

Міністерство освіти і науки України
Національний авіаційний університет
Факультет кібербезпеки та програмної інженерії



Кафедра інженерії програмного забезпечення
Курсова робота
з дисципліни «Бази даних»
на тему «Розробка бази даних та СКБД «Облік клієнтів банку»

Виконав: студент групи ПІ-321Б
Кондратюк Андрій
Олександрович
Прийняла:
Шибицька Н. М.
Захищено:

Київ 2023

Зміст

Зміст	2
Вступ	4
Поточний стан і актуальність	4
Глобальні тенденції та технологічний прогрес	4
Актуальність і мета	4
Сфера застосування	5
Пояснювальна записка	6
Аналіз предметної області	6
Проблеми в існуючому програмному забезпеченні	6
Значення бази даних або інформаційної системи	6
Інформація, що вноситься до бази даних	6
Полегшення роботи кінцевих користувачів	7
Завдання та мета інформаційної системи	7
Постановка задачі	7
Завдання, необхідні до виконання	7
Інформація, що зберігається в базі даних	8
Функції, які повинна підтримувати інформаційна система:	8
Звіти, що генеруються системою	9
Проектування бази даних	10
Виявлення основних сутностей предметної області	10
Основні сутності	10
Зв'язки	10
Діаграма ER	10
Побудова схеми бази даних	12
Визначення сутностей, атрибутів та зв'язків	12
Нормалізація	13
Опис програмного забезпечення	15
Загальний опис програмного продукту	15
Опис архітектури програмної системи	16
Рівень представлення:	17
Рівень логіки	18
Рівень даних	18
Основні функції	18
Опис концептуальної моделі бази даних	20
Опис програмної реалізації	21

Сторінка Клієнта:	21
Сторінка Реєстрація клієнта:	21
Сторінка Вхід у систему	22
Сторінка таблиці Клієнти	23
Сторінка таблиці Рахунки	23
Сторінка таблиці Транзакції	24
Опис задач автоматизації та інтерфейсу користувача	25
Автоматизація додавання транзакцій:	25
Редагування та видалення транзакцій:	26
Сортування та фільтрація транзакцій:	27
Взаємодія з базою даних:	28
Висновки	30
Перелік посилань	31

Вступ

У динамічному ландшафті сучасних інформаційних технологій ефективне управління та організація даних відіграють ключову роль у різноманітних секторах, маючи особливе значення у сфері банківських та фінансових послуг. Оскільки фінансова індустрія продовжує розвиватися, потреба в надійних і складних системах керування базами даних (СУБД) стає все більш важливою.

У курсовій роботі розглядаються принципи, що лежать в основі побудови та підтримки реляційних баз даних, зосереджуючись на практичному застосуванні мови структурованих запитів (SQL) і використання відомої системи керування базами даних з відкритим кодом MySQL.

Поточний стан і актуальність

Оцінюючи поточний стан банківського та фінансового менеджменту, стає очевидним, що непідйомний обсяг даних про клієнтів, які генеруються щодня, вимагає ретельного ведення записів і впорядкованих процесів пошуку. Традиційних методів виявляється недостатньо для того, щоб впоратися зі складнощами та вимогами сучасних банківських операцій. Отже, зростає поштовх до впровадження технологічних рішень, які можуть забезпечити надійну структуру для ефективного управління, аналізу та доступу до інформації про клієнтів

Глобальні тенденції та технологічний прогрес

Спостерігаючи за глобальними тенденціями у фінансовому секторі, стає очевидним, що установи в усьому світі все більше покладаються на передові системи управління базами даних для підвищення своїх операційних можливостей. Інтеграція передових технологій не тільки сприяє безперебійній обробці даних, але й дає банкам можливість отримувати значущу інформацію, оптимізувати взаємодію з клієнтами та відповідати нормативним вимогам. Траєкторія глобального технологічного прогресу підкреслює необхідність для фінансових установ бути в курсі сучасних практик управління базами даних.

Актуальність і мета

Актуальність цієї курсової роботи підкреслюється гострою потребою в системі управління базою даних, що розроблена спеціально для обліку

банківських клієнтів. Цей проект спрямований на усунення існуючих прогалин в управлінні даними в банківському секторі, пропонуючи рішення, яке відповідає поточним галузевим стандартам. Розробляючи СУБД і відповідну базу даних, наша мета полягає в тому, щоб створити надійну інфраструктуру, яка не тільки централізує інформацію про клієнтів, але й оптимізує різні банківські процеси, підвищуючи загальну ефективність і обслуговування клієнтів.

Сфера застосування

Багатогранний характер банківських операцій вимагає комплексної бази даних, яка може обробляти різні типи даних, підтримувати складні запити та забезпечувати цілісність даних. Результат цієї курсової роботи буде виходити за межі теоретичного розуміння, її результатом стане практична реалізація СУБД, адаптована до унікальних потреб банківського обліку клієнтів. Обсяг цього проекту охоплює проектування, розробку та впровадження взаємопов'язаних таблиць, запитів SQL і зручного програмного забезпечення, оснащеного формами та звітами для полегшення взаємодії з базою даних.

Підсумовуючи, неминуча потреба в передовій СУБД для обліку клієнтів банку підкреслює своєчасність і важливість цієї курсової роботи. Розглядаючи наступні розділи ми зосереджуватимемося на перетворенні цих вступних ідей у відчутне, функціональне та ефективне рішення для керування базами даних, що сприятиме постійній еволюції управління інформацією в банківському секторі.

Пояснювальна записка

Аналіз предметної області

Предметна область обліку клієнтів банку є критично важливою сферою у фінансовому секторі, що охоплює безліч процесів і взаємодій, які вимагають ефективного управління даними. Щоб зрозуміти тонкощі цієї предметної області, ми повинні заглибитися в проблеми, які наразі існують у програмному забезпеченні, що використовується для обліку клієнтів банку, важливість використання бази даних або інформаційної системи, необхідну інформацію, яку потрібно зберігати, а також потенційні можливості, які інформаційна система може запропонувати кінцевим користувачам.

Проблеми в існуючому програмному забезпеченні

Існуюче програмне забезпечення у сфері обліку клієнтів банків часто стикається з проблемами, пов'язаними з надмірністю даних, неефективністю пошуку та обмеженою масштабованістю. Традиційні методи можуть призвести до створення розрізнених сховищ даних, що ускладнює підтримку єдиного бачення інформації про клієнтів у різних банківських операціях. Крім того, відсутність централізованої системи може призвести до труднощів при спробах впровадити комплексні заходи безпеки та дотримуватися нормативних стандартів.

