні обмеження без використання спеціалізованих механізмів. Це особливо важливо для підтримки довіри користувачів до системи. Завдяки цьому, можна бути впевненими у тому, що дані залишатимуться актуальними і точними у будь-який момент.

3) **Інструменти для адміністрування**: *Microsoft SQL Server* має зручні та потужні інструменти для адміністрування бази даних, що значно спрощує її налаштування, управління та моніторинг, що є значною перевагою перед *PostgreSQL*, де адміністрування може бути складнішим. Адміністратори можуть швидко реагувати на запити та проблеми, що виникають, підвищуючи ефективність роботи бібліотеки. Наявність функцій автоматизації, таких як планування завдань та резервного копіювання, дозволяє зменшити трудозатрати на обслуговування системи.

4) **Висока безпека**: Завдяки розвинутим засобам безпеки *Microsoft SQL Server* забезпечує захист конфіденційних даних, таких як персональна

інформація користувачів бібліотеки. На відміну від більшості безкоштовних рішень це дозволяє не лише виконувати вимоги законодавства щодо захисту даних, а й підвищує загальний рівень довіри до бібліотеки як до організації. Впровадження сучасних технологій шифрування та аутентифікації дозволяє значно знизити ризики витоку інформації.

Хоча інші СУБД, такі як *MySQL* або *PostgreSQL*, є також дуже популярними для малих і середніх проектів, Microsoft SQL Server пропонує велику кількість інтегрованих інструментів, зокрема для автоматизації бекапів та моніторингу продуктивності. Це особливо важливо для середніх та великих шкіл, де база даних може обслуговувати багато користувачів одночасно. В результаті, вибір *Microsoft SQL Server* як основної СУБД для шкільної бібліотеки є обґрунтованим кроком, який забезпечить надійну та ефективну роботу системи в довгостроковій перспективі. Впровадження даної системи дозволить оптимізувати роботу бібліотеки, покращити обслуговування користувачів та забезпечити більш високий рівень управління бібліотечними ресурсами.

3.2 Створення таблиць

До складу бази даних входять такі основні таблиці:

– ***Books***. Таблиця ***Books*** містить основні дані про всі книги, що є в бібліотеці. Вона включає інформацію про назву книги, автора, жанр (категорію), рік видання та видавництво. Завдяки цій таблиці бібліотекар може легко знаходити потрібну книгу за різними параметрами (наприклад, за назвою, автором чи жанром) і вести системний облік книжкового фонду.

– ***Authors***. Таблиця ***Authors*** містить інформацію про авторів книг, що є у фонді бібліотеки. Вона дозволяє систематизувати книги за авторством і швидко здійснювати пошук літератури певного письменника.

– ***Categories.*** Таблиця *Categories* визначає жанри чи тематичні напрями літератури в бібліотеці. Це дозволяє структурувати книги за категоріями (наприклад, художня література, довідкова, навчальна тощо).

– ***Book Copies.*** Таблиця *BookCopies* містить інформацію про всі фізичні примірники книг. Одна книга може мати кілька примірників, кожен з яких має власний інвентарний номер і статус.

– ***Users.*** Таблиця ***Users*** зберігає дані про всіх користувачів бібліотеки: учнів, учителів, бібліотекаря. Це дозволяє персоналізовано обслуговувати читачів, зберігати історію їхніх запозичень та керувати доступом до ресурсів бібліотеки.

– ***Classes.*** Таблиця ***Classes*** використовується для обліку шкільних класів, у яких навчаються учні — користувачі бібліотеки. Це дозволяє групувати читачів за класами для зручності обліку та аналітики.

– ***Issued Books***. Таблиця ***IssuedBooks*** призначена для фіксації фактів видачі примірників книг користувачам. Вона містить інформацію про дату видачі, планову дату повернення та фактичну дату, коли книгу було повернуто.

