МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

15.03 — КМР. 1939–“С” 2022.12.30. 20 ПЗ

Харченка Владислава Євгеновича

2023 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 004.9:678.53

«ПОГОДЖЕНО»

Декан факультету
інформаційних технологій
Глазунова О.Г., д.пед.н., професор
_____

« » _____2023 р

«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»

Завідувач кафедри комп'ютерних наук
Голуб Б.Л., к.тех.н., доцент
_____

_____2023 р

МАГІСТЕРСЬКА РОБОТА

На тему: Система аналізу цифрових платежів

Спеціальність 121 Інженерія програмного забезпечення
(шифр і назва).

Освітньо-професійна програма Програмне забезпечення інформаційних систем
(назва)

Робота на здобуття кваліфікації магістра

Керівник магістерської роботи
старший викладач / Ящук Д.Ю. /
(вчене звання і ступінь) (підпис) (ПІБ)

Виконав _____ / Харченко В.Є. /
(підпис) (ПІБ студента)

КИЇВ – 2023

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) Інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри_____
_____
_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____20_____року

З А В Д А Н Н Я
ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

_____
(прізвище, ім'я, по батькові)
Спеціальність _____
(код і назва)
Освітня програма _____
(назва)
Орієнтація освітньої програми_____
(освітньо-професійна або освітньо-наукова)
Тема магістерської кваліфікаційної роботи: Система аналізу цифрових платежів
_____
_____
затверджена наказом ректора НУБіП України від "____"_____20___р. №_____
Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)
Вихідні дані до магістерської кваліфікаційної роботи _____
_____
_____
_____
_____
_____
Перелік питань, що підлягають дослідженню:
1. _____
2. _____
3. _____

Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "_____"_____20___ р.

Керівник магістерської кваліфікаційної роботи _____ _____
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

Актуальність теми. Банківський сектор є невід'ємною частиною глобальної фінансової системи та відіграє важливу роль у сучасній економіці. Незмінний та зростаючий попит на фінансові послуги обумовлений наростаючим населенням, розвитком підприємництва та змінами в споживчих звичках. Забезпечення ефективної та надійної банківської системи має вирішальне значення для забезпечення стабільності та подальшого розвитку фінансового сектору.

Цифрова трансформація: Банки активно інвестують у розробку цифрових рішень, щоб відповідати сучасним потребам клієнтів і конкурувати з фінтех-компаніями. Дослідження цифрових інновацій та їх впливу на банківську систему мають вирішальне значення.

Кібербезпека: Запит на забезпечення кібербезпеки в банківському секторі надзвичайно високий, оскільки зростає кількість кіберзагроз. Захист клієнтських даних та фінансових операцій є невід'ємною частиною банківської діяльності та актуальною темою досліджень.

Інновації та автоматизація: Банки постійно шукають способи автоматизувати свої операції та знизити витрати, використовуючи роботизовані системи та штучний інтелект. Дослідження в галузі автоматизації процесів та впровадження нових технологій є надзвичайно важливими.

Регулювання та внутрішні вимоги: Зміни в законодавстві та регулятивних вимогах потребують постійного оновлення банківських систем та політики. Дослідження в області регулювання та внутрішніх стандартів є надзвичайно актуальними.

Відповідь на очікування клієнтів: Вимоги клієнтів до якості та доступності банківських послуг постійно зростають. Дослідження, спрямовані на задоволення цих очікувань, є ключовими для привертання та утримання клієнтів.

Глобалізація та міжнародні відносини: Банківські установи активно взаємодіють на міжнародному рівні, здійснюючи глобальні операції та надаючи послуги міжнародним клієнтам. Дослідження в галузі міжнародних фінансових відносин та міжнародного банкінгу є важливим завданням, оскільки вони впливають на економічну стійкість та міжнародний обмін.

Екологічні та соціальні вимоги: Сучасні банки дедалі більше приділяють увагу екологічним та соціальним питанням. Дослідження в галузі відповідального

банкінгу та сталого розвитку є актуальними для банківських установ, які бажають враховувати вплив своєї діяльності на довкілля та суспільство.

Глобальні кризи та фінансова стійкість: Історія світових фінансів свідчить про періодичні фінансові кризи. Дослідження та розробка стратегій для забезпечення фінансової стійкості та запобігання кризам є важливим завданням, особливо в умовах глобальних змін та невизначеності.

Інклюзивна фінансова система: Забезпечення доступу до фінансових послуг для всіх верств населення є важливою метою. Дослідження, спрямовані на розвиток інклюзивних банківських послуг, сприяють зменшенню фінансової нерівності та підтримці соціального розвитку.

Глобальна конкуренція: Банківська галузь стикається зі зростанням конкуренції як з боку традиційних банків, так і зі сторони нових учасників, таких як фінтех-стартапи. Дослідження та розробка стратегій конкурентної переваги є актуальними для банків, які прагнуть зберегти та розширити свою присутність на ринку.

Усі ці аспекти підтверджують важливість та актуальність дослідження банківської системи для забезпечення її стабільності та успішності в глобальному фінансовому середовищі.

Також ці пункти підкреслюють актуальність дослідження та розвитку банківської системи для забезпечення її стабільності та конкурентоздатності в сучасному фінансовому світі.

Мета роботи. Мета роботи полягає у вивчені потреб та запитів клієнтів банку шляхом розробки аналітичної системи яка що надасть менеджерам і аналітикам банку створювати більш привабливі пропозицій які сприятимуть конкурентноздатності банку.

Зміст поставлених завдань. Для досягнення поставленої мети необхідно:

- провести системний аналіз об’єкта;
- сформувати вимоги та структуру до системи аналізу онлайн банку;
- розробити структуру сховища даних;
- реалізувати систему аналізу, провести випробування із реальними даними.

Об'єкт і предмет дослідження. Об'єктом дослідження процес роботи банківської системи. Предметом дослідження – система аналізу для онлайн банкінгу.

Методи дослідження. При проведенні досліджень були використані наступні програмні продукти: Visual studio, Data Modeler, SQL Server, IntelliJ IDEA.

Наукова новизна. У процесі дослідження був розроблений аналітичний модуль який допомагає аналітикам та менеджерам банку створювати пропозиції для клієнтів які прогнозовано будуть успішними.

Апробація результатів дослідження. Виступ відбувся виступ на XIV Міжнародній науково- практичній конференції молодих вчених «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: ЕКОНОМІКА, ТЕХНІКА, ОСВІТА '2023», 26-27 жовтня 2023 року, НУБіП України, Київ, 2023.

Публікації. За темою магістерської були опубліковані тези «Система аналізу цифрових платежів відбувся виступ на XIV Міжнародній науково-практичній конференції молодих вчених «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: ЕКОНОМІКА, ТЕХНІКА, ОСВІТА '2023», 26-27 жовтня 2023 року, НУБіП України, Київ, 2023.

Структура роботи. Структурні елементи роботи розміщені в такій послідовності.

У ***першому розділі*** розглядається предметна область, сформульована постановка задачі дослідження та аналіз стану автоматизації сучасних тепличних господарств.

Другий розділ присвячено моделюванню системи. Були розроблені діаграми прецедентів використання, пакетів, класів та розгортання, які дозволяють отримати передумови для аналізу поведінки системи.

Третій розділ присвячено розробці та реалізації інформаційного забезпечення системи. Було побудовано логічну модель даних, реляційну БД та спроектовано сховище даних та створено розгорнутий куб системи.

У ***четвертому розділі*** розглянуто розробка програмного забезпечення системи. Було розроблені алгоритми роботи системи, програмне забезпечення аналітичного модулю онлайн банку.

У ***п'ятому розділі*** розглянуто охорону праці в банку.

1. АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

Предметна область - це сукупність взаємопов'язаних функцій та завдань управління, призначених для досягнення конкретних цілей. Вона представляє собою частину реального світу, яка викликає інтерес для проведення конкретного дослідження або є об'єктом певної діяльності.

Процес дослідження інформаційної системи – це аналіз предметної області, який базується на засобах і методах обстеження і орієнтований на формування вимог до системи, концептуалізацію і моделювання моделей з об’єктів для цієї області, а також на дослідження функцій і необхідних структур даних, зокрема типів і видів документів для їхнього виконання. Аналіз предметної області є багатозначним, в ньому ведеться міркування в межах тієї чи іншої стратегії діяльності предметної області.

Для представленнями системи на кожному етапі діяльності аналізуються поняття, об’єкти та будуються моделі на їхній основі, що відповідає обраному сценарію.

В процесі обстеження інформаційної системи відбувається концептуалізація, тобто перехід від природного представлення предметної галузі до її формальної моделі із заміною процесів предметної галузі на процеси системи. Процес аналізу предметної області в розробці інформаційних систем передбачає виділення основних і допоміжних бізнес-процесів, які необхідні для забезпечення виробництва продукту чи послуги. Але, поряд з цим, виділення і розгляд бізнес-процесів надає можливість визначитися з бізнес-елементами і структурами даних, які повинні брати участь в обробці даних. Такі можливості вимагають від розробника інформаційної системи в моделюванні бази даних відштовхуватися не тільки від документів, які використовуються в діяльності 5 предметної області, а й оточення кожного бізнес-процесу і функцій, що включає визначення бізнес-елементів, об'єктів даних, виконавців обробки, власників процесів і функцій,

попередніх і наступних функцій, що ініціюють і результуючих подій, інші елементи.

Проведення аналізу предметної області в інтересах подальшого проектування бази даних є завданням, що формує єдиний погляд на відомості, які в предметної області обробляються, враховуючи не тільки їх структури, а й правила зберігання і обробки, що відбивається в виділених функціях і завданнях.

1.1. Загальна характеристика онлайн банку

Банки в Україні утворюються відповідно до вимог законодавства, зокрема Законів України "Про банки і банківську діяльність" і "Про господарські товариства", а також у відповідності до нормативно-правових актів, що приймає Національний банк України.

Учасниками банку можуть бути юридичні та фізичні особи, незалежно від їхнього статусу резидента або нерезидента, а також можуть включати державні органи в особі Кабінету Міністрів України або їх повноважені органи.

Банки, створені в Україні, приймають організаційно-правові форми публічних акціонерних товариств або кооперативних банків. Вони можуть діяти як універсальні банки або спеціалізовані заклади.

Використання терміну "банк" і його похідних в назвах юридичних осіб дозволяється лише тим, які отримали реєстрацію в Національному банку України і мають відповідну банківську ліцензію.

Ще на самому початку розвитку банківської системи як незалежного учасника ринкових відносин, пов'язаних з обігом цінностей, інформація виявилася важливим аспектом функціонування банків. Навіть у середньовіччі, коли торговцям загрожували ризики розбійницьких нападів та грабежів, з'явилася ідея перевозити із собою не самі цінності або гроші, а важливу інформацію.

На той час, ця "банківська інформація" приймала форму розписок та доручень, які передавались від одного банкіра іншому, навіть коли вони знаходилися в різних містах чи навіть країнах. Ці документи містили певні фінансові зобов'язання, проте для зловмисників вони не мали жодної цінності, оскільки доступ до коштів мали лише конкретні особи. Отже, ще на ранніх стадіях розвитку банківської системи, інформаційні ресурси стали критично важливими для функціонування банків, і саме цей інформаційний фактор сприяв поширенню банківської системи в цілому.

Сучасна електронна банківська система повинна мати найвищий рівень захисту від зламу та хакерських атак, оскільки в ній міститься важлива інформація про користувачів, їхні рахунки та паролі. Крім того, така система повинна розгортатися на потужних платформах, щоб гарантувати багатозадачність і надійну продуктивність.

Забезпечення постійної доступності електроенергії є надзвичайно важливим для забезпечення безперебійної роботи системи, особливо оскільки система взаємодіє з платежами та транзакціями. Для захисту системних дій в такій системі використовується принцип ACID (Atomicity, Consistency, Isolation, Durability) - це набір властивостей, які гарантують надійну роботу транзакцій у базі даних. Ці властивості включають атомарність, узгодженість, ізольованість та довговічність, і вони дозволяють розглядати послідовність операцій у базі даних як одну логічну операцію, що називається транзакцією.

Atomicity — Атомарність У транзакціях часто включаються різні операції. Атомарність (Atomicity) гарантує, що жодна транзакція не буде виконана частково. Всі операції, що взяли участь у транзакції, виконуються повністю або ж не виконуються зовсім. Навіть якщо під час виконання однієї з операцій виникне помилка і операцію доведеться скасувати, то всі інші зміни, внесені в рамках транзакції, також скасовуються. Важливо, щоб система завжди дотримувалася атомарності, навіть при виникненні таких ситуацій, як втрата електроенергії,

помилки чи збої. Гарантування атомарності запобігає неповному або частковому оновленню бази даних, що може створити більше проблем, ніж вирішити.

Прикладом атомарної транзакції є операція переказу коштів з одного рахунку на інший, яка включає дві операції: зняття коштів з одного рахунку і їхнє зарахування на інший. Виконання цих операцій в рамках атомарної транзакції забезпечує узгодженість даних у базі, означаючи, що кошти не будуть віднесені чи зараховані, якщо одна з операцій не вдасться.

Consistency — Узгодженість Згідно з вимогою узгодженості (Consistency), система повинна знаходитися в узгодженому, непротирічивому стані перед початком транзакції і після її завершення. Під час виконання транзакції система може перебувати в тимчасово неузгодженому стані, але ця неузгодженість не повинна бути видимою за межами транзакції завдяки іншим аспектам, таким як атомарність та ізоляція.

Узгодженість гарантує стабільність бази даних, забезпечуючи відповідність даних усім визначеним правилам та обмеженням, включаючи ключові обмеження та посилальну цілісність. Це допомагає запобігти пошкодженню бази даних через некоректну транзакцію, хоча воно не гарантує правильність самої транзакції. Посилальна цілісність забезпечує відповідність унікального ключа зовнішньому ключу.

Isolation — Ізоляція Ізоляція (Isolation) означає, що жодні проміжні зміни не будуть видимі за межами транзакції до її завершення. Питання ізоляції стає актуальним, коли багато транзакцій одночасно взаємодіють із тими самими даними. За цією вимогою, якщо дві транзакції спробують змінити одні й ті самі дані, одну з них буде відхилено або призупинено до завершення іншої.

Durability — Довговічність Довговічність (Durability) гарантує, що незалежно від інших негараздів, після відновлення працездатності системи результати завершених транзакцій залишаються незмінними. Іншими словами, якщо користувач отримав підтвердження про успішне завершення транзакції, він може бути впевнений, що дані будуть збережені і відновлені в разі будь-яких помилок чи збоїв.

1.2. Постановка задач дослідження

Мету цього завдання буде досягнуто шляхом створення системи, яка надасть підтримку адміністрації тепличного комплексу в процесі ухвалення рішень.

Сучасні системи підтримки прийняття рішень (СППР) є результатом об'єднання керівницьких інформаційних систем і систем управління базами даних. Вони спеціалізуються на розв'язанні завдань повсякденного керівництва і служать інструментом для тих, хто повинен здійснити вибір. СППР дозволяють вирішувати завдання з вибору рішень в сфері неструктурованих і слабоструктурованих задач, включаючи ті, які мають багато критеріїв.

Основним завданням СППР є надання аналітикам інструментів для аналізу даних. Аналіз даних може бути поділений на такі типи, залежно від обсягу обробки даних:

Інформаційно-пошуковий, де СППР виконує пошук необхідних даних, відповідаючи заданим запитам.

Оперативно-аналітичний, де СППР групує та узагальнює дані для аналітика.

Інтелектуальний, де СППР виявляє функціональні та логічні закономірності у наявних даних, створює моделі та правила, які пояснюють ці закономірності і здатні прогнозувати розвиток процесів.

Загальна архітектура СППР може бути подана на схемі

Функціонально, СППР включає наступні компоненти: сервер сховища даних, інструменти OLAP та інструменти Data Mining.