Значення бази даних або інформаційної системи

Надійна база даних або інформаційна система є необхідною для вирішення виявлених проблем та впорядкування процесів обліку клієнтів банку. Централізація даних про клієнтів у структурованій базі даних полегшує ефективний пошук даних, мінімізує надмірність та підвищує цілісність даних. Система діє як сховище для різних типів даних, таких як профілі клієнтів, реквізити рахунків, історії транзакцій та контактна інформація, забезпечуючи безперешкодний доступ до них та їх обслуговування.

Інформація, що вноситься до бази даних

Інформація, що вноситься до бази даних, охоплює низку ключових об'єктів предметної області. Первинні об'єкти включають профілі клієнтів, кожен з яких характеризується такими атрибутами, як ім'я, адреса, контактні дані, ідентифікаційні номери та відповідна інформація про рахунок. Крім того, записи про транзакції потребують ретельної

реєстрації, включаючи такі деталі, як дата, сума, тип транзакції та пов'язана з нею інформація про рахунок. База даних повинна також містити інформацію про стан рахунків, забезпечуючи актуальне уявлення про фінансовий стан кожного клієнта.

Полегшення роботи кінцевих користувачів

Інформаційна система, призначена для обліку клієнтів банку, має спрощувати роботу кінцевих користувачів, підвищуючи ефективність та надаючи зручний інтерфейс. Такі функції, як швидкий пошук інформації про клієнтів, спрощена обробка транзакцій та автоматизована звітність, можуть значно зменшити навантаження на банківський персонал. Добре розроблена система може також сприяти кращому обслуговуванню клієнтів, надаючи своєчасну і точну інформацію, підвищуючи загальну задоволеність клієнтів

Завдання та мета інформаційної системи

Завдання, які повинна виконувати інформаційна система в цій предметній області, включають оновлення інформації про клієнтів в режимі реального часу, сприяння безперебійній обробці транзакцій, створення вичерпних звітів для аналізу. Мета полягає у створенні централізованого хабу, який не лише ефективно зберігає та організовує дані, але й надає інструменти для ефективного прийняття рішень, управління ризиками та дотримання регуляторних вимог.

Отже, аналіз предметної області обліку клієнтів банку підкреслює гостру потребу в складній базі даних або інформаційній системі. Вирішуючи існуючі проблеми з програмним забезпеченням і комплексно збираючи та керуючи даними, пов'язаними з клієнтами, запропонована система має на меті підвищити ефективність, безпеку та загальну функціональність банківських операцій, сприяючи безперервному розвитку інформаційного менеджменту у фінансовому секторі.

Постановка задачі

Завдання, необхідні до виконання

Управління даними про клієнтів

Завдання: Створити централізовану базу даних для зберігання профілів клієнтів з такими атрибутами, як ім'я, адреса і номер телефону.

Ідентифікаційні номери.

Завдання: Реалізувати механізм оновлення для забезпечення точності та актуальності інформації про клієнтів.

Обробка транзакцій

Завдання: Розробити функції для безперебійної обробки транзакцій, включаючи депозити, зняття коштів і перекази, з негайним оновленням залишків на рахунках.

Завдання: Впровадити систему відстеження статусів транзакцій для надання своєчасної інформації про завершені та очікувані транзакції.

Цілісність і безпека даних

Завдання: Впровадити заходи для підтримки цілісності даних, мінімізації надмірності та забезпечення відповідності регуляторним стандартам.

Завдання: Впровадити надійні протоколи безпеки для захисту конфіденційної інформації про клієнтів та дотримання вимог щодо захисту даних.

Інформація, що зберігається в базі даних

Профілі клієнтів:

Специфіка: Ім'я, адреса, контактні дані, ідентифікаційні номери (наприклад, номер соціального страхування), тип рахунку та пов'язана з ним інформація.

Записи про транзакції

Специфіка: Дата, сума, тип транзакції (наприклад, поповнення, зняття, переказ), пов'язана інформація про рахунок і статус транзакції.

Інформація про стан рахунку

Специфіка: Поточний баланс рахунку, статус рахунку (активний, неактивний, закритий) та історія транзакцій.

Функції, які повинна підтримувати інформаційна система:

Сортування

Функція: Увімкніть сортування даних про клієнтів на основі таких атрибутів, як ім'я, тип рахунку або залишок на рахунку.

Пошук

Функція: Реалізувати функцію пошуку, що дозволяє отримувати інформацію про клієнта за певними параметрами, такими як ідентифікатор клієнта або номер рахунку.

Фільтрація

Функція: Увімкніть фільтрацію транзакцій на основі таких критеріїв, як діапазон дат, тип транзакції або статус рахунку.

Формулювання запиту

Запит: Отримати загальний баланс конкретного клієнта.

Запит: Генерувати список транзакцій за вказаний діапазон дат.

Звіти, що генеруються системою

Виписки за рахунками клієнтів

Звіт: Детальний звіт про транзакції та залишки на рахунках за певний період.

Звіти по історії транзакцій

Звіт: Компіляція історії транзакцій для окремих клієнтів.

Проектування бази даних

Виявлення основних сутностей предметної області

Після ретельного аналізу предметної області обліку клієнтів банку можна виділити кілька ключових об'єктів, кожен з яких відіграє ключову роль у загальній інформаційній системі. Зв'язки між цими сутностями сприяють цілісному представленню даних про клієнтів. Для візуалізації цих сутностей і зв'язків діаграма "сутність-зв'язок" (ER-діаграма) слугує ефективним інструментом для ілюстрації структури бази даних.

Основні сутності

Клієнт

Атрибути: ID (Primary Key), Ім'я, Прізвище, Номер телефону, ІПН.

Рахунок

Атрибути: ID (Primary Key), ID клієнта (Foreign Key), Тип, Баланс, Статус.

Транзакція

Атрибути: ID (Primary Key), З якого рахунку (Foreign Key), До якого рахунку, Дата й час, Сума, Статус.

Зв'язки

Customer - Account (One-to-Many)

Кожен клієнт може мати кілька рахунків.

Кожен рахунок належить одному клієнту.

Відображає зв'язок між сутностями "Клієнт" і "Рахунок".