3.2.1 *SQL*-запити для створення таблиць

Нижче наведено *SQL*-запити для створення основних таблиць бази даних:

```
CREATE TABLE Authors (
   AuthorID INT PRIMARY KEY IDENTITY(1,1),
   FullName NVARCHAR(255) NOT NULL
);

CREATE TABLE Categories (
   CategoryID INT PRIMARY KEY IDENTITY(1,1),
   Name NVARCHAR(100) NOT NULL,
```

```
    Description NVARCHAR(255)
);

CREATE TABLE Books (
    BookID INT PRIMARY KEY IDENTITY(1,1),
    Title NVARCHAR(255) NOT NULL,
    AuthorID INT NOT NULL,
    CategoryID INT NOT NULL,
    Year INT,
    Publisher NVARCHAR(255),
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

CREATE TABLE BookCopies (
    CopyID INT PRIMARY KEY IDENTITY(1,1),
    BookID INT NOT NULL,
    InventoryNumber NVARCHAR(50) NOT NULL,
    Status NVARCHAR(50) NOT NULL,
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);

CREATE TABLE Classes (
    ClassID INT PRIMARY KEY IDENTITY(1,1),
    ClassName NVARCHAR(50) NOT NULL
);


CREATE TABLE Users (
    UserID INT PRIMARY KEY IDENTITY(1,1),
```

```
    FullName NVARCHAR(255) NOT NULL,
    BirthDate DATE,
    Phone NVARCHAR(20),
    Role NVARCHAR(50),
    ClassID INT,
    FOREIGN KEY (ClassID) REFERENCES Classes(ClassID)
);

CREATE TABLE IssuedBooks (
    IssueID INT PRIMARY KEY IDENTITY(1,1),
    CopyID INT NOT NULL,
    UserID INT NOT NULL,
    IssueDate DATE NOT NULL,
    ReturnDate DATE NOT NULL,
    ActualReturnDate DATE,
    FOREIGN KEY (CopyID) REFERENCES BookCopies(CopyID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

3.2.2 Опис типів даних

Для забезпечення коректного зберігання інформації використано такі типи даних:

- *INT* — ціле число, призначене для зберігання ідентифікаторів записів, кількості примірників книг, року видання.
- *VARCHAR* — рядок змінної довжини, використовується для зберігання текстової інформації: назв книг, імен користувачів, жанрів, контактних даних.
- *DECIMAL* — числовий тип із фіксованою кількістю знаків після коми, застосовується для відображення сум штрафів.

- *DATE* — тип для зберігання календарних дат (дата видачі, повернення, накладення штрафу).
- *BIT* — логічний тип даних, що приймає значення 0 або 1, використовується для позначення стану оплати штрафу.

3.2.3 Визначення зв'язків між таблицями

Зв’язки між таблицями забезпечують логічну цілісність даних:

Основні зв’язки між таблицями:

- ***Books – Authors*:** Кожна книга має одного автора, але один автор може бути пов’язаний із кількома книгами. Тип зв’язку: **1:M** (*Authors → Books*).
- ***Books – Categories*:** Кожна книга належить до однієї категорії, але одна категорія може містити багато книг. Тип зв’язку: **1:M** (*Categories → Books*).
- ***Books – BookCopies*:**Одна книга може мати декілька примірників у бібліотеці. Тип зв’язку: **1:M** (*Books → BookCopies*).
- ***Classes – Users*:** Учень або користувач належить до одного класу, але клас може містити багато користувачів. Тип зв’язку: **1:M** (*Classes → Users*).
- ***BookCopies – IssuedBooks*:** Кожен конкретний примірник книги може бути виданий багато разів у різний час. Тип зв’язку: **1:M** (*BookCopies → IssuedBooks*).
- ***Users – IssuedBooks*:** Один користувач може брати багато книг у різний час, але кожен запис про видачу належить лише одному користувачу. Тип зв’язку: **1:M** (*Users → IssuedBooks*).

Таким чином, створена структура таблиць та встановлені зв’язки між ними забезпечують цілісність даних і надають можливість виконання комплексних запитів для організації бібліотечної системи.

3.3 Реалізація операцій видачі та повернення книг

Операції видачі та повернення книг у бібліотеці становлять важливі, основоположні компоненти, що забезпечують ефективність та злагодженість її функціонування. Ці процеси не лише підтримують організацію обліку літератури, але й активно сприяють покращенню користувацького досвіду, дозволяючи читачам мати легкий доступ до необхідних джерел знань. Важливо зазначити, що для забезпечення безперебійної роботи цих операцій необхідно створити потужні механізми обробки даних у базі даних, які передбачають не лише внесення інформації про видачу книг, але й обробку їх повернення, включаючи нарахування штрафів за прострочення терміну.

Для реалізації процесів видачі та повернення книг у базі даних використовуються операції вставки та оновлення даних у таблиці *IssuedBooks*.

Видача книги:

```
BEGIN TRANSACTION;

IF EXISTS (SELECT 1 FROM BookCopies WHERE BookID = 16 AND Status = 'Доступна')
BEGIN
    DECLARE @CopyID INT;
    SELECT TOP 1 @CopyID = CopyID
    FROM BookCopies
    WHERE BookID = 16 AND Status = 'Доступна';

    UPDATE BookCopies
    SET Status = 'На руках'
    WHERE CopyID = @CopyID;
```

```
    INSERT INTO IssuedBooks (CopyID, UserID, IssueDate, ReturnDate,
ActualReturnDate)
    VALUES (@CopyID, 1, GETDATE(), DATEADD(DAY, 14, GETDATE()),
NULL);

    COMMIT;
    PRINT 'Книга (BookID = 16) видана успішно.';
END
ELSE
BEGIN
    ROLLBACK;
    PRINT 'Книга (BookID = 16) недоступна для видачі.';
END
```

Пояснення:

- *BEGIN TRANSACTION* – починає транзакцію, яка об’єднує кілька *SQL*-операцій в одну логічну групу. Це дозволяє або виконати всі дії разом, або скасувати їх у разі помилки.
- *IF EXISTS (SELECT 1 FROM BookCopies WHERE BookID = 16 AND Status* = 'Доступна') – перевіряє, чи існує в таблиці *BookCopies* хоча б один запис, де:

BookID = 16 – ідентифікатор книги.

Status = 'Доступна' – книга зараз вільна для видачі.

Якщо такий запис є – виконується блок *BEGIN ... END*. Якщо ні – виконується блок *ELSE*.

- *DECLARE @CopyID INT* – оголошення змінної для збереження ідентифікатора доступної копії книги.
- *SELECT TOP 1 @CopyID = CopyID FROM BookCopies WHERE BookID = 16 AND Status* = 'Доступна' – вибирає перший доступний екземпляр книги з *BookID* = 16 і записує його ідентифікатор у змінну *@CopyID*.

– *UPDATE BookCopies SET Status* = 'На руках' *WHERE CopyID = @CopyID* – оновлює статус обраного екземпляра книги на 'На руках', що означає – книгу видано.

– *INSERT INTO IssuedBooks (CopyID, UserID, IssueDate, ReturnDate, ActualReturnDate)* – додає новий запис про видачу книги до таблиці *IssuedBooks*.

– *VALUES (@CopyID, 1, GETDATE(), DATEADD(DAY, 14, GETDATE()), **NULL)*** – визначає значення для полів:

CopyID = @CopyID – ідентифікатор екземпляра книги, яку видано.

UserID = 1 – ідентифікатор користувача, якому видано книгу.

IssueDate = GETDATE() – поточна дата видачі.

ReturnDate = DATEADD(DAY, 14, GETDAT*E())* – дата, коли книгу потрібно повернути (через 14 днів).

ActualReturnDate = NULL – фактична дата повернення поки невідома.

– *COMMIT* – підтверджує виконання всіх дій у транзакції. Дані остаточно записуються в базу.

– *PRINT* 'Книга (*BookID* = 16) видана успішно.' – виводить повідомлення про успішну видачу.

– *ELSE BEGIN ROLLBACK; PRINT* 'Книга (*BookID* = 16) недоступна для видачі.'; *END* – якщо доступної копії немає:

– *ROLLBACK* скасовує всі дії транзакції.

– *PRINT* виводить повідомлення про те, що книгу видати неможливо.

Результат: у таблиці *IssuedBooks* з’явиться новий запис про видачу книги.

Повернення книги:

```
BEGIN TRANSACTION;

DECLARE @CopyID INT;
```

```
SELECT @CopyID = CopyID FROM IssuedBooks WHERE IssueID = 7;

UPDATE BookCopies
SET Status = 'Доступна'
WHERE CopyID = @CopyID;

UPDATE IssuedBooks
SET ActualReturnDate = GETDATE()
WHERE IssueID = 7;

COMMIT;
PRINT 'Книга повернута успішно.';
```

Пояснення:

– *BEGIN TRANSACTION* – починає транзакцію, яка об'єднує кілька *SQL*-операцій в одну групу, щоб або виконати всі дії разом, або скасувати їх у разі помилки.

– *DECLARE @CopyID INT* – оголошення змінної для збереження ідентифікатора примірника книги, який потрібно повернути.

– *SELECT @CopyID = CopyID FROM IssuedBooks WHERE IssueID = 7* – вибирає значення *CopyID* з таблиці *IssuedBooks* для запису, де:

IssueID = 7 – ідентифікатор конкретного факту видачі книги.

– *UPDATE BookCopies SET Status = 'Доступна' WHERE CopyID = @CopyID* – оновлює статус примірника книги на 'Доступна', що означає – книга повернута до бібліотеки й доступна для наступної видачі.

– *UPDATE IssuedBooks SET ActualReturnDate = GETDATE() WHERE IssueID = 7* – встановлює дату фактичного повернення книги в таблиці *IssuedBooks*:

ActualReturnDate = GETDATE() – поточна дата та час повернення.

WHERE IssueID = 7 – для конкретного запису видачі книги.

– *COMMIT* – підтверджує виконання всіх дій транзакції. Дані остаточно записуються в базу.

– *PRINT* 'Книга повернута успішно.' – виводить повідомлення про успішне повернення книги.

Результат: Примірник книги зі статусом **'На руках'** повертається в бібліотеку (статус змінюється на **'Доступна'**), а в таблиці ***IssuedBooks*** фіксується дата фактичного повернення.

Після видачі книги статус її екземпляра має змінюватися на «Видано», а після повернення – на «Доступна».

Приклад зміни статусу на «На руках»:

```
UPDATE BookCopies
SET Status = N'На руках'
WHERE CopyID = 12;
```

Аналогічно, після повернення книги:

```
UPDATE BookCopies
SET Status = N'Доступна'
WHERE CopyID = 12;
```

Пояснення:

– ***UPDATE BookCopies*** – вказуємо таблицю з екземплярами книг.

– ***SET Status = ...*** – задаємо нове значення статусу.

– ***WHERE CopyID = 12*** – уточнюємо, який саме екземпляр книги потрібно оновити.

Результат: зміна статусу книги на «На руках» або «Доступно».

Перевірка наявності вільного екземпляра книги

Щоб перед видачею переконатися у наявності доступного екземпляра книги, виконується запит:

```
SELECT TOP 1 CopyID
FROM BookCopies
WHERE BookID = 5 AND Status = N'Доступна';
```

Пояснення:

- ***SELECT TOP 1 CopyID*** – обираємо перший доступний екземпляр.
- ***FROM BookCopies*** – з таблиці екземплярів книг.
- ***WHERE BookID = 5 AND Status = N'*****Доступна'** – умова вибору екземпляра конкретної книги за її ***BookID***, який має статус **«Доступна»**.

Результат: ідентифікатор доступного екземпляра книги, готового до видачі.

Видалення запису про видачу (у разі помилки)

Якщо запис про видачу було внесено помилково, його можна видалити:

```
DELETE FROM IssuedBooks
WHERE IssueID = 10;
```

Пояснення:

- ***DELETE FROM IssuedBooks*** – видаляємо дані з таблиці видачі книг.
- ***WHERE IssueID = 10*** – умова видалення за унікальним ідентифікатором запису.

Результат: запис про видачу книги буде видалено з бази.

Реалізовані *SQL*-запити для операцій видачі та повернення книг дозволяють ефективно управляти процесами обліку, контролювати статус екземплярів книг, фіксувати факти видачі та повернення, а також оперативно коригувати помилкові записи. Це забезпечує стабільну та зручну роботу бібліотечної системи.

3.4 Приклади запитів (облік, пошук та аналітика)

Для забезпечення повноцінного обліку бібліотечних ресурсів, користувачів, контролю операцій видачі та повернення книг, а також для отримання аналітичних звітів у системі реалізовано низку *SQL*-запитів. Ці запити дозволяють отримувати необхідну інформацію в зручному вигляді, здійснювати пошук за певними параметрами, додавати, редагувати та видаляти дані.

У цьому підрозділі розглянемо основні приклади запитів із поясненням їхнього призначення та принципів роботи.

Пошук книг за автором:

```
SELECT Books.Title, Authors.FullName
FROM Books
INNER JOIN Authors ON Books.AuthorID = Authors.AuthorID
WHERE Authors.FullName = 'Тарас Шевченко';
```

Пояснення:

- *SELECT Books.Title, Authors.FullName* – виводимо назву книги та повне ім’я автора.
- *FROM Books* – джерелом даних є таблиця *Books*.
- *INNER JOIN Authors ON Books.AuthorID = Authors.AuthorID* – об’єднуємо таблиці *Books* та *Authors* за полем *AuthorID*.
- *WHERE Authors.FullName* = 'Тарас Шевченко' – відображаємо лише ті записи, де автор має вказане ім’я.

Результат: список книг, написаних Тарасом Шевченком.

Облік виданих книг:

```
SELECT Users.FullName, Books.Title, IssuedBooks.IssueDate,
IssuedBooks.ReturnDate
FROM IssuedBooks
INNER JOIN BookCopies ON IssuedBooks.CopyID = BookCopies.CopyID
INNER JOIN Books ON BookCopies.BookID = Books.BookID
```

```
INNER JOIN Users ON IssuedBooks.UserID = Users.UserID;
```

Пояснення:

- ***SELECT Users.FullName, Books.Title, IssuedBooks.IssueDate, IssuedBooks.ReturnDate*** – виводимо ім'я користувача, назву книги, дату видачі та дату повернення.
- ***FROM IssuedBooks*** – основна таблиця видачі книг.
- ***INNER JOIN BookCopies*** – об'єднуємо з таблицею екземплярів книг за ***CopyID***.
- ***INNER JOIN Books*** – далі з таблицею книг за ***BookID***.
- ***INNER JOIN Users*** – об'єднуємо з користувачами за ***UserID***.

Результат: таблиця з переліком виданих книг і даними про користувачів.

Аналіз кількості виданих книг за місяць:

```
SELECT MONTH(IssueDate) AS Month, COUNT(*) AS IssuedCount
FROM IssuedBooks
GROUP BY MONTH(IssueDate)
ORDER BY Month;
```

Пояснення:

- ***SELECT MONTH(IssueDate) AS Month*** – визначаємо номер місяця за датою видачі.
- ***COUNT(*) AS IssuedCount*** – рахуємо кількість виданих книг.
- ***FROM IssuedBooks*** – дані беремо з таблиці видачі книг.
- ***GROUP BY MONTH(IssueDate)*** – групуємо дані за місяцем.
- ***ORDER BY Month*** – сортуємо результат за зростанням місяця.

Результат: статистика кількості виданих книг по місяцях**.**

Облік користувачів: Для управління інформацією про користувачів реалізовано такі запити:

Додавання нового користувача:

```
INSERT INTO Users (FullName, BirthDate, Phone, Role, ClassID)
VALUES (N'Іван Петренко', '2009-05-20', '0971234567', N'Учень', 3);
```

Пояснення:

- ***SELECT UserID, FullName, BirthDate, Phone, Role*** – вибираємо унікальний ідентифікатор, ім’я користувача, дату народження, телефон та роль.
- ***FROM Users*** – дані беруться з таблиці користувачів.

Результат: список усіх користувачів бібліотеки з основною інформацією.

Видалення користувача:

```
DELETE FROM Users
WHERE UserID = 5;
```

Пояснення:

- ***DELETE FROM Users*** – вказуємо, з якої таблиці видаляти.
- ***WHERE UserID = 5*** – умова видалення: за унікальним ідентифікатором.

Результат: з таблиці ***Users*** буде видалено користувача з ***UserID = 5***.

Реалізація таких запитів дозволяє не лише зберігати дані, а й активно ними оперувати: здійснювати пошук, перегляд інформації, контролювати видачу та повернення книг, формувати аналітичні звіти та керувати даними користувачів. Це забезпечує оперативність роботи бібліотеки та підвищує зручність користування системою.

РОЗДІЛ 4
ТЕСТУВАННЯ І ОПТИМІЗАЦІЯ

4.1 Перевірка працездатності бази даних

Перевірка працездатності бази даних є важливим етапом розробки, що дозволяє впевнитись у тому, що система працює правильно і відповідає всім вимогам. Вона включає перевірку правильності виконання *SQL*-запитів, коректність збереження даних, обробку помилок та загальну ефективність бази даних. У цьому підрозділі розглянемо основні етапи перевірки працездатності бази даних для шкільної бібліотеки, які включають тести на правильність функціонування операцій, а також використання інструментів для моніторингу і оптимізації бази даних.

4.1.1 Тестування функціональності основних операцій

Перевірка працездатності бази даних починається з тестування основних операцій, таких як додавання, оновлення та видалення записів, а також виконання запитів на вибірку даних.

1. **Тестування додавання нових користувачів**. Тестуємо додавання нового користувача до бази даних за допомогою *SQL*-запиту:

```
INSERT INTO Users (FullName, BirthDate, Phone, Role, ClassID)
VALUES ('Марія Іваненко', '2010-05-21', '+380501234567', 'Учень', 1),
```

Перевірка результату:

- Після виконання запиту перевіряємо, чи додано нового користувача в таблицю ***Users***.
- Виконуємо запит для перевірки наявності нового запису:

```
SELECT * FROM Users WHERE FullName = 'Марія Іваненко';
```

2. **Тестування оновлення даних користувача.** Тестуємо оновлення інформації про користувача, наприклад, змінюючи його номер телефону:

```
UPDATE Users
SET Phone = '+380601234567 '
WHERE FullName = 'Марія Іваненко';
```

Перевірка результату:

– Після виконання запиту перевіряємо, чи змінився номер телефону у записі користувача.

3. **Тестування видалення користувача.** Тестуємо видалення користувача з бази даних:

```
DELETE FROM Users WHERE FullName = 'Марія Іваненко';
```

Перевірка результату:

– Виконуємо запит для перевірки, чи дійсно запис про користувача було видалено:

```
SELECT * FROM Users WHERE FullName = 'Марія Іваненко';
```

4.1.2 Тестування операцій видачі та повернення книг

Важливо перевірити, чи правильно працюють операції видачі та повернення книг, адже вони є основними функціями в управлінні бібліотекою.

1. Тестування видачі книги

Перевірка коректності видачі книги може бути здійснена за допомогою *SQL*-запиту на видачу:

```
BEGIN TRANSACTION;

IF EXISTS (SELECT 1 FROM BookCopies WHERE BookID = 16 AND Status =
'Доступна')
BEGIN
    DECLARE @CopyID INT;
```

```
    SELECT TOP 1 @CopyID = CopyID
    FROM BookCopies
    WHERE BookID = 16 AND Status = 'Доступна';

    UPDATE BookCopies
    SET Status = 'На руках'
    WHERE CopyID = @CopyID;

    INSERT INTO IssuedBooks (CopyID, UserID, IssueDate, ReturnDate,
ActualReturnDate)
    VALUES (@CopyID, 1, GETDATE(), DATEADD(DAY, 14, GETDATE()),
NULL);

    COMMIT;
    PRINT 'Книга (BookID = 16) видана успішно.';
END
ELSE
BEGIN
    ROLLBACK;
    PRINT 'Книга (BookID = 16) недоступна для видачі.';
END
```

Перевірка результату:

– Перевіряємо, чи було зменшено кількість доступних примірників у таблиці ***Books*** та чи з'явився запис про видачу в таблиці ***IssuedBooks***.

2. **Тестування повернення книги**. Перевірка коректності повернення книги після терміну позики:

```
BEGIN TRANSACTION;

DECLARE @CopyID INT;
SELECT @CopyID = CopyID FROM IssuedBooks WHERE IssueID = 7;
```

```
UPDATE BookCopies
SET Status = 'Доступна'
WHERE CopyID = @CopyID;

UPDATE IssuedBooks
SET ActualReturnDate = GETDATE()
WHERE IssueID = 7;

COMMIT;
PRINT 'Книга повернута успішно.';
```

Перевірка результату:

– Перевіряємо, чи змінилася дата повернення у таблиці ***IssuedBooks*** і чи змінився статус примірника у таблиці ***Books***.

4.1.3 Тестування запитів для обліку користувачів і книг

Перевірка працездатності запитів для обліку користувачів і книг є важливою для гарантування коректної роботи системи:

1. **Тестування запиту на отримання списку всіх користувачів**

```
SELECT
  UserID, FullName, BirthDate, Phone, Role, ClassID
FROM Users;
```

Перевірка результату:

– Переконуємось, що запит повертає всі необхідні дані про користувачів і що вони коректно відображаються.

2. **Тестування запиту на пошук книги за назвою**

```
SELECT BookID, Title, AuthorID, CategoryID, Year, Publisher
FROM Books
WHERE Title LIKE '%Кобзар%';
```

Перевірка результату:

– Переконуємось, що запит знаходить правильні книги за вказаним критерієм і виводить потрібні дані.

4.1.4 Перевірка продуктивності бази даних

Окрім перевірки функціональності, важливо тестувати продуктивність бази даних, щоб забезпечити її ефективну роботу за великих обсягів даних. Для цього можна використовувати інструменти моніторингу, такі як ***SQL Server Profiler***, а також проводити тести на швидкість виконання запитів за допомогою команди ***EXPLAIN*** для аналізу виконання запитів.

1. **Тестування швидкості виконання запиту**. Для перевірки швидкості виконання запиту можна використовувати таку команду в *SQL Server*:

```
SET STATISTICS TIME ON;

SELECT BookID, Title, AuthorID, CategoryID, Year, Publisher
FROM Books

SET STATISTICS TIME OFF;
```

Цей запит дозволяє отримати інформацію про час виконання запиту, що дозволяє виявити можливі вузькі місця в базі даних.

4.2 Оптимізація структури та запитів

Оптимізація структури бази даних і SQL-запитів є важливою частиною забезпечення ефективної роботи системи управління шкільною бібліотекою. Вона дозволяє скоротити час виконання запитів, покращити загальну продуктивність і забезпечити ефективне використання ресурсів. У цьому

підрозділі розглянуто основні методи оптимізації структури бази даних та *SQL*-запитів для проєктованої системи.

4.2.1 Оптимізація структури бази даних

Нормалізація бази даних

Нормалізація – це процес організації структури бази даних із метою зменшення надмірності даних і забезпечення їх логічної цілісності. У проєктованій базі даних шкільної бібліотеки було реалізовано нормалізацію до третьої нормальної форми. Це дозволило винести повторювані дані в окремі таблиці та створити між ними зв’язки.

Приклад нормалізації:

- Таблиця *Books* містить інформацію про книги: назва, ідентифікатор автора, рік видання, жанр.
- Таблиця *Authors* зберігає дані про авторів.
- Таблиця *Users* містить дані про читачів.
- Таблиця *IssuedBooks* фіксує факти видачі книг, включаючи ідентифікатори книги та користувача, дати видачі й повернення.

Це дозволяє уникнути дублювання даних і забезпечити логічну структуру.

Індексація таблиць

Щоб пришвидшити вибірку даних, було створено індекси на стовпцях, які найчастіше використовуються для пошуку:

```
CREATE INDEX IDX_AuthorID ON Books(AuthorID);
```

Використання зовнішніх ключів для забезпечення цілісності

Зовнішні ключі забезпечують цілісність зв’язків між таблицями, наприклад:

```
ALTER TABLE Books
ADD CONSTRAINT FK_Books_Authors FOREIGN KEY (AuthorID)
REFERENCES Authors(AuthorID);
```

Це гарантує, що неможливо буде видати неіснуючу книгу чи видати її неіснуючому користувачу.

4.2.2 Оптимізація SQL-запитів

Оптимізація *SQL*-запитів допомагає зменшити час виконання і навантаження на систему.

Використання індексів у запитах

Наприклад, для часто виконуваного запиту пошуку книг за автором:

```
SELECT Title
FROM Books
WHERE AuthorID = 5;
```

Індекс на стовпець *AuthorID* значно скорочує час виконання такого запиту.

Оптимізація операцій *JOIN*

Замість підзапитів доцільно використовувати *JOIN*. Наприклад:

Поганий варіант із підзапитом:

```
SELECT Title
FROM Books
WHERE BookID IN (SELECT BookID FROM IssuedBooks WHERE UserID = 10);
```

Оптимізований варіант з *JOIN*:

```
SELECT B.Title
FROM Books B
INNER JOIN IssuedBooks I ON B.BookID = I.BookID
WHERE I.UserID = 10;
```

Фільтрація даних у *WHERE*

Щоб прискорити запити, умови фільтрації потрібно вказувати на початку виконання. Наприклад:

```
SELECT I.IssueID, U.LastName, B.Title, I.IssueDate
FROM IssuedBooks I
JOIN Users U ON I.UserID = U.UserID
JOIN Books B ON I.BookID = B.BookID
WHERE I.ReturnDate IS NULL AND I.IssueDate < GETDATE();
```

Так система одразу обмежує вибірку тільки актуальними записами.

Аналіз плану виконання

Для контролю продуктивності виконання запитів використовувався ***Execution Plan*** у *Microsoft SQL Server Management Studio*, який допоміг виявити «вузькі місця» і визначити, які операції займають найбільше часу.

Приклад запиту з планом виконання:

```
SET STATISTICS IO ON;
SELECT * FROM Books WHERE CategoryID = '1';
```

Аналіз плану показав, що запити на фільтрацію за жанром доцільно оптимізувати додаванням індексу на стовпець *CategoryID*.

4.3 Аналіз проблем і варіанти їх вирішення

Під час розробки і впровадження бази даних для управління бібліотекою можуть виникнути різноманітні проблеми, які можуть впливати на її ефективність, функціональність і цілісність даних. Виявлення та вирішення цих проблем є важливим етапом для забезпечення безперебійної роботи

системи. У цьому підрозділі будуть розглянуті основні можливі проблеми, з якими можна зіткнутися під час розробки та експлуатації бази даних для шкільної бібліотеки, а також запропоновані методи їх вирішення.

4.3.1 Проблеми з продуктивністю запитів

Однією з найпоширеніших проблем є зниження продуктивності запитів за великих обсягів даних або через неправильно побудовані запити. Це може бути викликано кількома факторами:

1. **Великі обсяги даних**: У випадку, коли в базі даних зберігається велика кількість книг, користувачів або записів про видачу, запити можуть значно уповільнюватися, особливо якщо не використовуються індекси.

Рішення:

– **Індексація**: Створення індексів на стовпцях, за якими часто здійснюються запити (наприклад, на ***BookID***, ***UserID***, ***LoanDate***) дозволить значно пришвидшити виконання запитів.

– **Пагінація**: Для запитів, які повертають велику кількість рядків, можна використовувати пагінацію, обмежуючи кількість записів, що повертаються в одному запиті.

Приклад запиту з пагінацією:

```
SELECT * FROM Books
ORDER BY Title
OFFSET 0 ROWS FETCH NEXT 20 ROWS ONLY;
```

Неправильне використання підзапитів: Підзапити можуть значно сповільнити виконання запитів, особливо коли їх застосовують до великих таблиць.

Рішення:

– Заміна підзапитів на ***JOIN*** може значно покращити продуктивність, оскільки ***JOIN*** дозволяє обробляти дані більш ефективно.

Приклад заміни підзапиту на ***JOIN***:

Поганий запит:

```
SELECT Title FROM Books WHERE BookID IN (SELECT CopyID FROM IssuedBooks WHERE UserID = 1);
```

Оптимізований запит:

```
SELECT B.Title
FROM Books B
JOIN IssuedBooks I ON B.BookID = I.CopyID
WHERE I.UserID = 1;
```

4.3.2 Проблеми з цілісністю даних

Цілісність даних є важливою для забезпечення коректності роботи бази даних. Проблеми з цілісністю можуть виникати, якщо зв’язки між таблицями не реалізовані належним чином або якщо з бази даних видаляються чи змінюються важливі записи, що можуть призвести до неконсистентних даних.

1. **Відсутність зв'язків між таблицями**: Якщо не створено зовнішні ключі або інші обмеження на зв'язки між таблицями, можуть виникнути проблеми з цілісністю даних, наприклад, при видаленні користувача, який все ще має неповернуті книги.

Рішення:

– Використання **зовнішніх ключів** для забезпечення правильних зв'язків між таблицями дозволяє автоматично підтримувати цілісність даних.

Приклад створення зовнішнього ключа:

```
ALTER TABLE IssuedBooks
ADD CONSTRAINT FK_UserID FOREIGN KEY (UserID) REFERENCES Users(UserID);
```

Невірне видалення або оновлення даних: У разі неправильного видалення або оновлення записів можуть виникнути ситуації, коли відсутні важливі дані або зв'язки між записами порушуються.

Рішення:

– Використання **триггерів** для перевірки, чи не буде порушена цілісність даних при оновленні або видаленні записів.

Приклад створення триггера для запобігання видаленню користувача, якщо у нього є неповернуті книги:

```
CREATE TRIGGER PreventDeleteUser
ON Users
FOR DELETE
AS
BEGIN
    IF EXISTS (SELECT 1 FROM IssuedBooks WHERE UserID = (SELECT UserID
FROM deleted))
    BEGIN
        RAISERROR ('Не можна видалити користувача, у якого є неповернуті
книги!', 16, 1);
        ROLLBACK;
    END
END;
```

4.3.3 Проблеми з безпекою

Безпека бази даних є критично важливою, особливо при зберіганні персональних даних користувачів, таких як їх контактна інформація, а також даних про книги, які можуть бути конфіденційними.

1. **Несанкціонований доступ до бази даних**: Якщо не реалізовано належних механізмів авторизації та автентифікації, база даних може бути

піддана несанкціонованому доступу, що призведе до витоку або пошкодження даних.

Рішення:

– Використання механізмів **аутентифікації та авторизації**, таких як ролі користувачів та привілеї для кожного користувача бази даних.

– Шифрування чутливих даних, таких як номери телефонів або електронні адреси, щоб захистити їх від несанкціонованого доступу.

Приклад створення ролей користувачів:

```
CREATE ROLE AdminRole;
CREATE ROLE UserRole;

GRANT SELECT, INSERT, UPDATE, DELETE ON Users TO AdminRole;
GRANT SELECT ON Users TO UserRole;
```

Недостатня обробка помилок: Якщо база даних не має належної обробки помилок, можуть виникнути проблеми з некоректною роботою запитів або навіть пошкодженням даних.

Рішення:

– Реалізація механізмів **обробки помилок** через **транзакції**, що дозволяє відкотити зміни у разі виникнення помилки.

– Використання журналювання помилок для моніторингу та аналізу проблем.

Приклад використання транзакцій:

```
BEGIN TRANSACTION;

COMMIT;
```

4.3.4 Проблеми з масштабуванням

При зростанні обсягів даних можуть виникнути проблеми з масштабуванням бази даних, що може призвести до зниження продуктивності або до необхідності виконувати складні операції для обробки великих наборів даних.

Рішення:

- **Шардінг**: Розбиття таблиць на частини (шарди) дозволяє розподіляти навантаження між кількома серверами або базами даних.
- **Реплікація**: Для покращення доступності та зменшення навантаження на основну базу даних можна використовувати реплікацію даних.

4.4 Резервне копіювання та відновлення

Забезпечення збереженості та цілісності даних є одним із ключових аспектів функціонування будь-якої інформаційної системи, зокрема й системи шкільної бібліотеки. Втрата даних унаслідок збою обладнання, помилки користувача або програмного забезпечення може призвести до серйозних наслідків, включаючи втрату облікових записів, інформації про книги та історію видачі. Тому впровадження системи резервного копіювання та можливості оперативного відновлення даних є обов’язковим етапом експлуатації бази даних.

Організація резервного копіювання

У даному проєкті для реалізації функцій резервного копіювання та відновлення даних використовується стандартний механізм Microsoft SQL Server Management Studio (SSMS), який дозволяє створювати повні, диференціальні та транзакційні резервні копії.

Для забезпечення надійності даних пропонується реалізувати таку схему резервного копіювання:

- **Повне резервне копіювання** – виконується щотижня у визначений день, наприклад, у неділю після закриття бібліотеки.
- **Диференціальне резервне копіювання** – виконується щодня, зберігаючи зміни з моменту останньої повної копії.
- **Резервне копіювання журналу транзакцій** – виконується щогодини, що дозволяє відновлювати дані до конкретного моменту у разі помилки або збою.

Копії зберігаються на окремому сервері або зовнішньому носії для запобігання втратам у разі пошкодження основного сховища.

Створення резервної копії бази даних

Приклад *SQL*-запиту для створення повного резервного копіювання бази даних:

```
BACKUP DATABASE Library_DB
TO DISK = 'D:\Library_DB_full.bak'
WITH FORMAT, INIT, NAME = 'Full Backup of Library_DB';
```

Цей запит створює повну резервну копію бази даних *Library_DB* у вказаному каталозі.

Відновлення бази даних із резервної копії

У разі необхідності база даних може бути відновлена за допомогою наступного *SQL*-запиту:

```
RESTORE DATABASE Library_DB
FROM DISK = 'D:\Library_DB_full.bak'
WITH REPLACE;
```

Цей запит відновлює базу даних *Library_DB* із зазначеної резервної копії, замінюючи наявну базу даних.

Рекомендації щодо резервного копіювання

Для забезпечення максимальної надійності та ефективності системи резервного копіювання рекомендовано:

- Зберігати кілька копій резервних файлів на різних носіях.
- Регулярно перевіряти працездатність резервних копій шляхом тестового відновлення.
- Автоматизувати процеси резервного копіювання за допомогою *SQL Server Agent* або системного планувальника завдань.
- Забезпечити захищений доступ до директорії, де зберігаються резервні копії.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено проект бази даних для управління шкільною бібліотекою з використанням системи управління базами даних (СУБД) Microsoft SQL Server. Оцінено сучасні методи проектування баз даних і розглянуто основні вимоги до структури бази даних для шкільної бібліотеки. Під час роботи були досягнуті наступні ключові результати:

1. **Формулювання вимог до бази даних**: Було сформульовано вимоги до бази даних для шкільної бібліотеки, які враховують потреби користувачів, збереження інформації про книги, користувачів і процеси видачі та повернення книг. Особливу увагу було приділено вимогам до забезпечення збереження цілісності даних, захисту персональної інформації користувачів і безпеки доступу до бази даних.

2. **Проектування структури бази даних**: У процесі роботи було розроблено схему бази даних, яка включає сутності, такі як **Книги**, **Користувачі**, **Видачі книг** та інші. Проектування бази передбачало нормалізацію даних для зменшення надмірності та забезпечення логічної організації інформації. Також було створено ER-діаграму, що відображає взаємозв’язки між сутностями, що дозволяє забезпечити правильну роботу бази даних.

3. **Вибір СУБД**: Для реалізації бази даних була вибрана система управління базами даних **Microsoft SQL Server**, яка забезпечує високий рівень надійності, безпеки та підтримку масштабованості. Окрім того, SQL Server має вбудовані можливості для виконання складних запитів, управління транзакціями та забезпечення цілісності даних, що робить її оптимальним вибором для шкільної бібліотеки.

4. **Реалізація операцій з даними**: Базу даних було налаштовано для виконання основних операцій управління бібліотекою: реєстрація користувачів, облік книг, видача та повернення книг, а також створення звітів

про стан бібліотеки. Запити SQL для виконання цих операцій були написані і протестовані. Окремо розглянуто запити для обліку книг та користувачів, а також створення звітів про прострочені або відсутні книги.

5. **Оптимізація бази даних**: Під час реалізації було проведено оптимізацію структури бази даних і SQL-запитів для підвищення продуктивності. Було використано індексацію, нормалізацію даних, а також оптимізацію запитів через **JOIN** та інші методи для скорочення часу виконання запитів і зменшення навантаження на систему.

6. **Перевірка працездатності**: База даних була протестована на реальних прикладах. Проведено перевірку на коректність виконання операцій видачі і повернення книг, а також тестування на великих обсягах даних для виявлення можливих проблем із продуктивністю. Під час тестування були виявлені деякі недоліки, які були усунуті за допомогою оптимізації запитів та покращення структури бази даних.

7. **Пропозиції щодо покращення**: У процесі виконання роботи було виявлено кілька можливостей для подальшого розвитку системи, зокрема:

– Розширення функціоналу для автоматичної генерації звітів про стан бібліотеки.

– Впровадження модулю для онлайн-реєстрації користувачів і віддаленого доступу до бібліотеки через веб-інтерфейс.

– Інтеграція бази даних з іншими системами управління шкільними ресурсами для зручності користувачів.

Загалом, виконана кваліфікаційна робота дозволяє створити надійну і ефективну систему для управління шкільною бібліотекою, що відповідає сучасним вимогам до зберігання та обробки інформації. База даних здатна забезпечити необхідну продуктивність, збереження цілісності даних і надійний захист інформації.

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

ВІДГУК
керівника кваліфікаційної роботи

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Владислав ПИСЬМЕННИЙ
(ім'я, прізвище)

1. *Кваліфікаційна робота на тему «Створення бази даних для управління бібліотекою з використанням SQL» виконана в ініціативному порядку.*
2. *Метою кваліфікаційної роботи є створення ефективної інформаційної системи, яка забезпечує автоматизацію обліку книг, видачі та повернення, а також аналітичну підтримку для бібліотечного персоналу.*
3. *Кваліфікаційна робота відповідає темі, затвердженій наказом начальника коледжу.*
4. *Кваліфікаційна робота виконана здобувачем освіти самостійно.*
5. *Здобувач освіти показав високі вміння роботи з літературними джерелами, аналіз теоретичного та практичного матеріалу, приймання обґрунтованих рішень, застосовування сучасних комп'ютерних інформаційних технологій.*
6. *Владислав ПИСЬМЕННИЙ показав достатній рівень дотримання вимог державних стандартів при виконанні кваліфікаційної роботи в цілому та оформленні пояснювальної записки .*
7. *Рівень виконаної кваліфікаційної роботи заслуговує оцінку «добре», відповідає набутих випускником знань, умінь та навичок, вимогам освітньої характеристики фахівця і можливість присвоєння йому кваліфікації фахівця освітнього ступеня «Фаховий молодший бакалавр» спеціальності 123 «Комп'ютерна інженерія».*

Керівник кваліфікаційної роботи

«____»_____2025 р. _____ Олександр ГРИНЧЕНКО
(підпис) (ім'я, прізвище)

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

РЕЦЕНЗІЯ
на кваліфікаційну роботу

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Владислав ПИСЬМЕННИЙ
(ім'я, прізвище)

1. Актуальність теми: Обрана тема кваліфікаційної роботи «Створення бази даних для управління бібліотекою з використанням SQL» є актуальною.

2. Кваліфікаційна робота відповідає темі, затвердженій наказом.

3. Завдання на виконання кваліфікаційної роботи виконано у повному обсязі.

4. В результаті виконання кваліфікаційної роботи було виконано проектування бази даних бібліотеки за допомогою SQL запитів.

5. Якість виконання пояснювальної записки та ілюстративного (графічного) матеріалу відповідає вимогам Державних стандартів.

6. В кваліфікаційній роботі зроблений акцент на дані отримані на практиці («живі» експерименти).

7. Кваліфікаційна робота заслуговує оцінку «добре».

Рецензент

_____
(науковий ступінь, посада)

«13» 06 2025 р. _____ Анна ДРОЗДОВА
(підпис) (ім'я, прізвище)

З рецензією ознайомлений _____ Владислав ПИСЬМЕННИЙ
(підпис) (ім'я, прізвище)