У підсистемі введення даних OLTP (Online Transaction Processing) забезпечує операційну обробку даних, використовуючи звичайні системи управління базами даних (СУБД).

Загальна структура системи підтримки прийняття рішень відображає архітектуру СППР.

В підсистемі зберігання інформації використовують сучасні СУБД та концепцію сховища даних. Сховище даних передбачає структуру для оперативної обробки даних та аналітичних запитів.

Підсистема аналізу може включати наступне:

Підсистему інформаційно-пошукового аналізу на основі реляційних СУБД і статичних запитів, що використовують мову SQL (Structured Query Language).

Підсистему оперативного аналізу, де використовується технологія OLAP (Online Analytical Processing) для обробки багатовимірних даних.

Підсистему інтелектуального аналізу, яка використовує методи та алгоритми Data Mining для отримання даних.

Ці компоненти СППР розглядають питання щодо накопичення та моделювання даних на концептуальному рівні, ефективного завантаження даних з різних джерел та аналізу даних. Важливо зауважити, що сучасні системи оперативної аналітичної обробки обмежені доступом до багатовимірних даних.

Отже, ставлячи задачі з розробки аналіттичного модулю можна виділити наступну послідовність.

Спершу я визначив мету і об'єктиви впровадження цифрової системи аналізу в онлайн-банку. Моя команда та я чітко зрозуміли, що ми хочемо досягти, і визначили завдання для розвитку системи.

Після цього ми обрали відповідну платформу та інструменти для аналізу даних, які відповідають нашим потребам. Ми також налагодили процеси збору та зберігання даних, включаючи інтеграцію з банківською системою для отримання необхідної інформації.

Далі ми приступили до аналізу та обробки даних, використовуючи обрані інструменти. Ми знайшли корисну інформацію, виявили патерни та тренди, а також розробили систему ідентифікації аномалій.

Для візуалізації результатів ми використали відповідні інструменти, щоб представити дані у зрозумілій формі для наших користувачів.

Надалі ми розробили та впровадили модулі системи аналізу в роботу онлайн-банку. Ми також забезпечили належне навчання співробітників та користувачів, щоб вони могли використовувати нові аналітичні інструменти.

Наразі ми постійно моніторимо роботу системи та надаємо підтримку для наших користувачів. Ми також прагнемо постійно вдосконалювати систему на основі зворотного зв'язку та змін потреб користувачів.

Забезпечення безпеки та конфіденційності даних завжди залишається нашим пріоритетом, і ми ретельно враховуємо це в усіх аспектах роботи системи.

Завдяки виконаним крокам, ми досягли успішного впровадження цифрової системи аналізу в нашому онлайн-банку, поліпшили обслуговування клієнтів і забезпечили конкурентні переваги на ринку.

1.3 Аналіз стану автоматизації онлайн банку

Аналіз стану автоматизації онлайн-банку може бути важливим для оцінки ефективності і функціональності банківської системи. Ось деякі аспекти, які можуть бути включені в такий аналіз:

Зручність інтерфейсу користувача: Оцінка того, наскільки зручно користувачам взаємодіяти з онлайн-банком через веб-сайт або мобільний додаток. Важливо, щоб інтерфейс був інтуїтивно зрозумілим та дружнім до користувача.

Функціональність: Аналіз доступних функцій та можливостей. Це може включати можливість переказу коштів, оплати рахунків, перевірки балансу, створення депозитів і т. д.

Безпека: Оцінка рівня безпеки онлайн-банкінгу, включаючи захист від шахраїв, автентифікацію користувача, захист від шкідливих програм і зберігання конфіденційної інформації.

Час роботи та доступність: Аналіз того, наскільки онлайн-банк доступний для користувачів в будь-який час, а також часу, який займає виконання операцій.

Інтеграція з іншими сервісами: Перевірка можливості інтеграції з іншими банківськими послугами та зовнішніми платіжними системами.

Автоматизація обробки операцій: Оцінка ступеня автоматизації обробки операцій, включаючи обробку платежів та переказів, а також забезпечення швидкого і точного обліку та звітності.

Підтримка клієнтів: Оцінка доступності та ефективності служби підтримки для вирішення питань і проблем користувачів.

Аналітика та звітність: Перевірка можливості отримання звітів та аналітики щодо фінансових операцій та стану рахунків.

Оновлення і розвиток: Оцінка того, наскільки активно банк розвиває та оновлює свою онлайн-платформу для вдосконалення функціональності та безпеки.

Вартість обслуговування: Аналіз витрат на обслуговування та можливостей зменшення витрат для банку і його клієнтів.

Аналіз стану автоматизації онлайн-банку може допомогти банку вдосконалити свою платформу та надати клієнтам кращий досвід користування послугами.

Досліджуваний онлайн-банк високо автоматизований. Він надає клієнтам зручний інтерфейс для операцій, таких як оплата рахунків та перекази грошей, має системи автентифікації для безпечного входу та автоматично обробляє та зберігає дані користувачів. Клієнти отримують автоматичні звіти і можуть керувати кредитами та операціями з боргами. Крім того, банк надає онлайн-консультацію і підтримку для користувачів.

Отже можна зробити висновок що ми не маємо ніяких перешкод щоб почати розробку аналітичного модулю через те що рівень автомазації і цифровізації даних на підприємстві є високою.

2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1. Теоретичні відомості моделювання системи

Під час розробки автоматизованих систем управління та систем прийняття рішень на етапах кодування і тестування виявляється значна кількість недоліків, виправлення яких призводить до суттєвих змін в загальній системі розробки. Ці помилки ураховуються під час моделювання та докладного аналізу створюваних проектів. Моделювання дає можливість "передбачити" проект у процесі його розробки та створити умови для аналізу поведінки системи в залежності від початкових умов.

Для ефективного керування процесами в модельованій системі управління, необхідно створити структуру, тобто організувати процеси. Особливо важливе моделювання роботи інформаційної системи на початкових етапах її створення, оскільки виправлення помилок, допущених на цьому етапі, є дорогим заходом. Тому користь аналізу задачі та створення логічної моделі рішення очевидна.

З цієї причини необхідно детально вивчити і розібратися в області, пов'язаній з роботою банківської системи. Це включає в себе ознайомлення з термінологією цієї галузі, збір нормативних та правових документів, а також вивчення прикладів документів цього підприємства та їхнє переміщення, як внутрішньо, так і поза межами підприємства.

Наступним кроком у розробці є етап проектування. Перед розпочатком проектування та впровадження в життя важливо мати точне та докладне розуміння вимог на високому рівні. Крім того, корисно мати структуру вимог, яка може бути використана як вихідні дані для розробки системи. Усе це досягається через аналіз та моделювання.

Під час роботи на етапах моделювання і проектування необхідно отримати проект системи, який містить достатньо інформації для її реалізації. Також важливо провести аналіз роботи банку для визначення ступеня завантаженості кожного

відділу та визначення, що потрібно автоматизувати в першу чергу та як це можна зробити.

Основними цілями моделювання при розробці проектів є:

Представлення діяльності підприємства та використання технологій у вигляді ієрархії діаграм, які забезпечують наочність та повноту їх відображення.

Формування рекомендацій щодо перебудови організаційно-управлінської структури на основі аналізу пропозицій.

Оптимізація інформаційних потоків в межах підприємства, включаючи документообіг.

Розробка рекомендацій для раціональних технологій роботи відділів підприємства та його взаємодії з зовнішнім світом.

Аналіз вимог і розробка специфікацій корпоративних інформаційних систем.

2.2. Діаграма прецедентів

Діаграма прецедентів (Use Case Diagram) є одним із видів діаграм UML (Unified Modeling Language) і використовується для моделювання функціональності системи з точки зору її користувачів або акторів. Діаграма прецедентів допомагає визначити, як система взаємодіє з різними користувачами або зовнішніми сутностями, а також які функції або можливості надаються цим користувачам.

Основні компоненти діаграми прецедентів включають:

Прецеденти (Use Cases): Прецеденти представляють окремі функціональні можливості або дії, які система може виконувати. Кожен прецедент зазвичай описується назвою і може бути підписаний коротким описом. Наприклад, в онлайн-магазині прецедентами можуть бути "Оформлення замовлення", "Пошук товарів", "Оплата" і т. д.

Актори (Actors): Актори представляють ролі або сутності, які взаємодіють з системою. Актори можуть бути користувачами, іншими системами або будь-якими

зовнішніми сутностями. Кожен актор представляється у вигляді піктограми або назви і має відношення до одного або декількох прецедентів.

Зв'язки (Associations): Зв'язки відображають взаємодію між акторами та прецедентами. Зв'язки вказують, які прецеденти доступні для конкретного актора. Зазвичай зв'язки позначаються стрілками, що вказують на актора, який користується прецедентом.

Включення та розширення (Inclusion and Extension): Включення (inclusion) і розширення (extension) - це спеціальні відношення між прецедентами, які вказують на включення певних дій у інші прецеденти або розширення їх функціональності. Ці відношення допомагають уточнити логіку взаємодії між прецедентами.

Діаграма прецедентів є потужним інструментом для з'ясування вимог до системи і визначення її функціональності з орієнтацією на користувача. Вона допомагає команді розробників та зацікавленим сторонам краще розуміти, як користувачі будуть взаємодіяти з системою і як вона повинна виконувати свої завдання.

Однією з основних причин краху програмних проектів є недостатньо зрозумілий зв’язок між основними його учасниками або повна відсутність контакті в між ними. Вкрай необхідний такий підхід до розробки системи, який дозволив би зрозуміти цілі її функціонування розробникам і системи, при цьому, не відволікаючи штатних співробітників від їхньої основної роботи. Інструментом рішення такої задачі є діаграми прецедентів використання. Вони відображають елементи системи, що взаємодіють один з одним.

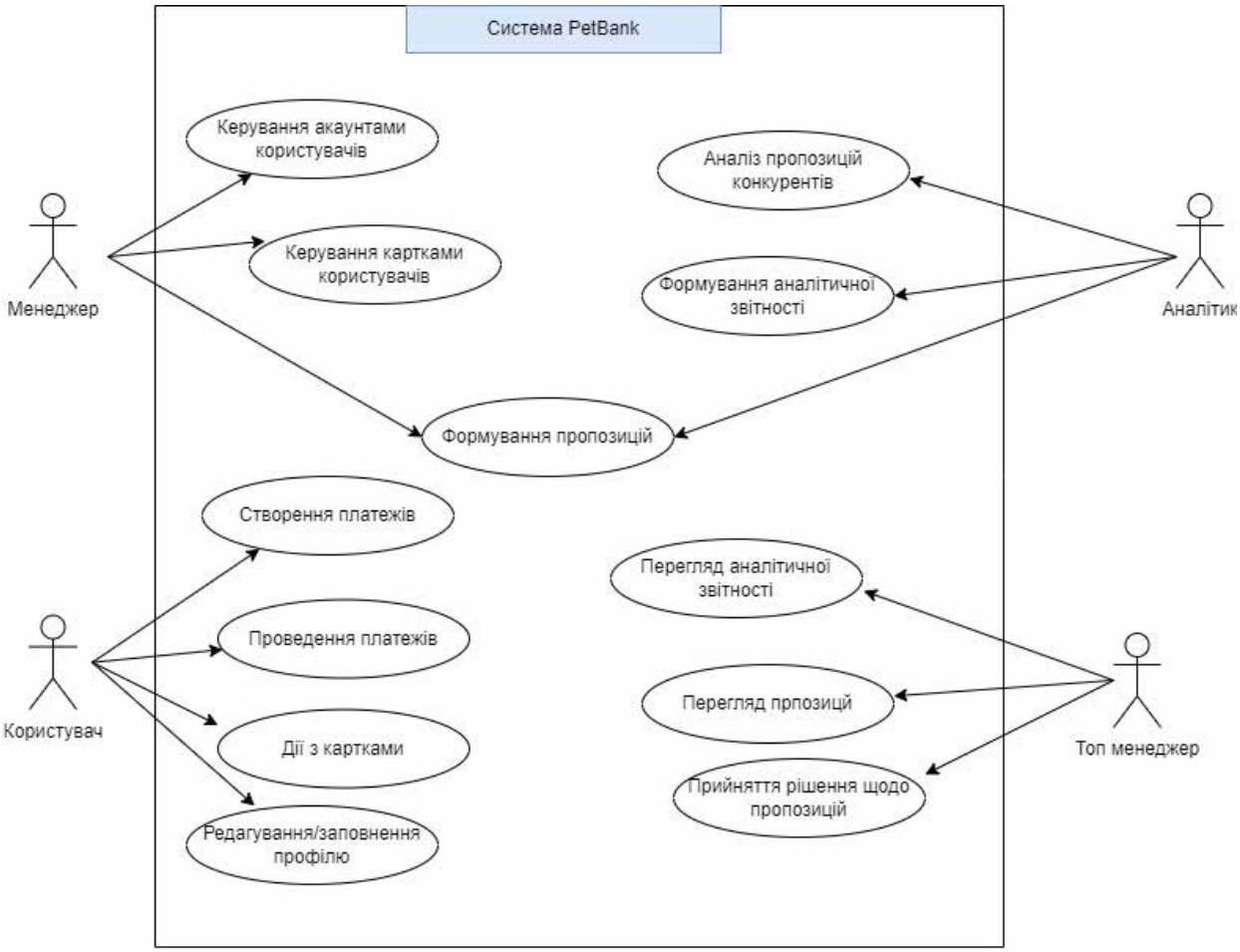


Рис. 2.1. Діаграма прецедентів аналітичної системи онлайн банкінгу

Діаграма прецедентів визначає ключові дії, які можуть бути виконані у нашій системі та ролі, які відповідають за виконання цих дій. В даному контексті система орієнтована на обробку платежів та аналіз даних.

У цій системі:

Менеджер: Менеджер в системі відповідає за керування фкаунтами користувачів та керування картками користувачів.

Аналітик: Аналітик займається формуванням аналітичної звітності та аналізом позицій конкурентів.

Користувач: Користувач виконує дії, такі як створення платежів, проведення платежів та інші дії з картками.

Топ-менеджер: Топ-менеджер в системі відповідає за перегляд аналітичної звітності, прийняття рішень та перегляд пропозицій.

Прецеденти (Use Cases):

Формування аналітичної звітності (Аналітик): Аналітик може створювати аналітичну звітність на основі доступних даних.

Аналіз позицій конкурентів (Аналітик): Аналітик аналізує позиції конкурентів на ринку.

Керування фкаунтами користувачів (Менеджер): Менеджер може керувати аккаунтами користувачів, включаючи створення, оновлення та видалення.

Керування картками користувачів (Менеджер): Менеджер має можливість керувати картками користувачів, виконуючи дії з ними.

Створення платежів (Користувач): Користувач може створювати нові платежі в системі.

Проведення платежів (Користувач): Користувач може проводити оплату засновану на створених платежах.

Дії з картками (Користувач): Користувач може виконувати різні дії з картками, які пов'язані з їхнім використанням.

Перегляд аналітичної звітності (Топ-менеджер): Топ-менеджер має доступ до аналітичної звітності, яку можна переглядати в системі.

Прийняття рішень (Топ-менеджер): Топ-менеджер може приймати рішення на основі інформації, що міститься в аналітичній звітності.