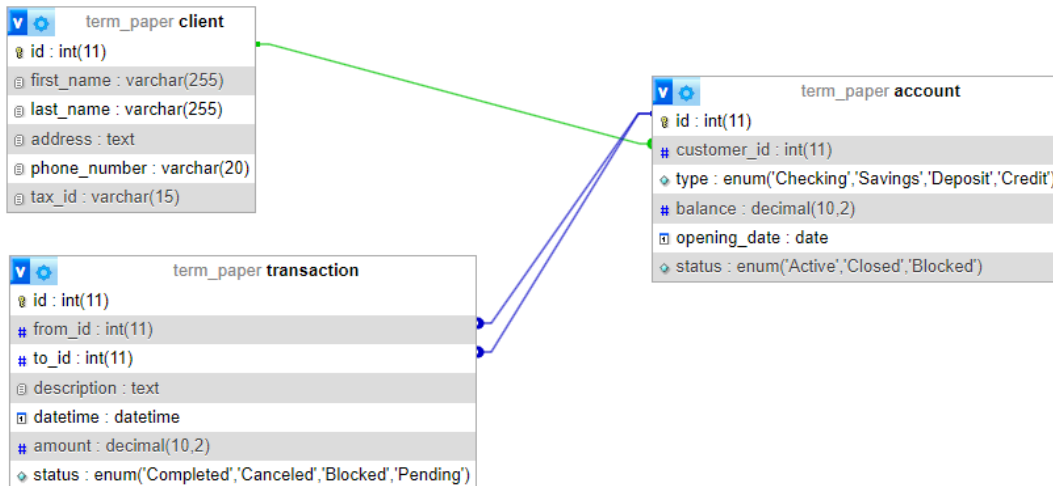
Account - Transaction (One-to-Many)

Кожен рахунок може мати кілька транзакцій.

Кожна транзакція належить одному рахунку.

Відображає зв'язок між сутностями "Рахунок" і "Транзакція".

Діаграма ER



Пояснення

Сутність "Клієнт" представляє фізичних осіб з такими атрибутами, як унікальний ідентифікатор (ID), ім'я, адреса, телефон та ІПН.

Сутність "Рахунок" зберігає інформацію про окремі рахунки, включаючи унікальний номер рахунку (ID), пов'язаного з ним клієнта (CustomerID як зовнішній ключ), тип рахунку, поточний баланс і статус.

Сутність "Транзакція" фіксує деталі кожної транзакції з унікальним ідентифікатором транзакції (ID), асоційованими рахунками (FromID і ToID як зовнішні ключі), датою, сумою, типом і статусом.

Зв'язки (показані лініями, що з'єднують сутності) відображають, як сутності пов'язані між собою. Зв'язок "один-до-багатьох" між Клієнтом і Рахунком означає, що один клієнт може мати кілька рахунків.

Аналогічно, зв'язок "один-до-багатьох" між Рахунком і Транзакцією означає, що один рахунок може мати кілька транзакцій.

Побудова схеми бази даних

Визначення сутностей, атрибутів та зв'язків

Ідентифікація сутностей, атрибутів і зв'язків є вирішальним кроком у розробці схеми реляційної бази даних, оскільки вона закладає основу для організації та структурування даних у спосіб, який точно відображає реальний сценарій. У цьому випадку було визначено три основні сутності - Клієнт, Рахунок і Транзакція. Кожна сутність представляє окремий об'єкт або концепцію в системі. Наступний крок передбачає визначення атрибутів, пов'язаних з кожною сутністю, які є властивостями або характеристиками, що їх описують. Наприклад, сутність "Клієнт" включає такі атрибути, як ідентифікатор, ім'я, прізвище, дата народження та інші, що містять основну інформацію про клієнта. Аналогічно, сутність Рахунок містить такі атрибути, як тип, власник, баланс і дата відкриття, що надають детальну інформацію про різні фінансові рахунки. Сутність "Транзакція", що представляє фінансову взаємодію між рахунками, має такі атрибути, як сума, статус, а також дата і час здійснення.

Відносини між сутностями встановлюють зв'язки і залежності, визначаючи, як дані пов'язані між собою в базі даних. У цьому контексті між клієнтом і рахунком існує зв'язок "один до багатьох", що означає, що один клієнт може мати кілька рахунків, тоді як кожен рахунок належить лише одному клієнту. Це відображається в реляційній схемі через зовнішній ключ, який пов'язує таблицю Рахунок з таблицею Клієнт. Тим часом, сутність "Транзакція" включає зв'язок "багато-до-багатьох" між двома обліковими записами, вказуючи на те, що між різними обліковими записами можуть відбуватися численні транзакції. Ці ідентифіковані сутності, атрибути та зв'язки слугують будівельними блоками для побудови ефективної та добре організованої схеми реляційної бази даних, що сприяє ефективному зберіганню та пошуку інформації.

Сутність Клієнт

Атрибути: ID (первинний ключ), ім'я, прізвище, дата народження, дата реєстрації, податковий номер, номер телефону.

Зв'язки: Клієнти можуть бути пов'язані з декількома рахунками.

Сутність Рахунок

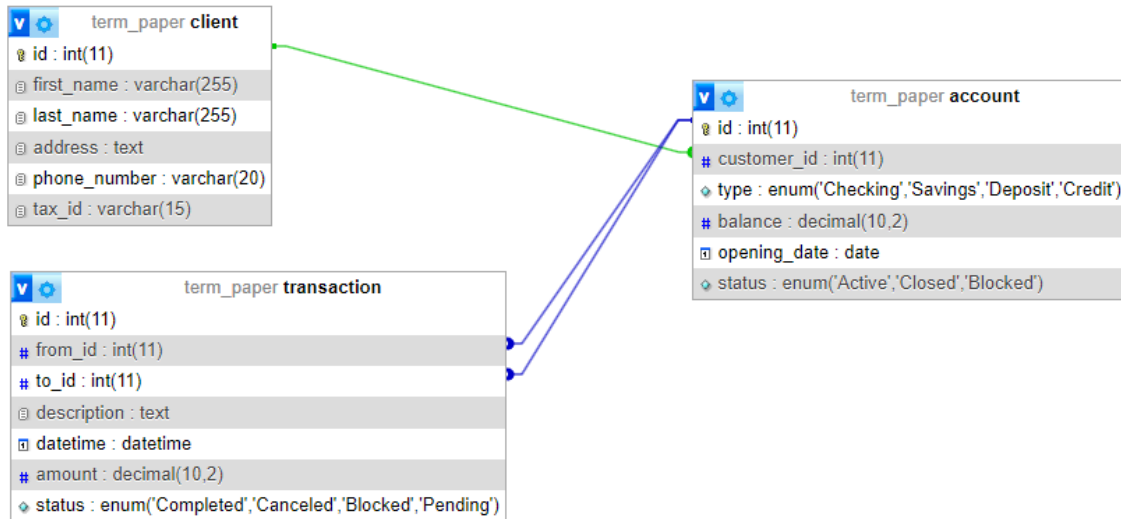
Атрибути: Ідентифікатор (первинний ключ), тип, власник (зовнішній ключ, що посиляється на клієнта), баланс, статус, дата відкриття.

Відносини: Кожен рахунок належить клієнту.

Транзакція (Transaction Entity):

Атрибути: Рахунок, з якого здійснюється (зовнішній ключ Account), рахунок, на який здійснюється, сума, статус, дата і час здійснення.

Зв'язки: Транзакція пов'язана з двома рахунками - джерелом і призначенням.



Нормалізація

Щоб оцінити, чи знаходиться схема у 3-й нормальній формі (3НФ), нам потрібно дослідити функціональні залежності та нормалізацію.

Функціональні залежності:

Для сутності Клієнт:

ID → ім'я, прізвище, дата народження, дата реєстрації, податковий номер, номер телефону

(Припускаючи, що податковий номер є унікальним для кожного клієнта, як це часто буває)

Для сутності Рахунок:

ідентифікатор → тип, власник, баланс, статус, дата відкриття

власник → ідентифікатор, тип, баланс, статус, дата відкриття

Для сутності Операція:

(рахунок, з якого здійснюється, рахунок, на який здійснюється) → сума, статус, дата і час здійснення

Процес нормалізації:

Перша нормальна форма (1НФ):

Всі атрибути повинні бути атомарними. У нашій схемі атрибути вже є атомарними, тому вона задовольняє 1НФ.

Друга нормальна форма (2НФ):

Для сутності Клієнт:

Немає часткових залежностей; вона вже відповідає 2НФ.

Для сутності Рахунок:

Часткових залежностей не існує, вона вже є в 2 НФ.

Для сутності Транзакції:

(рахунок, з якого здійснюється, рахунок, на який здійснюється) → сума, статус, дата і час здійснення (Часткових залежностей немає)

3-я нормальна форма (3NF):

Для сутності Клієнт:

Перехідних залежностей немає, вона вже є в 3 НФ.

Для сутності Рахунок:

Транзитивних залежностей не існує, вона вже є в 3 НФ.

Для сутності Транзакції:

(рахунок, з якого здійснюється, рахунок, на який здійснюється) → сума, статус, дата і час здійснення (транзитивних залежностей немає)

Опис програмного забезпечення

Загальний опис програмного продукту

Програмний продукт, надійна система управління фінансами, працює в комплексному середовищі, що включає XAMPP для локального сервера Apache, інтерфейс HTML + CSS для взаємодії з користувачем, PHP для внутрішньої обробки даних і JavaScript для інтерактивного інтерфейсу. Функціональність бази даних системи забезпечується реляційною системою управління базами даних (СКБД) MySQL.



Для створення локального серверного середовища Apache використовується XAMPP, крос-платформне рішення для веб-серверів. Цей вибір сприяє безперешкодній розробці, тестуванню та розгортанню системи менеджменту клієнтів банку, забезпечуючи стабільну основу для програми.

Інтерфейс користувача розроблений з використанням HTML і CSS, що забезпечує адаптивний і візуально привабливий інтерфейс. HTML структурує контент, тоді як CSS використовується для стилізації та макетування, створюючи інтуїтивно зрозумілий і зручний інтерфейс для взаємодії з системою фінансового менеджменту.



PHP слугує мовою сценаріїв внутрішнього інтерфейсу, обробляючи дані на стороні сервера та полегшуючи зв'язок між інтерфейсом і базою даних MySQL. Його інтеграція з базою даних MySQL має вирішальне значення для виконання операцій з базою даних, таких як отримання та оновлення інформації про клієнтів, управління рахунками та обробка транзакцій. Поєднання PHP та MySQL дозволяє ефективно обробляти дані, забезпечуючи безперебійну роботу системи управління фінансами.



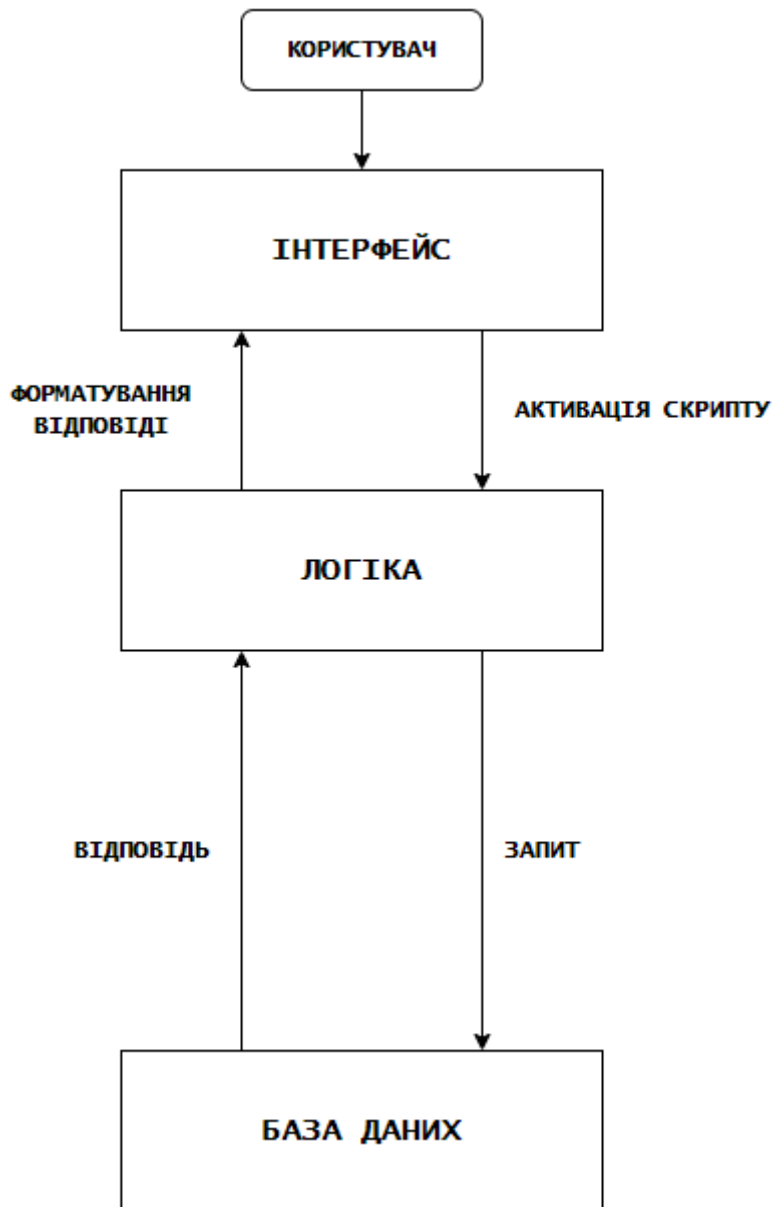
JavaScript відіграє вирішальну роль у підвищенні інтерактивності інтерфейсу системи. Він використовується для створення динамічних та адаптивних користувацьких інтерфейсів, що дозволяє оновлювати інформацію та взаємодіяти в режимі реального часу без необхідності перезавантаження сторінок. Це додає ще один рівень оперативності та залучення користувачів до загальної системи фінансового менеджменту.