Перегляд пропозицій (Топ-менеджер): Топ-менеджер може переглядати пропозиції запропоновані аналітиком

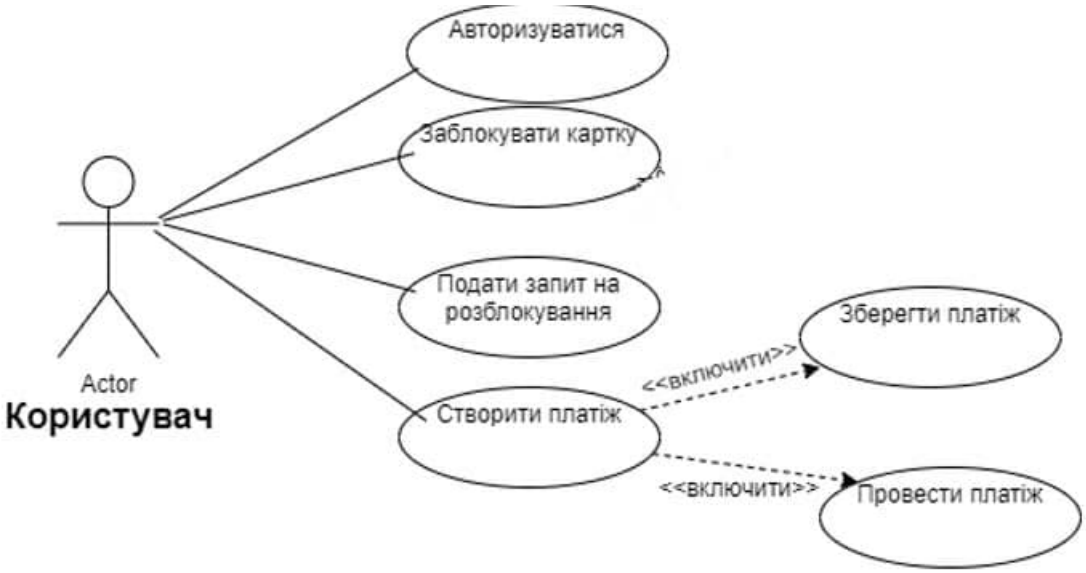


Рис. 2.2. Діаграма прецедентів користувача онлайн банкінгу

Ця діаграма прецедентів описує дії користувача (актора) в системі. Основні прецеденти включають:

Авторизуватися: Користувач може увійти до системи, вказавши свої облікові дані.

Заблокувати карту: Користувач має можливість заблокувати свою картку, наприклад, в разі втрати або крадіжки.

Подати запит на розблокування: Якщо картка користувача заблокована, він може подати запит на її розблокування.

Створити платіж: Користувач може створювати нові платежі, вказуючи необхідну інформацію про отримувача, суму тощо.

Зберегти платіж: Після створення платежу, користувач може зберегти його, можливо, для подальшого підтвердження або редагування.

Провести платіж: Коли платіж готовий, користувач може провести його та здійснити оплату.

Ця діаграма прецедентів показує, як користувач може взаємодіяти з системою для управління своєю карткою та платежами.

2.3. Діаграма пакетів

Діаграми пакетів в рамках UML відображають взаємозалежності між пакетами, що сприяє створенню моделі.

Пакет - це ключовий спосіб організації елементів у моделі UML. Кожен пакет включає всі свої елементи, і ці елементи є власністю цього пакету. Ми можемо говорити, що відповідні елементи належать до пакету або входять до нього. Зазначено, що кожен елемент може бути призначений лише одному пакету. В свою чергу, пакети можуть вкладатися один в одного, і перші називаються підпакетами. Всі елементи підпакету, таким чином, належать більш загальному пакету. Ця взаємодія між пакетами встановлює ієрархію в моделі, визначаючи відношення вкладеності між елементами моделі.

Пакети є незамінним інструментом для організації великих проектів.

Застосування діаграм пакетів:

- Організація та структурування складних систем, особливо великих програмних проектів.
- Показ взаємозалежностей між компонентами та пакетами.
- Визначення ієрархічних структур та інші структурні аспекти великих систем.

Діаграми пакетів допомагають розробникам та аналітикам отримати краще розуміння структури системи, спростити управління проектами і визначити взаємозв'язки між її складовими частинами.

Для нашої системи пакет Система тепличний комплекс містить 3 пакети:

- Контроль, управління та аналіз
- Управління підрозділу аналітики Контроль параметрами транзакцій

При цьому пакет Контроль, управління та аналіз залежить від двох інших.

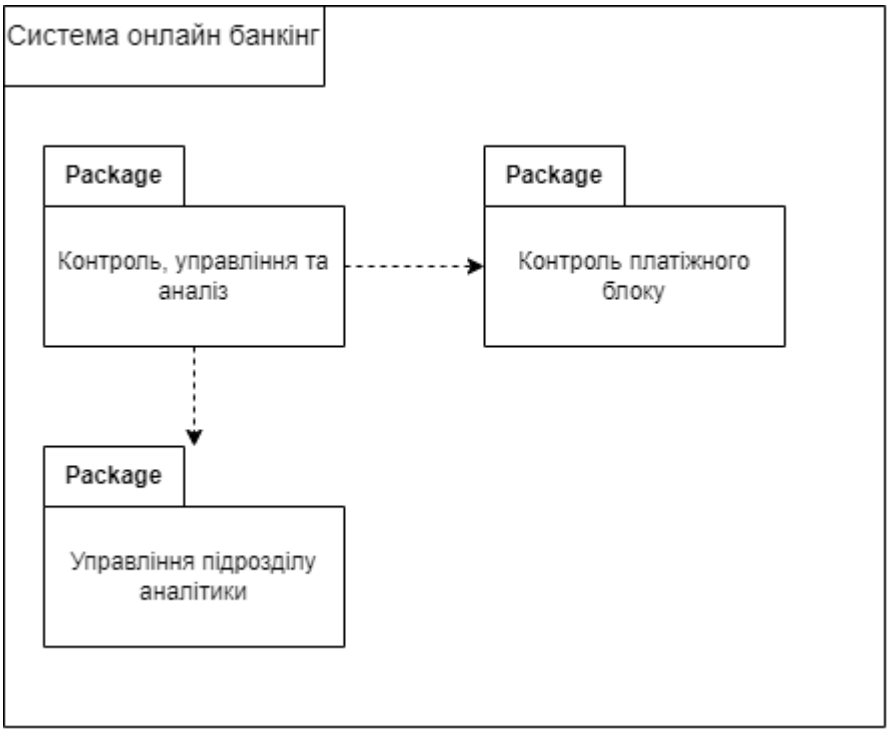


Рис. 2.3. Діаграма пакетів

2.4. Діаграма класів

Діаграма класів (Class Diagram) є однією з основних та найпоширеніших діаграм в рамках мови моделювання UML (Unified Modeling Language). Вона використовується для візуалізації структури системи та відображення класів, їх атрибутів, методів та взаємозв'язків між класами. Діаграма класів допомагає розробникам та аналітикам отримати чітке уявлення про структуру програмного проекту або системи.

Основні елементи діаграми класів включають:

Класи (Classes): Кожен клас відображає об'єкт або сутність, яка має атрибути (змінні) і методи (функції). Класи зображуються у вигляді прямокутників з ім'ям класу всередині.

Атрибути (Attributes): Атрибути вказують властивості класу та представляють змінні, які йому належать. Вони зображуються під іменем класу та включають ім'я атрибута та його тип даних.

Методи (Methods): Методи вказують операції або функції, які можна виконати над об'єктами класу. Вони зображуються під ім'ям класу та включають ім'я методу, параметри та тип повернення.

Взаємозв'язки (Associations): Взаємозв'язки показують, як класи спільно взаємодіють один з одним. Вони можуть мати різні типи взаємозв'язків, такі як асоціація, агрегація, композиція тощо.

Класифікація (Inheritance): Діаграма класів відображає ієрархію класів, де один клас може успадковувати властивості та методи від іншого класу. Це відображається за допомогою стрілок, які вказують на батьківський клас.

Залежності (Dependencies): Залежності показують, як класи взаємодіють один з одним і можуть бути використані для позначення, як один клас використовує функціональність іншого класу.

Інтерфейси (Interfaces): Діаграма класів може також включати інтерфейси, які вказують обов'язкові методи для класів, що реалізують цей інтерфейс.

Діаграми класів використовуються для аналізу, проектування, документування та комунікації структури системи, і вони є важливим інструментом при розробці програмного забезпечення.

Діаграма класів для таблиць "Card," "Account," "User," і "Payment" може виглядати так:

Card (Карта): Цей клас представляє таблицю "Card" і містить атрибути, які відображають інформацію про банківські карти, такі як "номер карти," "тип карти," "термін дії," і т.д.

Account (Рахунок): Цей клас представляє таблицю "Account" і містить атрибути, що описують банківські рахунки, такі як "номер рахунку," "баланс," "власник рахунку," і інші.

User (Користувач): Цей клас відображає таблицю "User" і має атрибути, що описують дані користувачів, такі як "ім'я," "прізвище," "електронна пошта," тощо.

Payment (Платіж): Цей клас відображає таблицю "Payment" і включає атрибути, які описують дані про платежі, включаючи "сума," "дата," "отримувач платежу," і інші параметри.

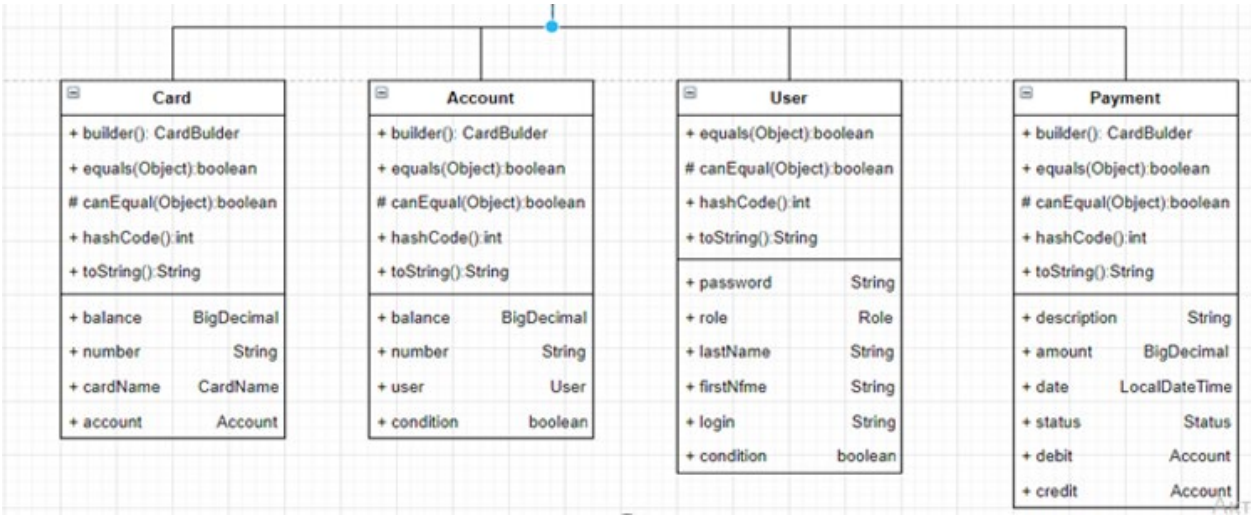


Рис.2.4. Діаграма класів

2.5. Топологія системи

Топологія системи - це структурний аспект системи, який визначає організацію її компонентів та взаємозв'язки між ними. Це описує спосіб, як умовні "вузли" або "елементи" системи пов'язані між собою та як вони обмінюються інформацією чи ресурсами.

Існують різні типи топологій систем, які можуть використовуватися в різних контекстах:

Зіркова топологія (Star Topology): Всі елементи системи підключені до центрального вузла. Це забезпечує простий спосіб управління мережею та можливість зосереджувати контроль, але якщо центральний вузол вийде з ладу, то всі інші вузли втратять з'єднання.

Кільцева топологія (Ring Topology): Кожен вузол підключений до двох сусідів, утворюючи кільце. Дана топологія надійна, але відсутність з'єднання в одному вузлі може призвести до розриву мережі.

Деревоподібна топологія (Tree Topology): Кілька груп зіркових чи інших підмереж об'єднуються в єдину мережу, створюючи ієрархічну структуру. Ця топологія може обслуговувати велику кількість вузлів, але вразлива до відмов у верхніх рівнях.

Мешковина топологія (Mesh Topology): Усі вузли пов'язані з кожним іншим вузлом в мережі. Ця топологія дуже надійна, оскільки може працювати, навіть якщо деякі з'єднання втратяться, але вона вимагає великої кількості кабелів та ресурсів для підтримки.

Сітка (Mesh Topology): Ця топологія є розширенням мешковини, але в цьому випадку не всі вузли з'єднані з усіма, а лише з окремими вузлами, що зменшує кількість з'єднань, не втрачаючи при цьому високу надійність.

Безпровідна топологія (Wireless Topology): Всі вузли безпровідно пов'язані між собою, використовуючи радіохвилі або інші безпровідні технології. Це дозволяє легко розширювати мережу, але може бути вразливою до перешкод та безпекових загроз.

Топологія системи визначається в залежності від конкретних потреб проекту та ресурсів, які доступні для побудови мережі чи системи. Коректний вибір топології допомагає забезпечити оптимальну продуктивність, надійність та легкість управління системою.

Користувач може входити в систему як з комп'ютера, так і з мобільного пристрою. Він має доступ до веб-інтерфейсу, який дозволяє виконувати різні дії, такі як створення платежів та перегляд даних. Веб-інтерфейс взаємодіє з користувачем та обробляє всі його запити.

База даних є центральним елементом системи та зберігає всі дані, необхідні для обробки платежів та аналізу даних. Вона взаємодіє з веб-інтерфейсом та аналітичним модулем для отримання та зберігання інформації.

Аналітичний модуль використовує дані зі сховища даних для проведення аналізу та допомагає користувачеві приймати рішення на основі накопичених даних. Він взаємодіє зі сховищем даних для отримання доступу до необхідних інформації.

Сховище даних використовується для зберігання великого обсягу даних, які використовуються для аналізу та прийняття рішень. Воно дозволяє аналітичному модулю отримувати доступ до даних та проводити різноманітний аналіз. Система в цілому допомагає користувачеві ефективно здійснювати платежі та аналізувати дані в онлайн режимі.

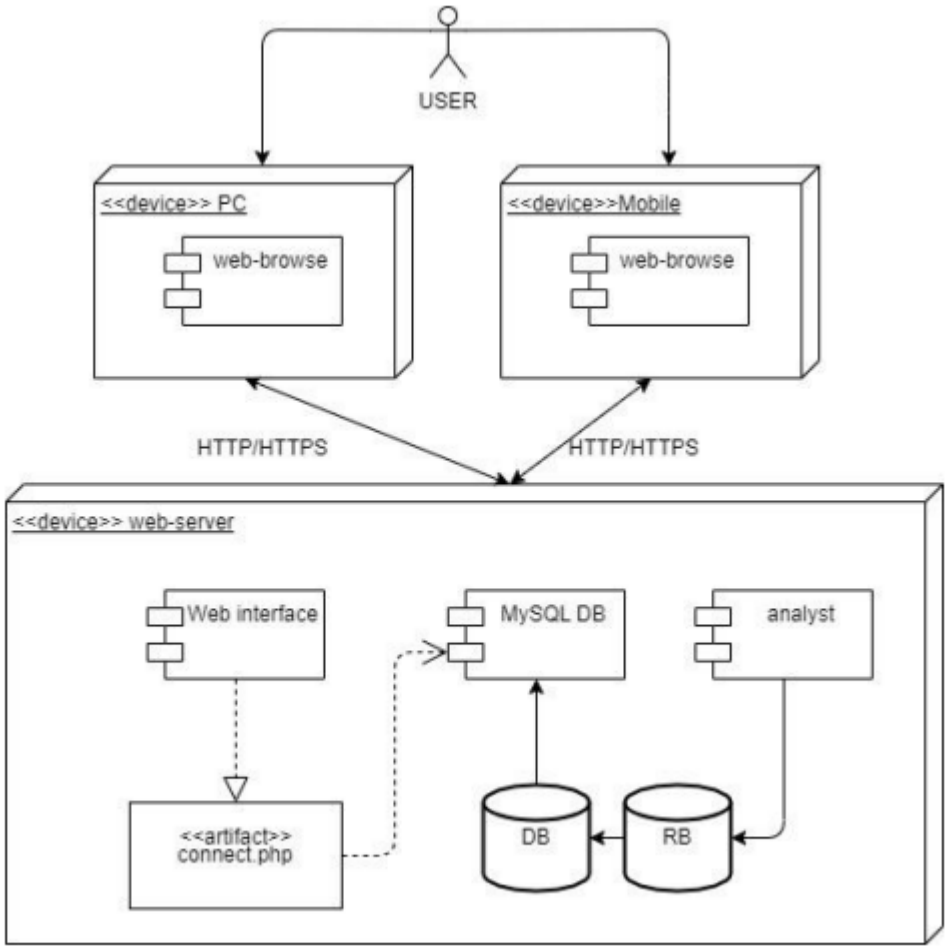


Рис. 2.5. Топологія системи

3. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1. Основні положення моделі інформаційного забезпечення

Модель інформаційного забезпечення - це концептуальна структура, яка визначає, як інформація обробляється, зберігається, передається та використовується в рамках системи або організації. Основні положення моделі інформаційного забезпечення (ІЗ) включають такі аспекти:

Структура інформації: Модель ІЗ визначає структуру інформації, включаючи типи даних, формати, поля, таблиці, бази даних та інші засоби для організації інформації.