СУБД MySQL була обрана завдяки її перевірений надійності, масштабованості та широкому використанню в галузі. Її відкритий вихідний код сприяє економічно ефективній розробці, а відповідність стандарту ACID забезпечує цілісність даних, що є важливою характеристикою для фінансових додатків. Підтримка спільноти MySQL та обширна документація сприяють безперебійному процесу розробки та постійному обслуговуванню, що робить її ідеальним вибором для базової системи управління базами даних системи фінансового менеджменту.

Опис архітектури програмної системи

Програмна система має багаторівневу архітектуру, що складається з рівнів презентації, додатку та даних. Діаграми ілюструють взаємозв'язки та взаємодію між цими компонентами.



Рівень представлення:

На рівні представлення система включає модуль інтерфейсу користувача (UI), який відповідає за відображення графічного інтерфейсу та взаємодію з кінцевими користувачами. Він складається з домашньої сторінки (index.html) з якої можна перейти до сторінок окремих таблиць, сторінки реєстрації користувача та логіну (входу в систему), де в свою чергу є можливість створювати транзакції та переглядати інформацію про користувача і його рахунки.

Рівень логіки

Додаток базується на серверних .php-скриптах, які відповідають за обробку взаємодії з елементами користувацького інтерфейсу. Кожна сторінка має свої власні файли, такі як `add_client.php`, `remove_transaction.php` і `index.php`, які зберігають логіку програмного додатку та забезпечують виконання різноманітних операцій.

У той час як серверні скрипти забезпечують обробку запитів та виконання операцій на бізнес-рівні, клієнтська сторона реалізована за допомогою скриптів на мові JavaScript, які вбудовані в HTML-файли чи розміщені в окремих файлах. Ці скрипти відповідають за взаємодію з користувацьким інтерфейсом, виконання GET та POST запитів, а також за оновлення сторінки, забезпечуючи користувачеві зручний та ефективний досвід взаємодії з додатком.

Рівень даних

На рівні даних використовується база даних `term_paper`, яка включає в себе різні таблиці, такі як `client`, `account`, та `transaction`. Кожна з цих таблиць має визначену структуру для зберігання відповідних даних. Наприклад, таблиця `client` може містити інформацію про клієнтів, таблиця `account` - дані про рахунки, а таблиця `transaction` - записи про транзакції.

Ця структура бази даних дозволяє ефективно зберігати та організовувати дані на серверному рівні. Серверні .php-скрипти, які активуються при взаємодії з користувацьким інтерфейсом, використовують ці дані для виконання операцій, таких як додавання клієнтів, видалення транзакцій, індексування тощо.

При цьому, відповіді, які надсилаються з сервера до клієнта, відповідають визначеній структурі бази даних. Це дозволяє забезпечити стандартизований та відформатований обмін даними між клієнтською та серверною частинами додатку, забезпечуючи коректну роботу та взаємодію між різними компонентами системи.

Основні функції

Функціональні можливості створеної СКБД включають в себе CRUD-операції з записами в базі даних а також користувацький інтерфейс для простої взаємодії від особи клієнта банку:

1. Сторінка “Клієнти”:

1.1 Відображення таблиці “client”

1.2 Додавання клієнта

1.3 Редагування інформації про наявного клієнта

1.4 Видалення клієнта

1.5 Пошук клієнта за окремими полями (ID, ім’я)

2. Сторінка “Рахунки”

2.1 Відображення таблиці “account”

2.2 Додавання рахунку

2.3 Редагування інформації про наявний рахунок

2.4 Видалення рахунку

2.5 Пошук рахунку за окремими полями

3. Сторінка “Транзакції”

3.1 Відображення таблиці “transactions”

3.2 Додавання транзакції

3.3 Редагування інформації про наявну транзакцію

3.4 Видалення транзакції

3.5 Пошук транзакції за окремими полями

4. Сторінка “Дім”

4.1 Перехід на інші сторінки СКБД

5. Сторінка “Реєстрація клієнта”

5.1 Введення інформації про клієнта

5.2 Перевірка введеної інформації на коректність

5.3 Внесення клієнта у базу даних

6. Сторінка “Вхід у систему”

6.1 Перевірка коректності введених даних

6.2 Перевірка на наявність даних у системі

6.3 Вхід в систему по номеру телефону

7. Сторінка “Клієнт”

7.1 Перегляд інформації про клієнта

7.2 Перегляд рахунків клієнта

7.3 Перегляд тотального балансу

7.3 Створення транзакції

7.3 Перегляд 5 останніх транзакцій для клієнта

Опис концептуальної моделі бази даних

У концептуальній моделі бази даних ми розробили три основні таблиці для управління клієнтськими даними в системі управління клієнтськими даними банку: клієнт, рахунок і транзакція. Кожна таблиця структурована з певними полями, типами даних та обмеженнями, пристосованими до вимог системи.

Таблиця клієнтів слугує основним сховищем інформації про клієнтів. Первинний ключ, `id`, є цілим числом довжиною 11, що забезпечує унікальний ідентифікатор для кожного клієнта. Таблиця містить такі поля, як `ім'я_та_прізвище` та `прізвище` змінного символічного типу, що вміщують до 255 символів кожне. Поле `адреса` має текстовий тип, що забезпечує гнучкість для зберігання довгих адрес. Поле `phone_number` має тип `varchar(20)`, що дозволяє зберігати різні формати телефонних номерів. Поле `tax_id` має тип `varchar(15)`, що відповідає типовій довжині податкового ідентифікаційного номера.

Таблиця `account` призначена для управління рахунками клієнтів. Первинний ключ `id` є цілим числом довжиною 11. Поле `customer_id` є зовнішнім ключем, що посилається на `id` в таблиці клієнтів, встановлюючи зв'язок між клієнтами та їхніми рахунками. Поле `type` є переліком з варіантами "Розрахунковий", "Ощадний", "Депозитний" і "Кредитний". Поле `balance` є десятковим числом (10,2), що представляє грошове значення зі значенням за замовчуванням 0.00. Поле `дата_відкриття` має тип `дата` зі значенням за замовчуванням, встановленим на поточну дату. Поле `status` є переліком з варіантами "Активний", "Закритий" та "Заблокований", що відображає поточний стан рахунку.

Таблиця транзакцій містить деталі фінансових операцій. Первинний ключ, `id`, є цілим числом довжиною 11. Поля `from_id` та `to_id` є цілими числами, що представляють ідентифікатори рахунку-джерела та рахунку-призначення відповідно. Поле `description` має текстовий тип, що дозволяє надати вичерпний опис транзакції. Поле `datetime` має тип `datetime` і містить мітку часу транзакції, за замовчуванням встановлену на поточний час. Поле `сума` є десятковим (10,2), що відображає грошову

вартість транзакції. Поле статусу є переліком з варіантами "Завершено", "Скасовано", "Заблоковано" і "Очікує", що вказує на поточний стан транзакції.

Опис програмної реалізації

Сторінка Клієнта:

Код цього компоненту створює HTML-сторінку для відображення інформації про клієнта та його облікові записи. Спочатку він містить основну структуру HTML-документа, підключає зовнішні файли зі стилями (styles.css) та скриптом (script.js). Головний функціонал розміщений у тілі HTML-документа.

На сторінці відображається форма для введення номера телефону клієнта. Якщо номер телефону валідний, відбувається підключення до бази даних (MySQL) за допомогою PHP. Здійснюється запит до бази для отримання інформації про клієнта та його облікові записи на основі введеного номера телефону.

Цікавий момент полягає в тому, що сторінка використовує AJAX-технологію для асинхронного виклику серверного скрипта (script.js та функція submitTransactionForm) без перезавантаження сторінки. Форми для введення даних транзакцій відображаються динамічно, і їх відображення можна включити або вимкнути за допомогою кнопок.

Сторінка також використовує функцій для взаємодії з базою даних, включаючи виконання SQL-запитів, обробку результатів та виведення даних на сторінку. Запити включають JOIN-операції для отримання інформації про транзакції та взаємодію з клієнтами та їх обліковими записами.

Сторінка Реєстрація клієнта:

Код створює сторінку для реєстрації клієнта у банку з використанням HTML, CSS, та JavaScript, а також мови програмування PHP для взаємодії з базою даних MySQL.

HTML-структура містить форму для введення даних клієнта, таких як ім'я, прізвище, адреса, номер телефону та інших особистих даних. Скрипти з бібліотеки Materialize використовуються для поліпшення вигляду та взаємодії елементів форми. На серверній стороні, у файлі register.php, відбувається підключення до бази даних та вставка нового клієнта з отриманими з форми даними.

Якщо операція успішна, виводиться сповіщення про успішну реєстрацію, і користувач перенаправляється на головну сторінку. JavaScript-скрипт app.js використовує бібліотеку jQuery для взаємодії з DOM. Скрипт включає функціонал, що дозволяє активувати кнопку "Зареєструватися" тільки тоді, коли всі обов'язкові поля заповнені, а також адреса введена.

Цікаві моменти включають в себе використання бібліотеки Materialize для стилізації та взаємодії форми і використання AJAX для асинхронної передачі даних на сервер без перезавантаження сторінки.

Сторінка Вхід у систему

Код представляє собою сторінку входу, де користувач вводить свій номер телефону для авторизації. Нижче подано короткий опис ключових елементів цього коду:

HTML-структура: Визначає основну структуру сторінки входу. Містить форму для введення номера телефону та кнопку "Увійти". Додано виклик скрипта validatePhoneNumber() для валідації номера телефону перед відправкою форми.

JavaScript-скрипт: Містить функцію validatePhoneNumber(), яка викликається при відправці форми. Вона перевіряє правильність формату введеного номера телефону за допомогою регулярного виразу. У випадку помилки виводиться вікно повідомлення, і форма не відправляється.

PHP-код: Обробляє відправку форми. З'єднання з базою даних MySQL і валідація номера телефону. Якщо номер телефону існує в базі даних, користувач перенаправляється на сторінку облікового запису клієнта. Якщо номер не знайдено, виводиться повідомлення про помилку.

Цей код практичний для створення простої сторінки входу з базовим функціоналом обробки та перевірки введених даних для цього проекту.

Сторінка таблиці Клієнти

Код представляє собою веб-сторінку для відображення та управління таблицею клієнтів. Давайте розглянемо ключові елементи цього коду:

HTML-структура: Визначає основну структуру сторінки. Містить форму для додавання клієнта, таблицю для відображення інформації про клієнтів та модальне вікно для редагування даних клієнта. Також, є кнопка для повернення на головну сторінку.

JavaScript: Включає скрипт `client.js`, який містить логіку для взаємодії з клієнтською стороною. Наприклад, функції для додавання, редагування та видалення клієнта, а також для пошуку клієнта за допомогою Ajax-запиту. Також, визначена функція `closeEditModal()` для закриття модального вікна.

CSS: Зовнішній файл стилів (`styles.css`), який використовується для визначення зовнішнього вигляду елементів сторінки.

Форма для додавання клієнта: Містить текстові поля для введення інформації про нового клієнта, а також кнопки для додавання та пошуку клієнта.