Зберігання інформації: Модель ІЗ розглядає, де і як зберігається інформація, включаючи фізичні пристрої, сервери, облікові записи та методи забезпечення безпеки даних.

Обробка інформації: Модель ІЗ визначає процеси та процедури для обробки інформації, включаючи взаємодію користувачів, програм, систем та інших компонентів для опрацювання даних.

Передача інформації: Модель ІЗ включає в себе механізми та протоколи для передачі інформації між різними частинами системи, а також зовнішніми системами та користувачами.

Забезпечення доступу і безпеки: Модель ІЗ враховує, як забезпечується доступ до інформації, а також методи та процедури для захисту даних від несанкціонованого доступу та збереження конфіденційності, цілісності та доступності.

Архітектура системи: Модель ІЗ визначає архітектурну структуру системи, включаючи логічну та фізичну організацію компонентів, серверів, мереж та інших інфраструктурних аспектів.

Моніторинг та управління: Модель ІЗ може включати системи моніторингу та управління, які дозволяють відстежувати стан інформації, а також вживати заходів для забезпечення її належного функціонування.

Стандарти та рекомендації: Модель ІЗ може враховувати відповідні стандарти та рекомендації, які визначають норми і практики у галузі інформаційної безпеки та управління інформацією.

Загальна мета моделі інформаційного забезпечення - це забезпечити належну організацію, безпеку та ефективну роботу інформації в межах системи або організації. Модель ІЗ є важливим інструментом для аналізу, проектування та управління інформаційними ресурсами.

Однією з основних частин інформаційного забезпечення є інформаційна база.

Інформаційна база - це структурована колекція даних і інформації, яка використовується для підтримки функціонування системи, організації або процесу. Ця база даних може включати в себе різні види інформації, такі як текст, числа, графічні дані, аудіо- та відеозаписи, документи тощо. Інформаційна база дозволяє зберігати, організовувати та отримувати доступ до цієї інформації для різних цілей.

Інформаційні бази використовуються в різних сферах, включаючи бізнес, науку, освіту, медицину, військовий сектор та багато інших. Вони є важливим компонентом інформаційних систем та допомагають організаціям та процесам ефективно керувати та використовувати інформацію для досягнення своїх цілей.

3.2. Побудова логічної моделі

Логічна модель в контексті бази даних - це абстрактна репрезентація структури даних і їх взаємозв'язків в інформаційній системі або програмному продукті без прив'язки до конкретного середовища бази даних або сховища даних. Логічна модель описує сутності (такі як таблиці чи сутності-зв'язки), їх атрибути і взаємозв'язки між ними. Вона не визначає, як саме дані будуть зберігатися, а замість цього фокусується на тому, що ці дані означають і як вони пов'язані між собою.

Для системи аналітики онлайн-банкінгу, логічна модель може виглядати наступним чином:

Користувачі (Users): Ця сутність представляє клієнтів онлайн-банку та містить атрибути, такі як ID користувача, ім'я, прізвище, адреса електронної пошти і інші особисті дані.

Банківські рахунки (Accounts): Ця сутність відображає інформацію про банківські рахунки користувачів. Вона може містити атрибути, які включають номер рахунку, баланс, тип рахунку та інші.

Банківські карти (Cards): Сутність "Cards" описує дані про банківські карти, такі як номер карти, термін дії, власник карти і інші атрибути.

Платежі (Payments): Ця сутність включає інформацію про платежі, включаючи суму, дату, відправника і отримувача платежу, а також статус операції.

Логічна модель також повинна відображати взаємозв'язки між цими сутностями. Наприклад, вона може показати, що кожен користувач може мати багато банківських рахунків та банківських карт, а кожен платіж пов'язаний з конкретним користувачем, який здійснив цей платіж. Також важливо відобразити можливість аналітики і відслідковування платежів через цю модель.

Логічна модель служить як основа для подальшого проектування фізичної моделі бази даних, де вже враховуються деталі та технічні аспекти зберігання даних.

3.3. Основні положення фізичної моделі даних

У сучасному світі майже неможливо уявити комп'ютерну техніку та мережі, які не використовують бази даних та системи управління ними. Бази даних є необхідною складовою прикладного програмного забезпечення, і ми з ними зіштовхуємося в різних аспектах нашого життя.База даних - це організована колекція даних, яка зберігається та управляється комп'ютерною системою. Бази даних використовуються для ефективного зберігання, організації, оновлення та доступу до великих обсягів інформації в різних галузях, включаючи бізнес, науку, освіту та інші. Вони забезпечують структурованість даних, цілісність, можливість виконання запитів, многокористувацький доступ, автоматизацію процесів, захист даних та можливість резервного копіювання і відновлення інформації. Тип бази

даних вибирається відповідно до конкретних потреб та вимог проекту чи організації.

Останнім часом найбільше розповсюдження отримали реляційні бази даних. Реляційна база даних - це тип бази даних, яка використовує реляційну модель даних для зберігання та організації інформації. Цей тип бази даних був розроблений Едгаром Коддом у 1970-х роках і є одним з найпоширеніших та надійних способів зберігання даних в сучасних інформаційних системах.

Основні характеристики реляційних баз даних включають:

Таблиці (реляції): Дані організовані у вигляді таблиць, які мають назви та структуру. Кожна таблиця представляє собою колекцію записів, де кожен запис відображає конкретний об'єкт чи сутність.

Структура стовпців (атрибутів): Кожна таблиця має стовпці, які визначають типи даних та характеристики інформації, яка зберігається в таблиці.

Ключі: У реляційних таблицях визначаються ключі, такі як первинні ключі (Primary Keys) і зовнішні ключі (Foreign Keys), що дозволяють встановлювати зв'язки між таблицями.

Мова запитів SQL: Для роботи з реляційними базами даних використовується структурована мова запитів SQL (Structured Query Language), яка дозволяє виконувати різноманітні операції над даними, включаючи вставку, оновлення, вибірку і видалення.

Цілісність даних: Реляційні бази даних дотримуються принципів цілісності даних, гарантуючи, що дані є консистентними та відповідають визначеним правилам.

Запити та звіти: Користувачі можуть виконувати складні запити для отримання конкретних даних та генерувати звіти на основі інформації, що зберігається в базі даних.

Можливість резервного копіювання та відновлення: Для запобігання втраті даних реляційні бази даних підтримують процедури резервного копіювання та відновлення інформації.

Реляційні бази даних використовуються в широкому спектрі галузей, включаючи бізнес, науку, освіту, медицину та інші. Вони дозволяють ефективно зберігати, організовувати та обробляти дані, що робить їх важливим інструментом в сучасних інформаційних системах.

3.4. Реалізація фізичної моделі

Після побудови логічних моделей, були створені фізичні структури реляційних баз даних для системи онлайн платежів. Під час проектування таблиць цілісною основою послужили сутності системи, а при розробці структури таблиць були враховані атрибути цих сутностей. В результаті, була сформована схема баз даних

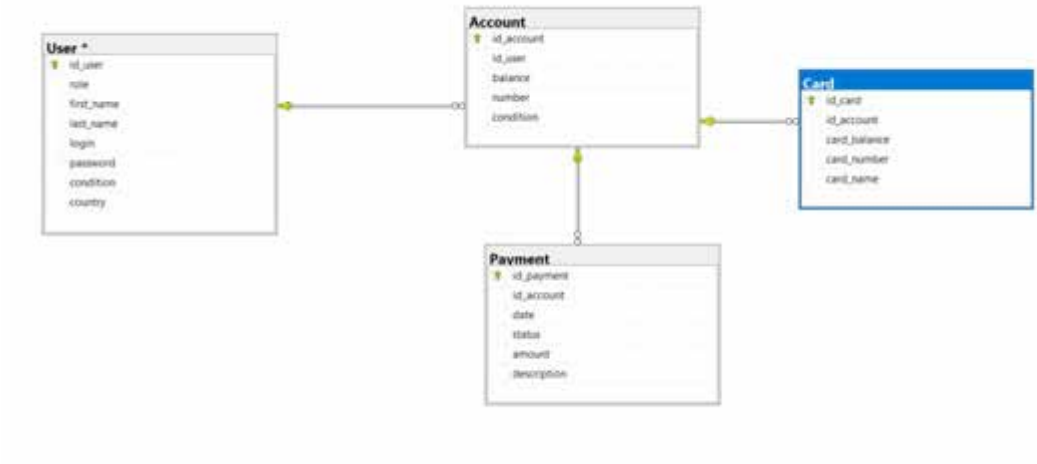


Рис.3.4. Схема бази даних онлайн системи платежів

3.5. Проектування сховища даних

Сховище даних - це структурована або неструктурована область, призначена для зберігання, управління та збереження інформації, даних та ресурсів. Сховища даних можуть бути фізичними або віртуальними, локальними або розподіленими, а також можуть використовувати різні технології для забезпечення доступу до даних.

Основні характеристики сховищ даних включають:

Зберігання даних: Сховища даних призначені для збереження інформації на тривалий термін. Це можуть бути дані клієнтів, фінансові записи, медична інформація, текстові документи, відеофайли та багато іншого.

Управління даними: Сховища даних надають інструменти для організації, індексації та категоризації інформації. Це дозволяє швидко знаходити та отримувати доступ до потрібних даних.

Захист даних: Однією з важливих функцій сховищ даних є забезпечення безпеки та конфіденційності інформації. Це включає в себе механізми автентифікації, авторизації, шифрування та резервного копіювання.

Спільний доступ: Сховища даних дозволяють багатьом користувачам одночасно отримувати доступ до даних. Це робить їх ідеальними для використання в командних проектах та обміну інформацією між користувачами.

Швидкий доступ: Сховища даних оптимізовані для швидкого пошуку та витягування інформації. Вони можуть включати індексацію та кешування, щоб полегшити доступ до даних.

Резервне копіювання і відновлення: Сховища даних зазвичай підтримують можливість створення резервних копій даних і їх відновлення в разі аварії або втрати.

Синхронізація і реплікація: Деякі сховища даних підтримують можливість синхронізації даних між різними пристроями або реплікації даних для забезпечення надійності та доступності.

Сховища даних можуть бути фізичними серверами, хмарними службами, локальними базами даних, файловими системами та іншими формами зберігання

інформації. Вони використовуються в різних сферах, включаючи бізнес, науку, охорону здоров'я, освіту та багато інших галузей для ефективного управління даними та забезпечення доступу до них.

Сховище даних є важливою складовою, необхідною для підтримки прийняття стратегічних рішень у сучасних інформаційних системах. Це предметно-орієнтований набір даних, який інтегрований, прив'язаний до часу і містить несуперечливі консолідовані історичні дані. Дані для сховища формуються на основі фіксованих протягом тривалого періоду часу моментальних знімків бази даних оперативної інформаційної системи, а також можуть походити з різних зовнішніх джерел. Для аналізу цих даних і прийняття стратегічних рішень застосовуються технології баз даних, OLAP (аналіз великих обсягів даних), інструменти глибинного аналізу даних та візуалізації.

Сховища даних призначені для даних, які змінюються рідко. Основним завданням сховищ є виконання аналітичних запитів, що допомагають керівникам і менеджерам приймати обґрунтовані рішення. При їх проектуванні важливо мати на увазі наступні вимоги: зрозуміла для користувачів структура даних, виділення статичних даних, спрощені вимоги до запитів для зменшення навантаження на систему та підтримка складних запитів SQL, які обробляють велику кількість записів.

Сховища даних відрізняються від реляційних СУБД за своєю структурою. Сховища даних не нормалізують дані, що дозволяє покращити продуктивність при виконанні аналітичних запитів. Нормалізація в реляційних СУБД призводить до створення багатьох пов'язаних таблиць, що зменшує продуктивність при складних запитах. Тому проектування сховища даних передбачає створення денормалізованої структури даних для забезпечення високої продуктивності аналітичних запитів.

Сховища даних відіграють ключову роль у підтримці прийняття стратегічних рішень в організаціях. Вони дозволяють зберігати великі обсяги історичних даних, які становлять цінний ресурс для аналізу та прогнозування подій. Завдяки сховищам даних керівництво може отримувати необхідну

інформацію для прийняття обґрунтованих рішень щодо стратегії розвитку, виробництва, маркетингу та інших сфер діяльності.

Однією з важливих особливостей сховищ даних є їх спрямованість на високопродуктивний аналіз даних. Вони дозволяють виконувати складні запити, які можуть охоплювати мільйони записів, і при цьому забезпечувати швидкість виконання. Це особливо важливо при роботі з великими обсягами інформації, де традиційні реляційні СУБД можуть ставити обмеження на продуктивність.

Загалом, сховища даних є потужним інструментом для організацій, які прагнуть використовувати свої дані для прийняття обґрунтованих стратегічних рішень. Вони допомагають структурувати історичні дані, забезпечують швидкий доступ до них та дозволяють виконувати складні аналітичні операції.

OLAP (Online Analytical Processing) - це технологія оперативного аналізу даних, яка використовує методи та інструменти для збору, зберігання та аналізу багатовимірних даних з метою підтримки прийняття стратегічних рішень. Основною метою систем OLAP є підтримка аналітичної діяльності користувачів-аналітиків та перевірка гіпотез.

OLAP визначає модель подання даних та технології їх обробки в сховищах даних. Використання багатовимірного представлення даних дозволяє отримувати швидкий доступ до стратегічно важливої інформації для глибинного аналізу. Основні вимоги до додатків OLAP включають багатовимірну структуру даних, підтримку складних розрахунків та врахування часового аспекту.

Переваги використання OLAP включають підвищення продуктивності персоналу, можливість внесення змін у схему користувачами, збереження контролю над цілісністю даних та зменшення навантаження на системи OLTP і сховища даних.

OLAP допомагає аналізувати дані та представляти результати у зручному форматі для прийняття рішень. Його основна ідея полягає в побудові багатовимірних кубів для доступу до даних. Зазвичай дані для цих кубів зберігаються в реляційних базах даних, спеціально призначених для обробки та аналізу інформації.

Структура сховища даних відрізняється від звичайних реляційних СУБД. Вона може бути денормалізованою, що дозволяє підвищити швидкість виконання запитів. Основними складовими структури сховища даних є таблиця фактів та таблиці вимірів, які містять необхідні дані для аналізу. У такій структурі сховища даних визначається відношення між даними та їх доступність для користувачів.

За допомогою програмного середовища SQL Server Management Studio було спроектоване сховище даних, що зображене на рис.3.5, яке призначене для зберігання та аналізу великих обсягів даних. СД містить таблицю фактів – PaymentFact та 3 таблиці вимірів:

- DateDim
- CardDim
- UserDim

Таблиця PaymentFact містить відомості про платежі дату, користувача, і картки які брали участь у транзакції.

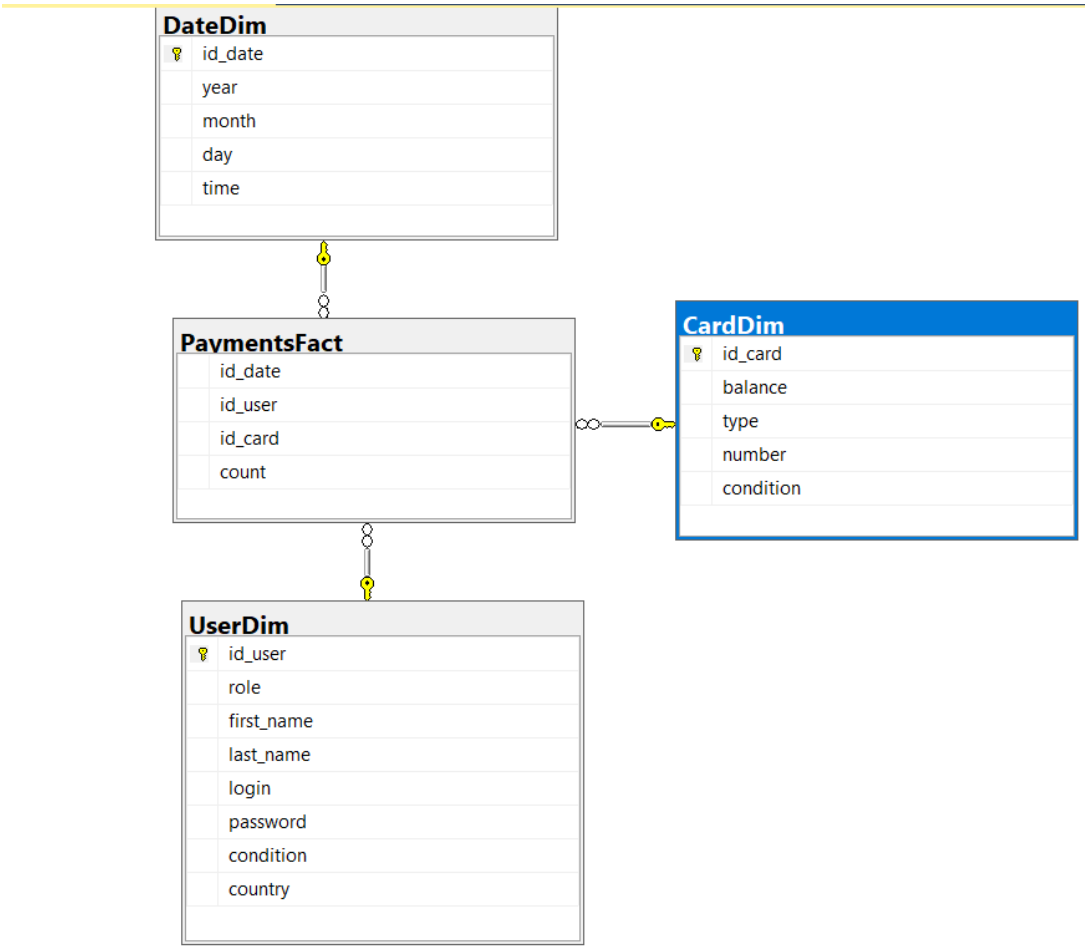


Рис.3.5. Сховище даних системи аналітики онлайн платежів

3.6. Розгортання OLAP-куба

Розгортання OLAP-куба (Online Analytical Processing Cube) - це процес створення і налаштування багатовимірної моделі даних для аналітичної обробки. Основною метою розгортання OLAP-куба є підготовка даних для швидкого та ефективного аналізу в контексті аналітичних систем, таких як системи управління платежами.

Основні етапи розгортання OLAP-куба включають наступне:

Збір та підготовка даних: Спочатку потрібно зібрати та очистити дані з різних джерел, таких як операційні бази даних, зовнішні джерела та інші. Це включає в себе етапи імпорту, трансформації та завантаження (ETL), де дані перетворюються в підходящий формат для аналізу.

Визначення структури куба: Після підготовки даних визначається структура OLAP-куба. Вона включає в себе виміри (dimensions), які представляють атрибути даних, і факти (facts), які представляють числові показники, що аналізуються. Структура куба визначає, як дані будуть організовані для подальшого аналізу.

Створення індексів та агрегацій: Для підвищення швидкодії аналізу додатково створюються індекси та агреговані дані в OLAP-кубі. Це допомагає зменшити час відповіді на аналітичні запити.

Розгортання куба: На даному етапі OLAP-куб створюється і заповнюється даними, що включають в себе підготовлені виміри та факти. Розгортання може виконуватися в реляційних базах даних або спеціалізованих OLAP-системах.

Рекомендації до розгортання OLAP-куба для аналітичної системи платежів:

Визначення бізнес-потреб: Перш за все, важливо чітко визначити бізнес-потреби та аналітичні завдання, які потрібно вирішити за допомогою OLAP-куба. Це допоможе правильно визначити структуру куба та виміри.

Оптимізація ETL-процесу: Важливо оптимізувати процес імпорту, трансформації та завантаження даних для забезпечення ефективності та актуальності даних в кубі.

Розробка плану агрегацій: Планування та створення агрегацій допомагає підвищити швидкість аналізу та зменшити навантаження на систему.

Тестування та оптимізація: Після розгортання OLAP-куба важливо провести тестування та оптимізацію для забезпечення ефективності та якості аналізу.

Навчання користувачів: Організуйте навчання користувачів, які будуть працювати з OLAP-кубом, щоб вони могли максимально використовувати його можливості для прийняття рішень.

Моніторинг та підтримка: Забезпечте постійний моніторинг та підтримку OLAP-куба для забезпечення актуальності та продуктивності аналітичних даних.

Під час розробки проекту для онлайн платежів у середовищі Business Intelligence Development Studio, проект регулярно розгортається на сервері розробки з метою створення бази даних для послуг аналітичної системи. Це необхідно для тестування та перевірки різних аспектів проекту в контексті операцій онлайн платежів, а також для створення статистики.

Під час розгортання проекту автоматично виконується наступний порядок дій:

Побудова проекту, в результаті якої створюються вихідні файли, що визначають структуру бази даних для аналітичної системи онлайн платежів.

Перевірка доступності обраного сервера для розгортання.

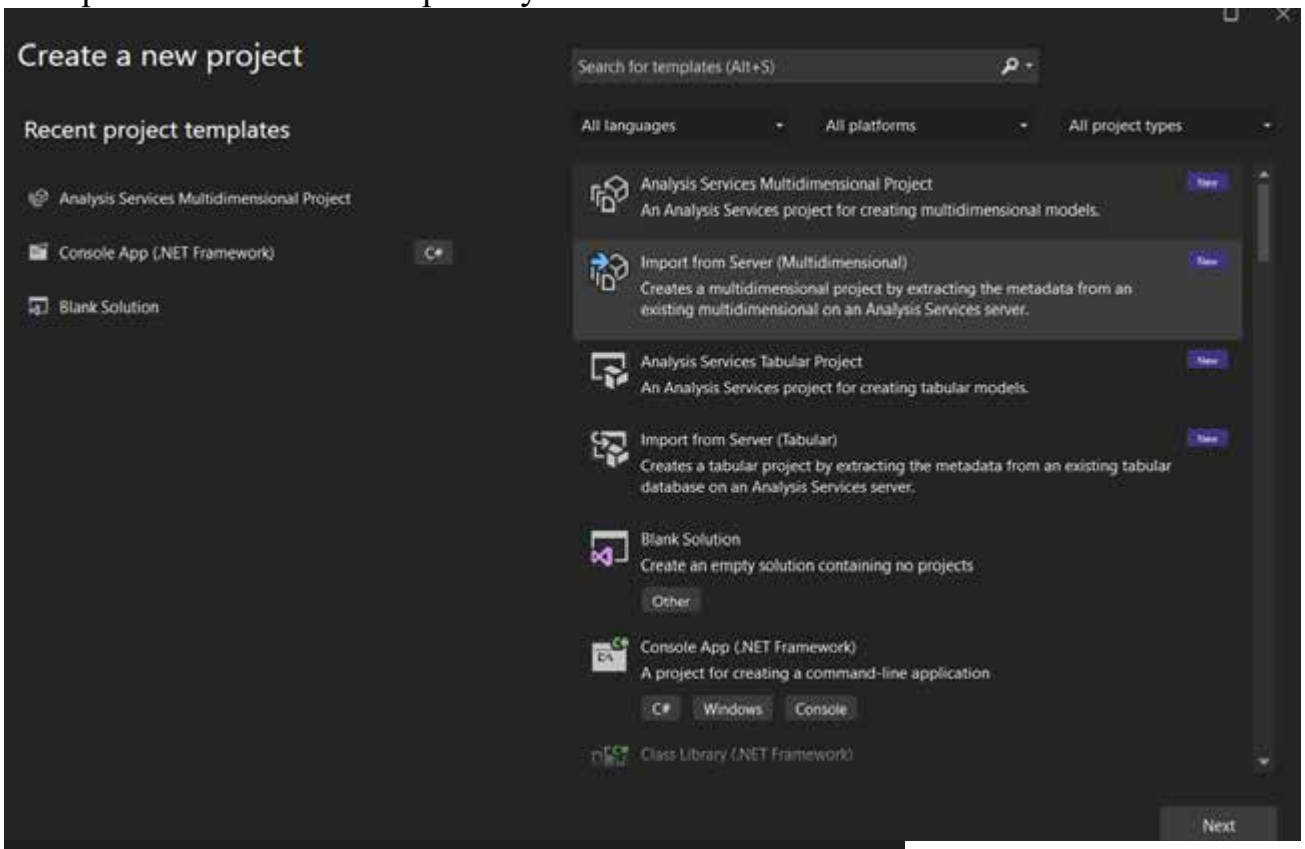
Створення бази даних та об'єктів аналітичної системи на обраному сервері.

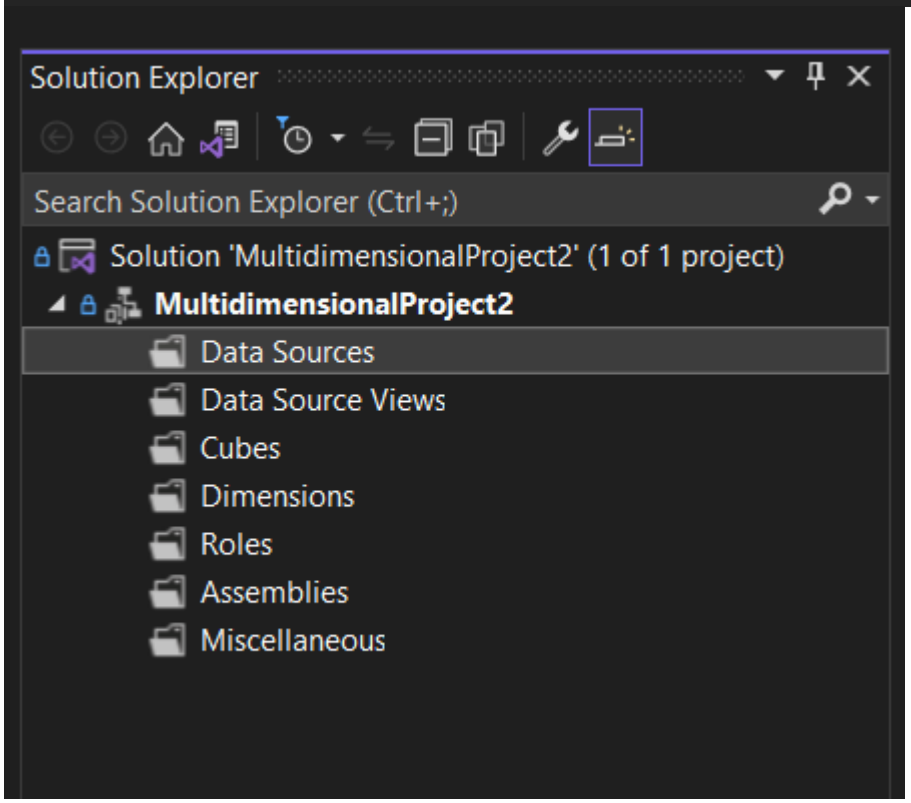
Під час процесу розгортання, існуюча база даних повністю замінюється вмістом проекту для забезпечення актуальності та коректності даних, що використовуються в системі онлайн платежів.

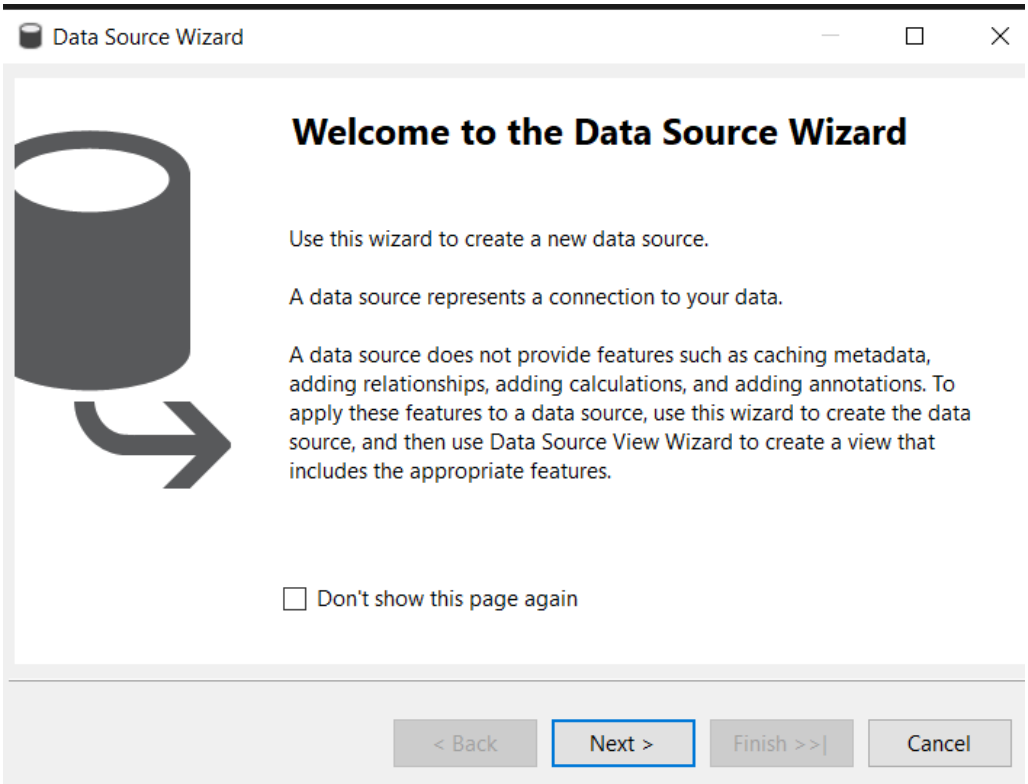
За допомогою SQL Server Visual Studio було успішно створено куб, який діаграмується на рис. 3.6 і використовується для аналізу даних у сфері онлайн платежів.

Плоя цього здійснені наступні кроки які проілюстровані на скріншотах.