Таблиця клієнтів: Виводить інформацію про клієнтів у вигляді таблиці з колонками: ID, Ім'я, Прізвище, Адреса, Номер телефону, ІПН та Дія. Використовується елемент `<tbody>` для динамічного оновлення таблиці.

Модальне вікно для редагування клієнтів: Використовується для введення та редагування інформації про клієнта. Містить текстові поля та кнопки для оновлення, видалення та закриття модального вікна.

Сторінка таблиці Рахунки

Даний HTML-код представляє собою веб-сторінку для керування інформацією про рахунки клієнтів. Давайте розглянемо ключові аспекти структури та функціоналу:

HTML-структура: Сторінка використовує елементи HTML для визначення структури та вигляду. Включає форму для додавання нових рахунків, таблицю для відображення інформації про рахунки, а також

модальне вікно для редагування рахунків. Присутня кнопка для повернення на головну сторінку.

JavaScript: Скрипт `account.js` забезпечує логіку взаємодії із клієнтською стороною. Включає функції для додавання, редагування та видалення рахунків, а також для пошуку за допомогою Аґах-запиту. Також визначена функція для закриття модального вікна.

CSS: Використовується зовнішній файл стилів (`styles.css`), щоб визначити зовнішній вигляд елементів сторінки.

Форма для додавання рахунку: Містить текстові поля та випадаючі списки для введення інформації про новий рахунок, а також кнопки для додавання та пошуку рахунку.

Таблиця рахунків: Показує інформацію про рахунки у вигляді таблиці з колонками: ID, Ідентифікатор клієнта, Тип рахунку, Дата відкриття, Статус рахунку та Дія. Використовується елемент `<tbody>` для динамічного оновлення таблиці.

Модальне вікно для редагування рахунків: Використовується для введення та редагування інформації про рахунок. Містить текстові поля та кнопки для оновлення, видалення та закриття модального вікна.

Цей HTML-код призначений для створення функціональної сторінки для управління рахунками клієнтів, їхнім додаванням, редагуванням та видаленням. Детальніша інформація про взаємодію з клієнтською стороною міститься в скрипті `account.js`, а стилі оформлення знаходяться в файлі `styles.css`.

Сторінка таблиці Транзакції

Даний HTML-код визначає веб-сторінку для управління інформацією про транзакції між обліковими записами клієнтів. Основні характеристики структури та функціоналу наведено нижче:

HTML-структура: Сторінка містить форму для додавання нових транзакцій, таблицю для відображення інформації про транзакції, модальне вікно для редагування транзакцій та декілька інтерактивних елементів, таких як блоки, що розкриваються та кнопка "На головну".

JavaScript: Скрипт transaction.js відповідає за логіку взаємодії із клієнтською стороною. Забезпечує функціонал для додавання, редагування, видалення та сортування транзакцій. Також визначає функції для розкриваючихся блоків та переходу на головну сторінку.

CSS: Зовнішній файл стилів (styles.css) використовується для визначення зовнішнього вигляду елементів сторінки.

Форма для додавання транзакцій: Містить текстові поля та випадаючі списки для введення інформації про нову транзакцію, включаючи ID облікового запису, опис, дату та час, суму та статус транзакції.

Таблиця транзакцій: Показує інформацію про транзакції у вигляді таблиці з колонками: ID, З облікового запису, На обліковий запис, Опис, Дата та час, Сума, Статус транзакції та Дія.

Модальне вікно для редагування транзакцій: Використовується для введення та редагування інформації про транзакцію. Містить текстові поля, випадаючі списки та кнопки для оновлення, видалення та закриття модального вікна.

Сортування транзакцій: Реалізовано випадаючий блок для вибору параметрів сортування, таких як дата, сума та статус транзакції.

Кнопка "На головну": Дозволяє перейти на головну сторінку.

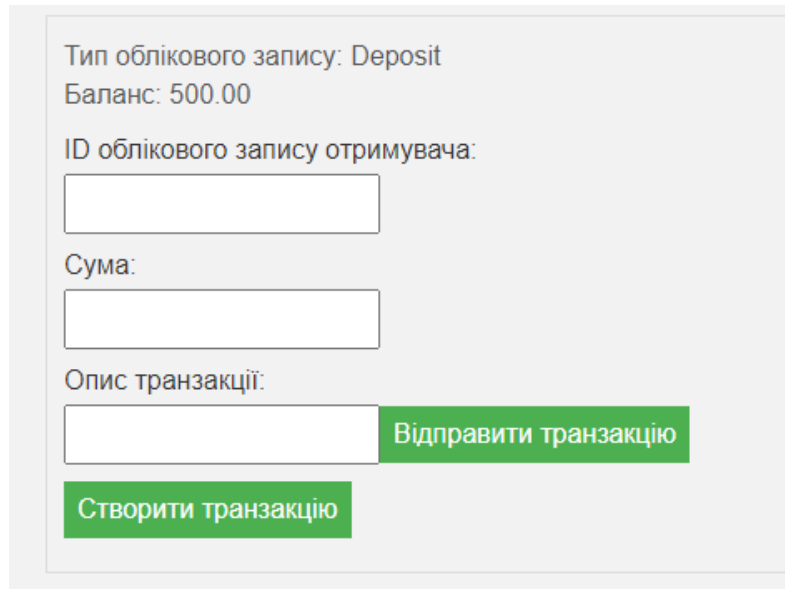
Цей HTML-код призначений для створення функціональної сторінки для управління транзакціями між обліковими записами клієнтів. Більш детальна інформація про взаємодію з клієнтською стороною міститься в скрипті transaction.js, а стилі оформлення знаходяться в файлі styles.css.

Опис задач автоматизації та інтерфейсу користувача

Автоматизація додавання транзакцій:

В системі реалізовано зручний інтерфейс для додавання нових транзакцій. На екранній формі "Додати транзакцію" користувач може

обрати облікові записи, вказати опис, дату та час, суму та статус транзакції. Алгоритм автоматизації передбачає попередню перевірку введених даних на коректність та унікальність, а після цього виконує вставку нової транзакції в базу даних. Це спрощує та прискорює процес введення нових транзакцій для користувачів системи.