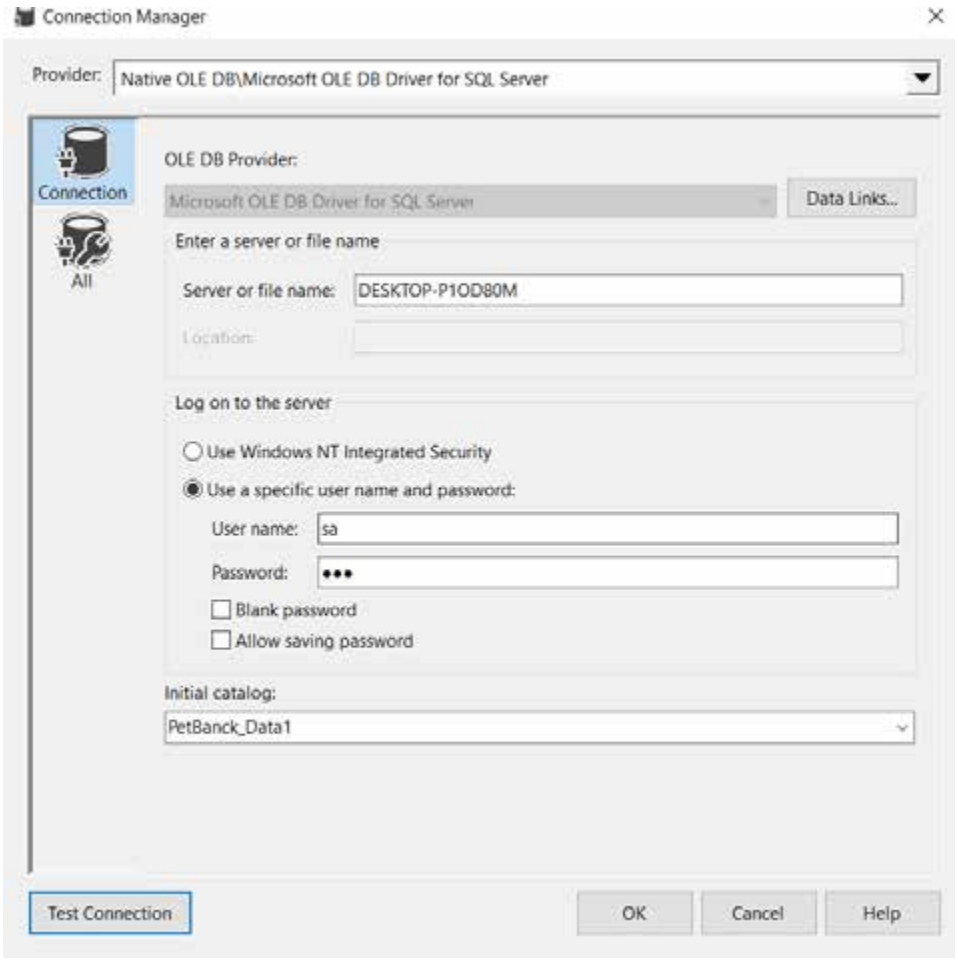
Створюємо аналітичний проект у Visual Studio

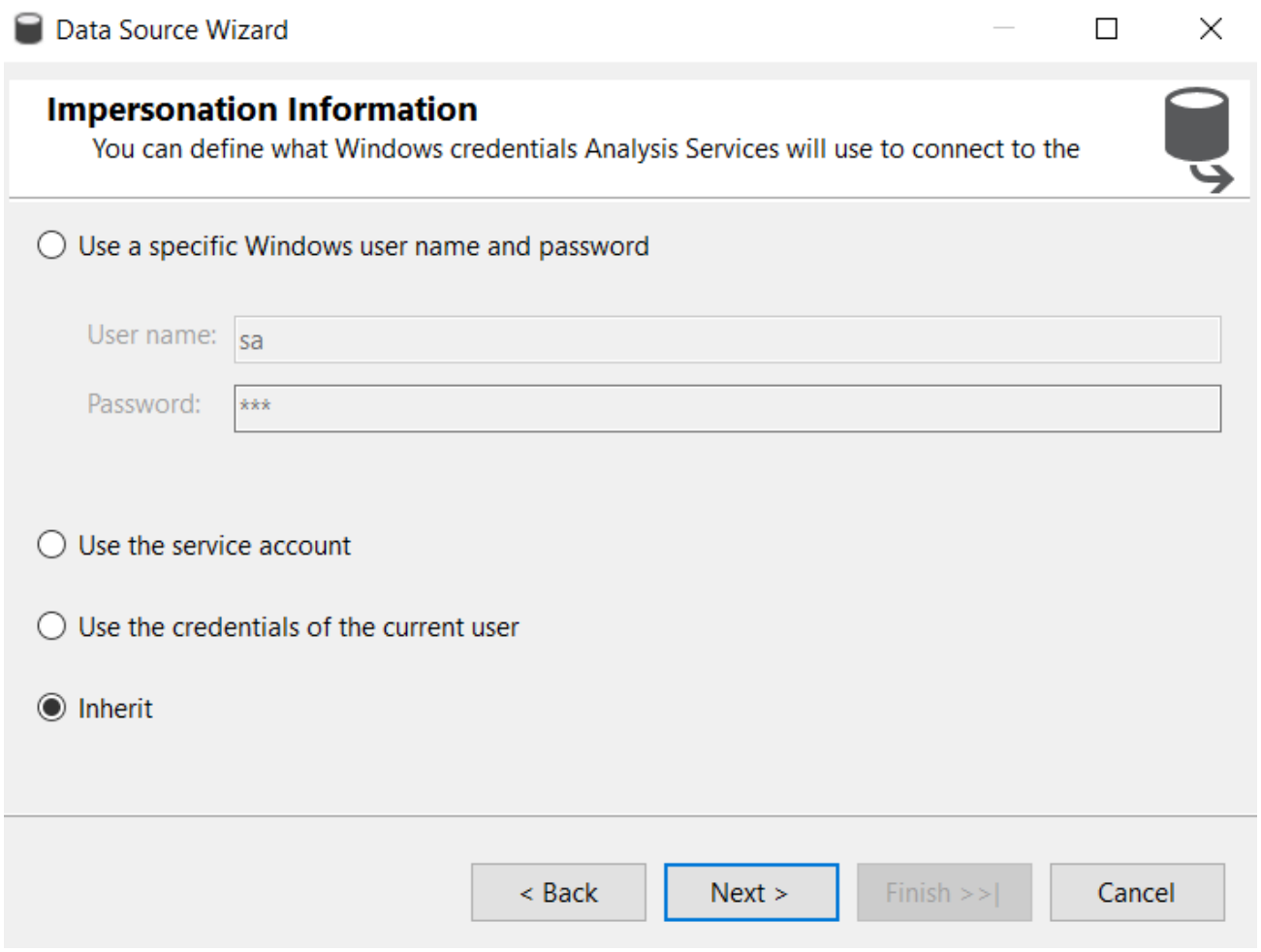




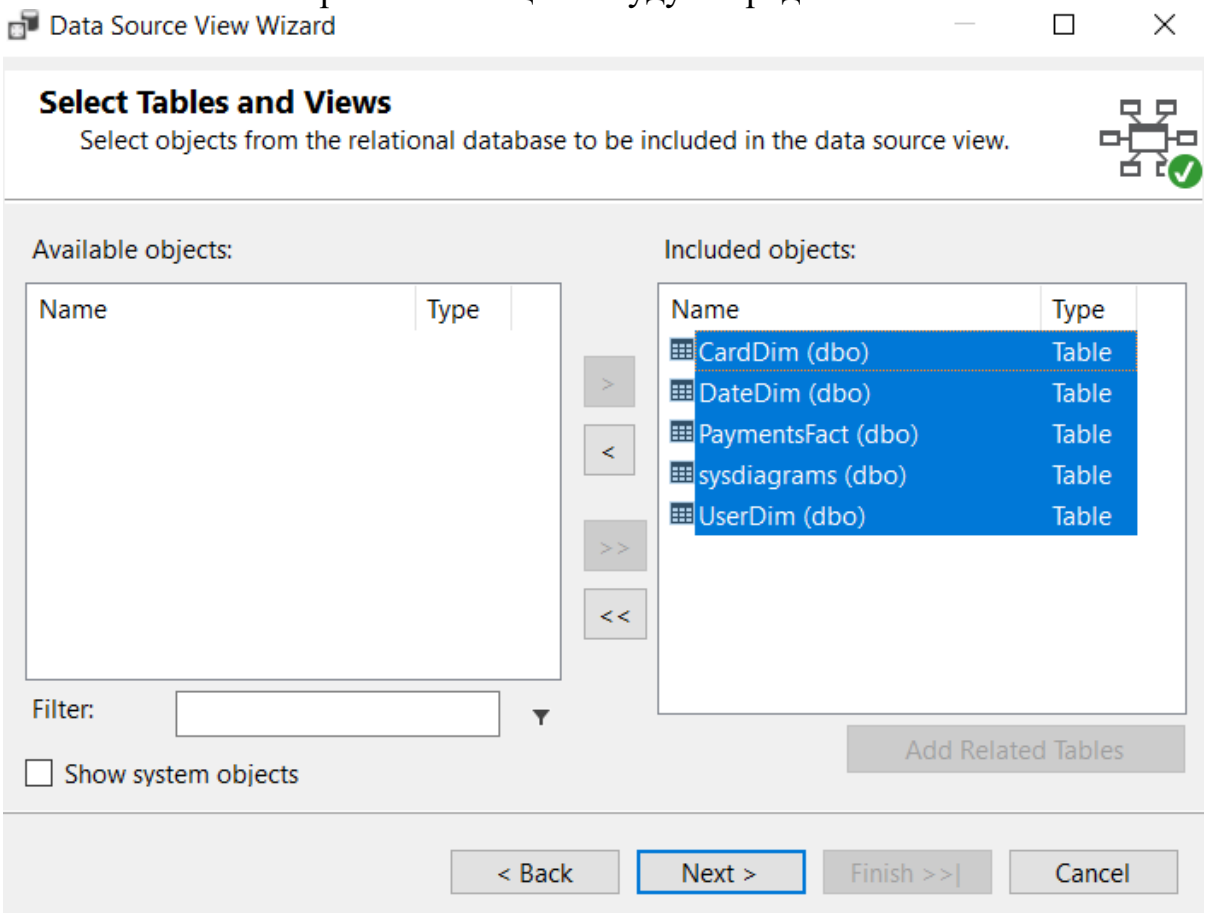


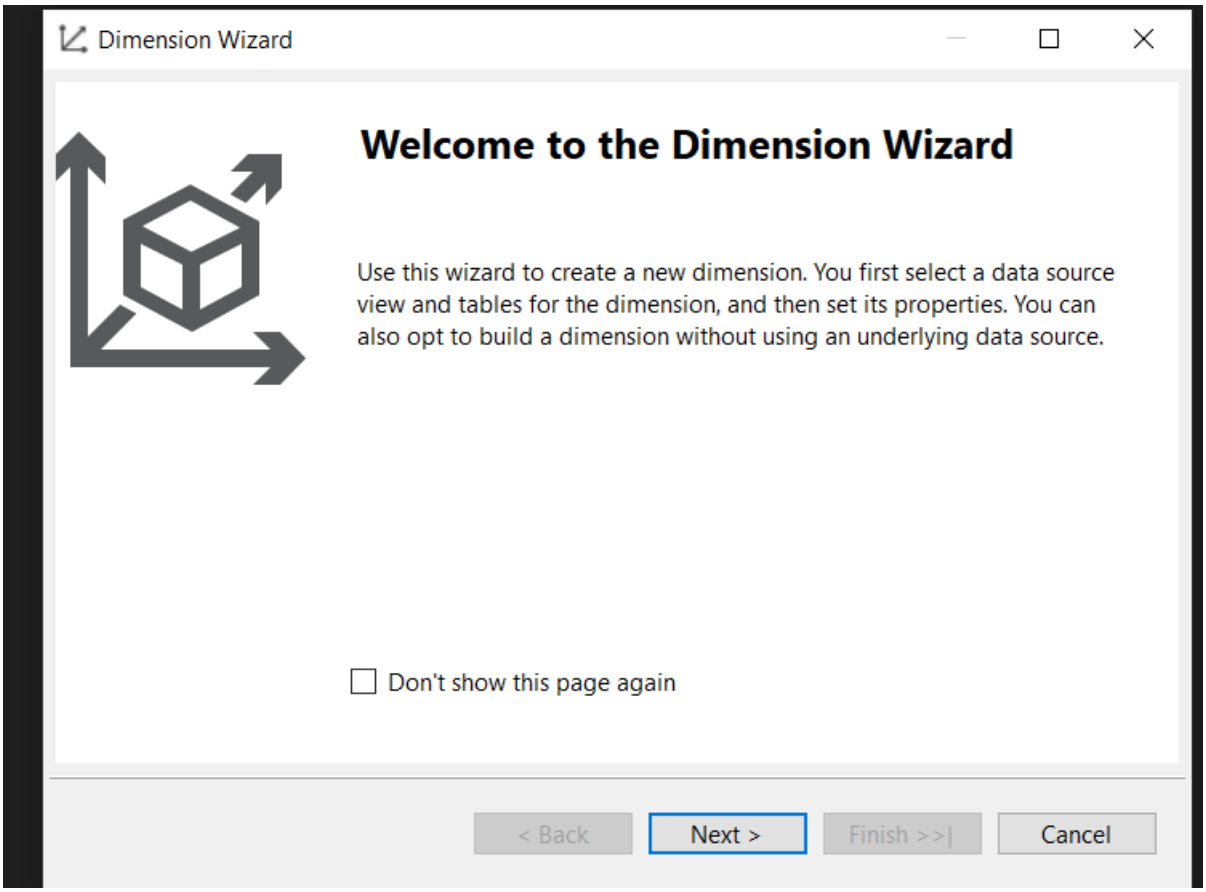
Налаштовуємо з’єднання

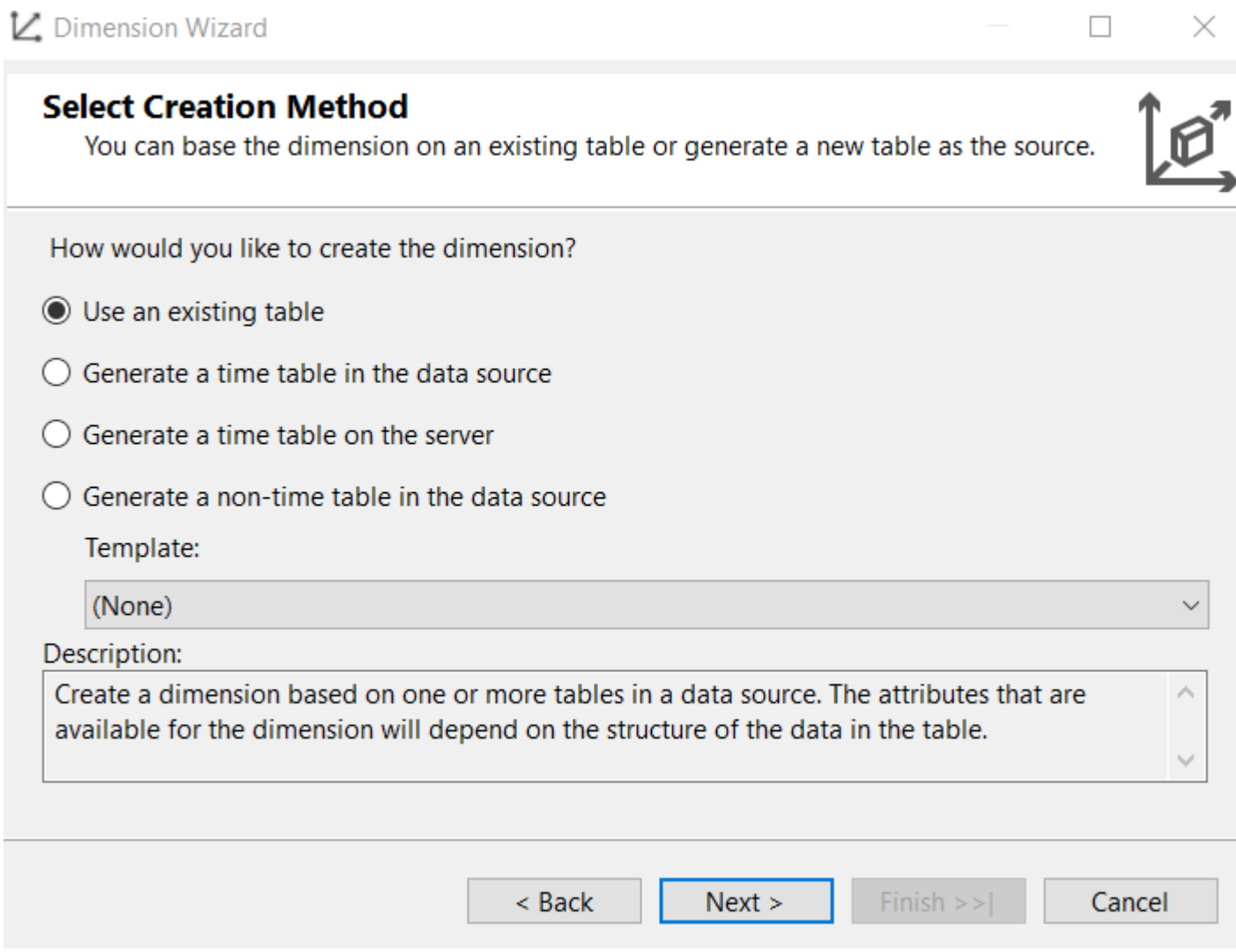

Connection Manager
Provider:
Native OLE DB\Microsoft OLE DB Driver for SQL Server
Connection
All
OLE DB Provider:
Microsoft OLE DB Driver for SQL Server
Data Links...
Enter a server or file name
Server or file name:
DESKTOP-P1OD80M
Location:
Log on to the server
Use Windows NT Integrated Security
Use a specific user name and password:
User name:
sa
Password:
Blank password
Allow saving password
Initial catalog:
PetBanck_Data1
Test Connection
OK
Cancel
Help


Data Source Wizard
Impersonation Information
You can define what Windows credentials Analysis Services will use to connect to the
Use a specific Windows user name and password
User name:
sa
Password:
***
Use the service account
Use the credentials of the current user
Inherit
< Back
Next >
Finish >>|
Cancel

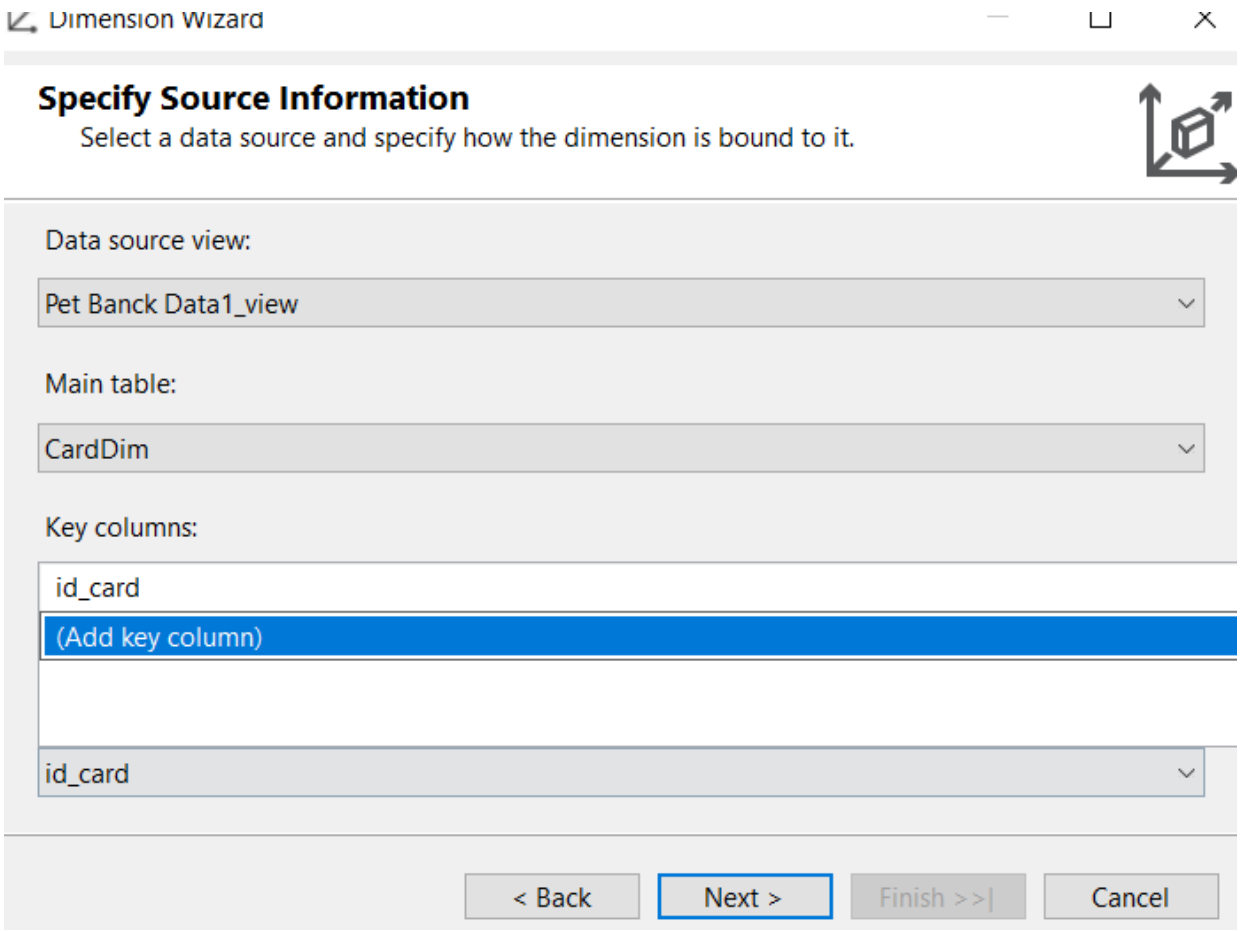
Обираємо таблиці які будуть представленні

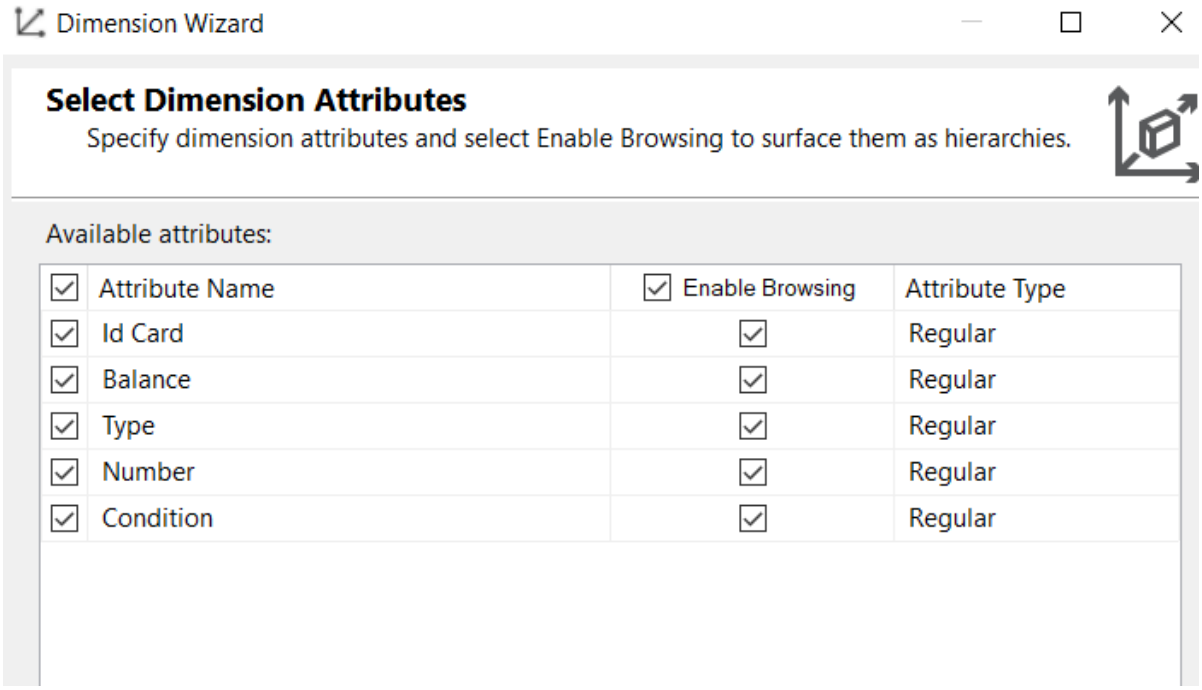



Dimension Wizard
Welcome to the Dimension Wizard
Use this wizard to create a new dimension. You first select a data source view and tables for the dimension, and then set its properties. You can also opt to build a dimension without using an underlying data source.
Don't show this page again
< Back
Next >
Finish >>|
Cancel

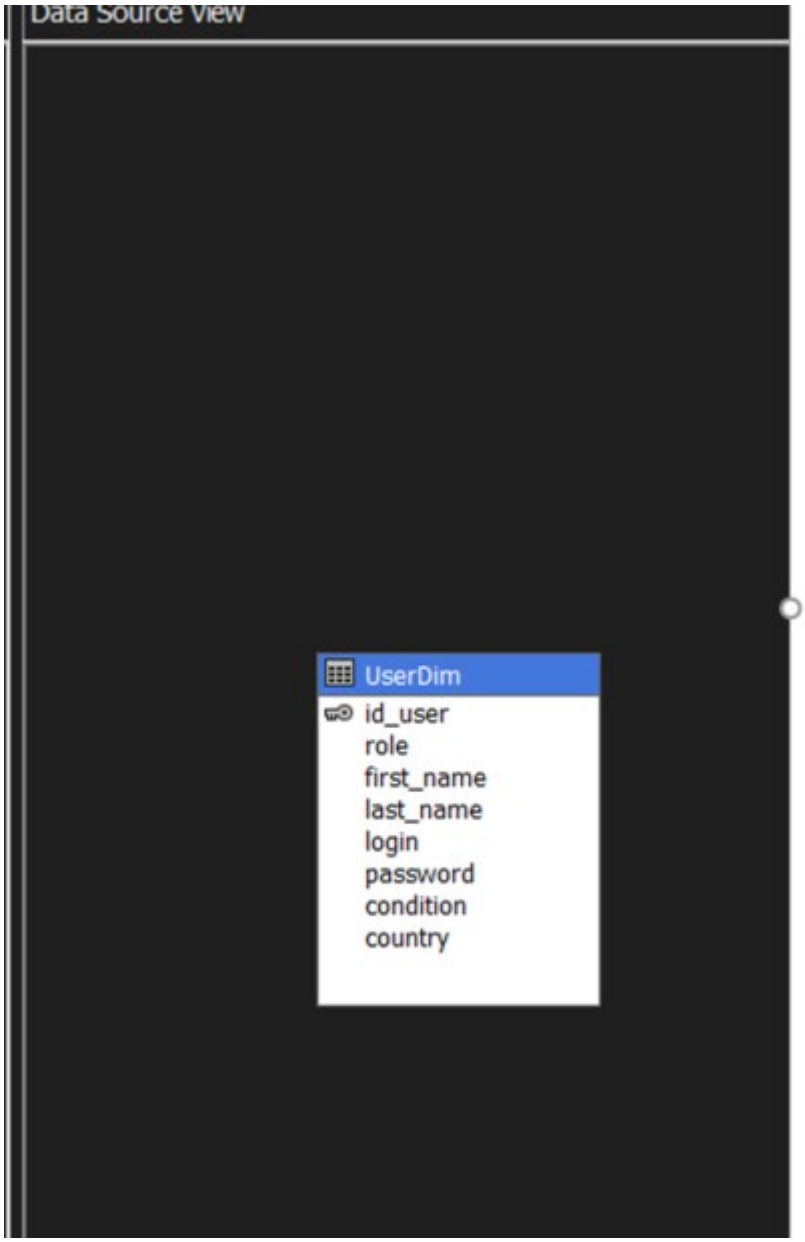

Dimension Wizard
Select Creation Method
You can base the dimension on an existing table or generate a new table as the source.
How would you like to create the dimension?
Use an existing table
Generate a time table in the data source
Generate a time table on the server
Generate a non-time table in the data source
Template:
(None)
Description:
Create a dimension based on one or more tables in a data source. The attributes that are available for the dimension will depend on the structure of the data in the table.
< Back
Next >
Finish >>|
Cancel