The screenshot shows a web form for creating a transaction. It has a light gray background. At the top, it displays 'Тип облікового запису: Deposit' and 'Баланс: 500.00'. Below this is a label 'ID облікового запису отримувача:' followed by an empty text input field. Then, a label 'Сума:' is followed by another empty text input field. Below that is a label 'Опис транзакції:' followed by a third empty text input field. To the right of the description input field is a green button with white text 'Відправити транзакцію'. At the bottom left of the form area is a green button with white text 'Створити транзакцію'.

Редагування та видалення транзакцій:

Система надає можливість зручного редагування та видалення існуючих транзакцій. Модальне вікно "Редагувати транзакцію" автоматично заповнюється поточними значеннями транзакції для подальшої модифікації. Реалізований алгоритм автоматичної перевірки та оновлення даних в базі після збереження змін або видалення транзакції. Це надає користувачам зручний спосіб управління історією транзакцій.

Опис	Дата та час
Товарі	2023-12-31 00:00:00

Редагувати транзакцію

З облікового запису з ID:

На обліковий запис з ID:

Опис:

Дата та час:

Сума:

Статус транзакції:

Оновити транзакцію

Видалити транзакцію

Закрити

Сортування та фільтрація транзакцій:

З метою полегшення взаємодії з обсягом даних, система надає можливість сортування та фільтрації транзакцій. На екранній формі "Сортувати транзакції" користувач може вибрати параметри сортування за сумою або датою, а також встановити фільтри за статусом транзакції та датою виконання. Алгоритм автоматизації обробки цих запитів забезпечує швидкий та ефективний доступ до необхідної інформації.

ID	З облікового запису	На обліковий запис	Опис	Дата та час	Сума	Статус транзакції	Дія
27	30	36	(2023-12-06 01:07:18	666.00	Completed	Edit

Сортувати транзакції

Початкова дата та час:

Кінцева дата та час:

Статус транзакції

Сортувати за:

Сортувати транзакції

Взаємодія з базою даних:

Наведені SQL-скрипти для створення таблиці транзакцій та основні SQL-запити. SQL-скрипт для створення таблиці транзакцій визначає структуру даних, таких як облікові записи, опис, дата та час, сума та статус транзакції. Основні SQL-запити дозволяють вибрати та оновлювати дані з таблиці транзакцій за різними умовами.

```
-- Create the term_paper database
```

```
CREATE DATABASE IF NOT EXISTS term_paper;
```

```
-- Switch to the term_paper database
```

```
USE term_paper;
```

```
-- Create the client table
```

```
CREATE TABLE IF NOT EXISTS client (
    id INT(11) PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    address TEXT,
    phone_number VARCHAR(20),
    tax_id VARCHAR(15)
);
```

```
-- Create the account table
```

```
CREATE TABLE IF NOT EXISTS account (
    id INT(11) PRIMARY KEY,
    customer_id INT(11),
    type ENUM('Checking', 'Savings', 'Deposit', 'Credit'),
    balance DECIMAL(10,2) DEFAULT 0.00,
    opening_date DATE DEFAULT CURRENT_DATE,
    status ENUM('Active', 'Closed', 'Blocked'),
    FOREIGN KEY (customer_id) REFERENCES client(id)
);
```

```
-- Create the transaction table
CREATE TABLE IF NOT EXISTS transaction (
  id INT(11) PRIMARY KEY,
  from_id INT(11),
  to_id INT(11),
  description TEXT,
  datetime DATETIME DEFAULT CURRENT_TIMESTAMP,
  amount DECIMAL(10,2),
  status ENUM('Completed', 'Canceled', 'Blocked', 'Pending')
);
```

Код проекту в деталях можна переглянути за посиланням:

github.com/levndays/term_paper

Висновки

У даній курсовій роботі ми ретельно розглянули принципи та практичні аспекти побудови реляційних баз даних, зокрема в контексті банківських та фінансових послуг. Спробували проаналізувати сучасні вимоги до управління даними в цій галузі та визначити ключові виклики, які стоять перед фінансовою індустрією в умовах динамічного ландшафту інформаційних технологій.

З урахуванням глобальних тенденцій в сфері фінансів та технологічного прогресу визначили, що управління базами даних відіграє критичну роль у покращенні ефективності банківських операцій, а також в забезпеченні дотримання нормативів та відповіді на зростаючі потреби клієнтів.

Основна мета курсової роботи полягає в розробці системи управління базою даних, спеціально адаптованої для банківського обліку клієнтів. В ході проекту ми працювали над створенням надійної інфраструктури, яка об'єднує інформацію про клієнтів та оптимізує банківські процеси, забезпечуючи при цьому загальну ефективність та високий рівень обслуговування клієнтів.

Зазначимо, що обсяг проекту включав в себе проектування, розробку та впровадження взаємопов'язаних таблиць, використання SQL-запитів та створення зручного програмного забезпечення з формами та звітами. Це практичне рішення спрямоване на полегшення взаємодії з базою даних і підтримання її актуальною та цілісною.

У підсумку, в даній курсовій роботі було не лише висвітлено теоретичні концепції управління базами даних у фінансовому секторі, але й реалізувано їх у практичному проекті. Створена система є поглядом на сучасні виклики управління інформацією в банківській сфері та визначає шлях до подальшого розвитку та вдосконалення в цьому напрямі.

Перелік посилань

1. Date, C. J. (2004). An Introduction to Database Systems. Addison-Wesley.
2. Elmasri, R., & Navathe, S. B. (2019). Fundamentals of Database Systems. Pearson.
3. Connolly, T., & Begg, C. (2014). Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.
4. Ullman, J. D., & Widom, J. (2008). A First Course in Database Systems. Pearson.
5. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). Database System Concepts. McGraw-Hill.
6. MySQL Documentation. (<https://dev.mysql.com/doc/>)
7. Oracle Database Documentation. (<https://docs.oracle.com/en/database/>)
8. Microsoft SQL Server Documentation. (<https://docs.microsoft.com/en-us/sql/>)
9. Kimball, R., & Ross, M. (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley.
10. Banking Technology. (<https://www.bankingtech.com/>)
11. Journal of Banking and Finance. (<https://www.journals.elsevier.com/journal-of-banking-and-finance>)
12. International Journal of Database Management Systems (IJDMS). (<https://airccse.org/journal/ijdms/ijdms.html>)
13. IEEE Transactions on Knowledge and Data Engineering. (<https://www.computer.org/csdl/journal/tk>)
14. Financial Stability Board. (<https://www.fsb.org/>)
15. World Bank Data. (<https://data.worldbank.org/>)