Обираємо поля які будуть відображені з кожної таблиці

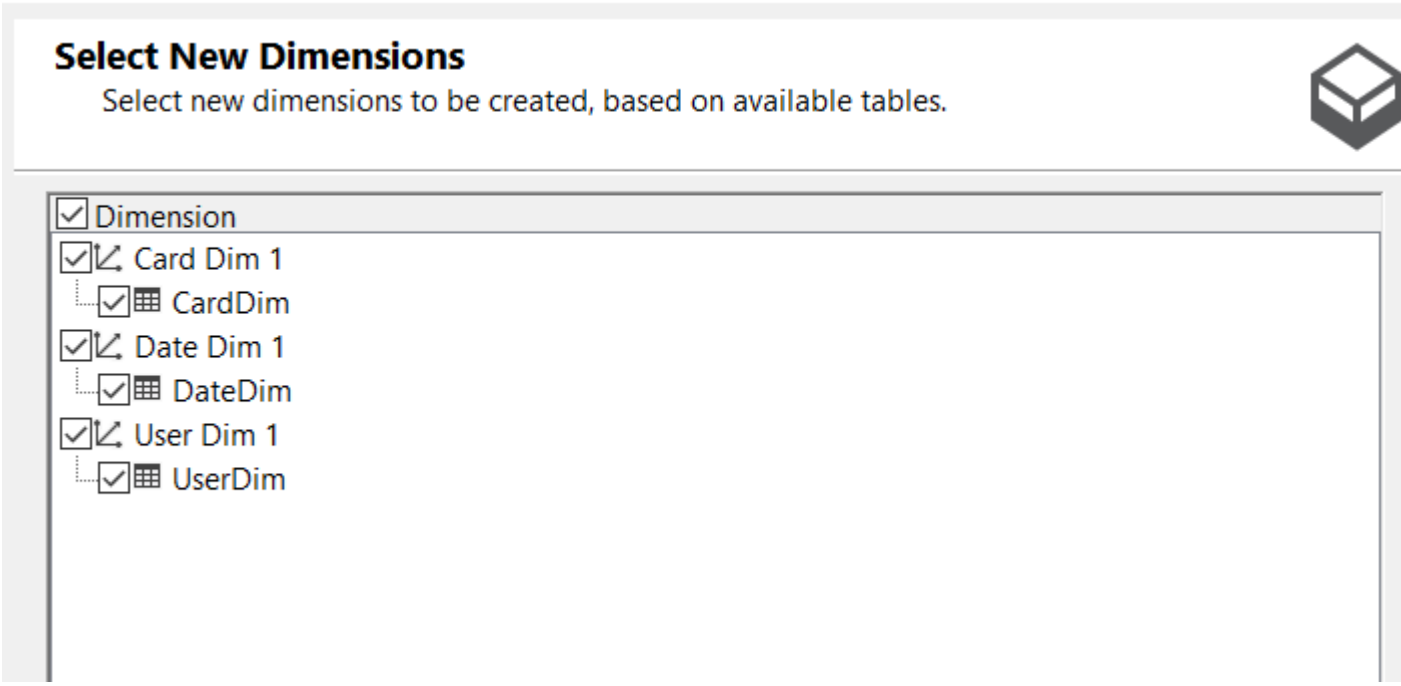




Готова таблиця користувача виміру



Обираємо всі талиці виміру



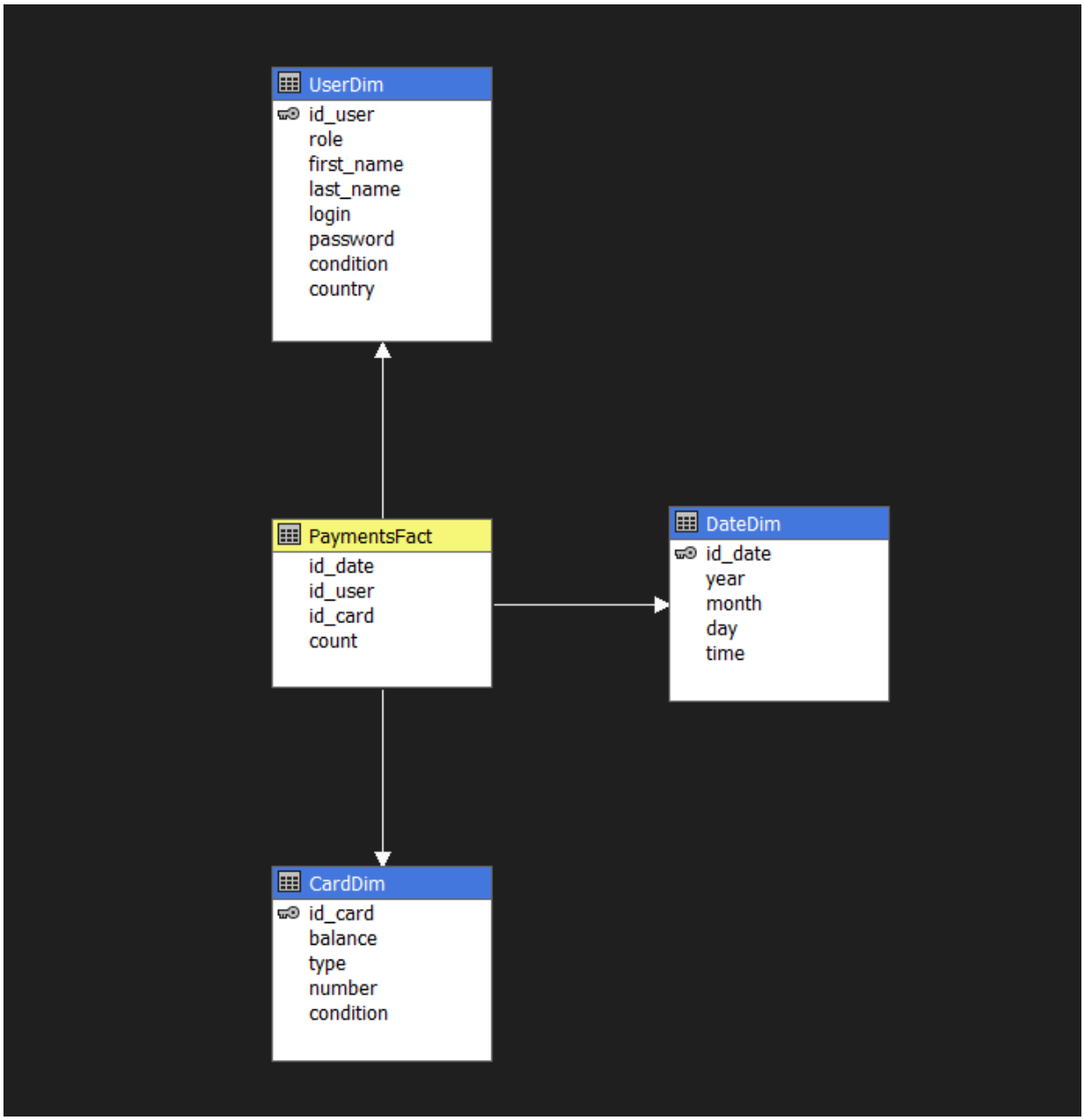
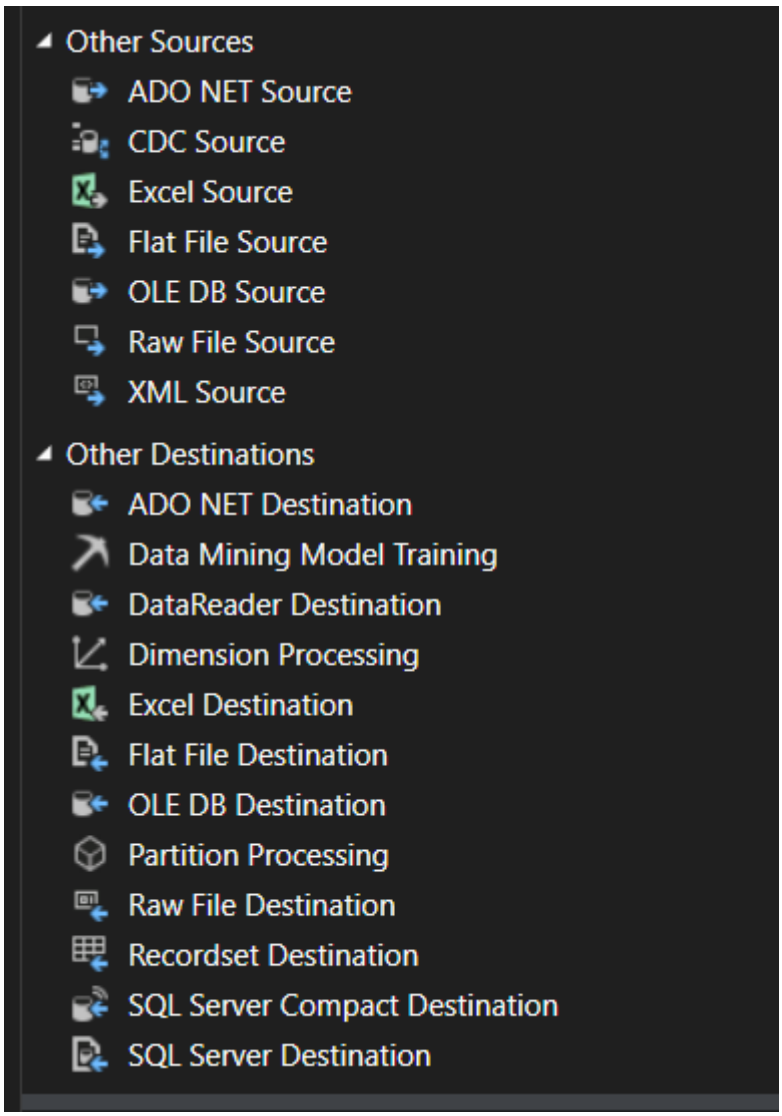


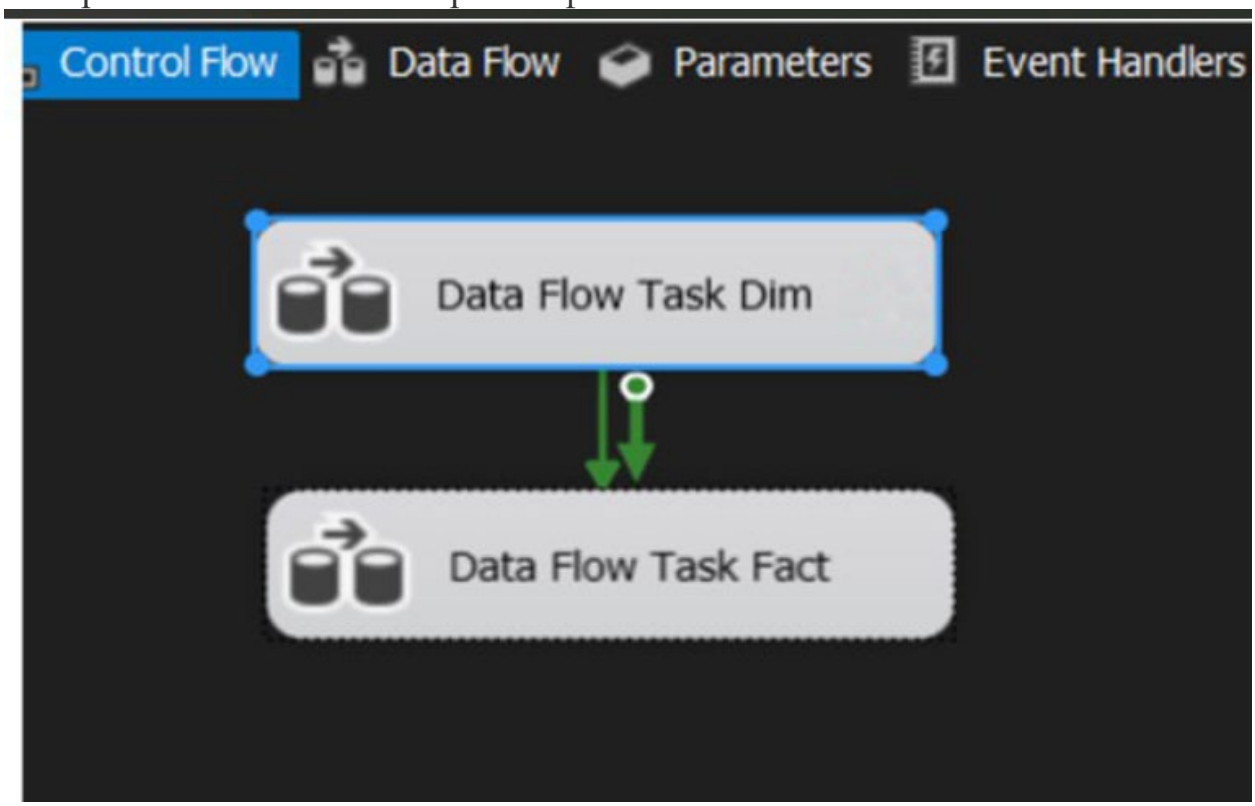
Рис.3.6. Куб системи системи аналітики онлайн платежів

Заповнюємо куб даними

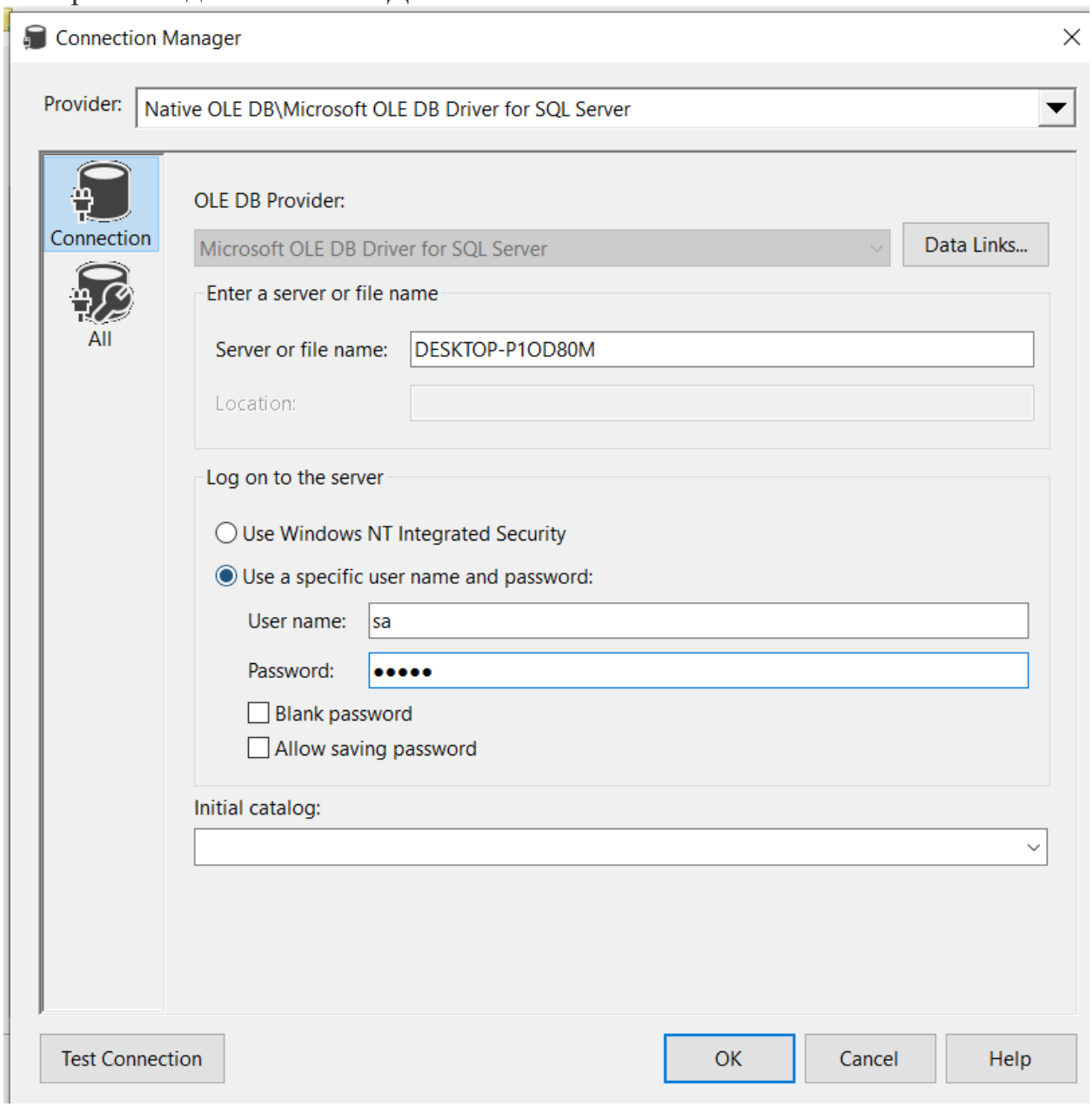
На першому етапі заповнюємо всі ключові поля таблиці-фактів.

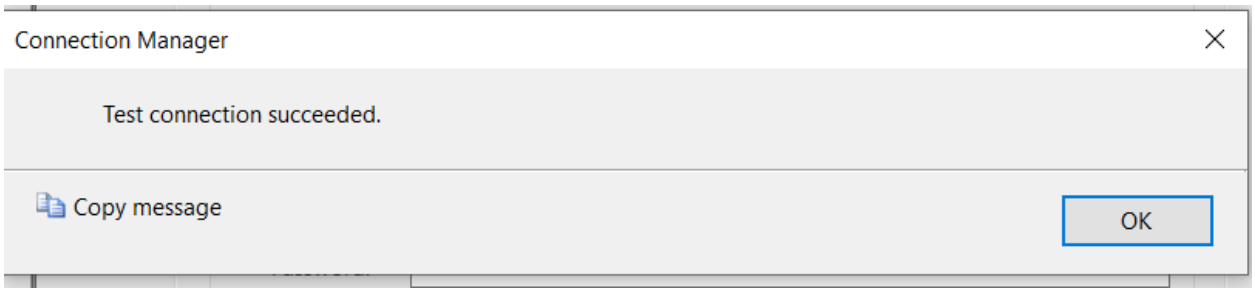


Створюємо звяски між вимірами і фактами

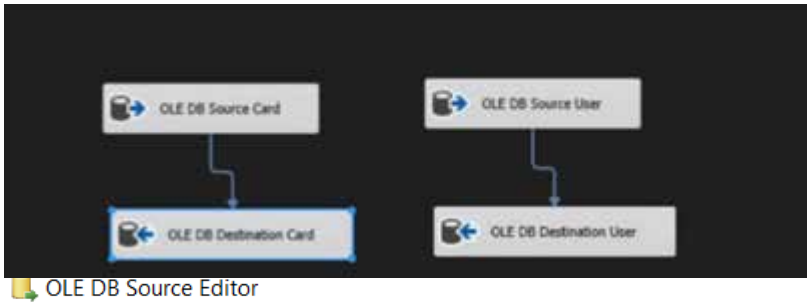


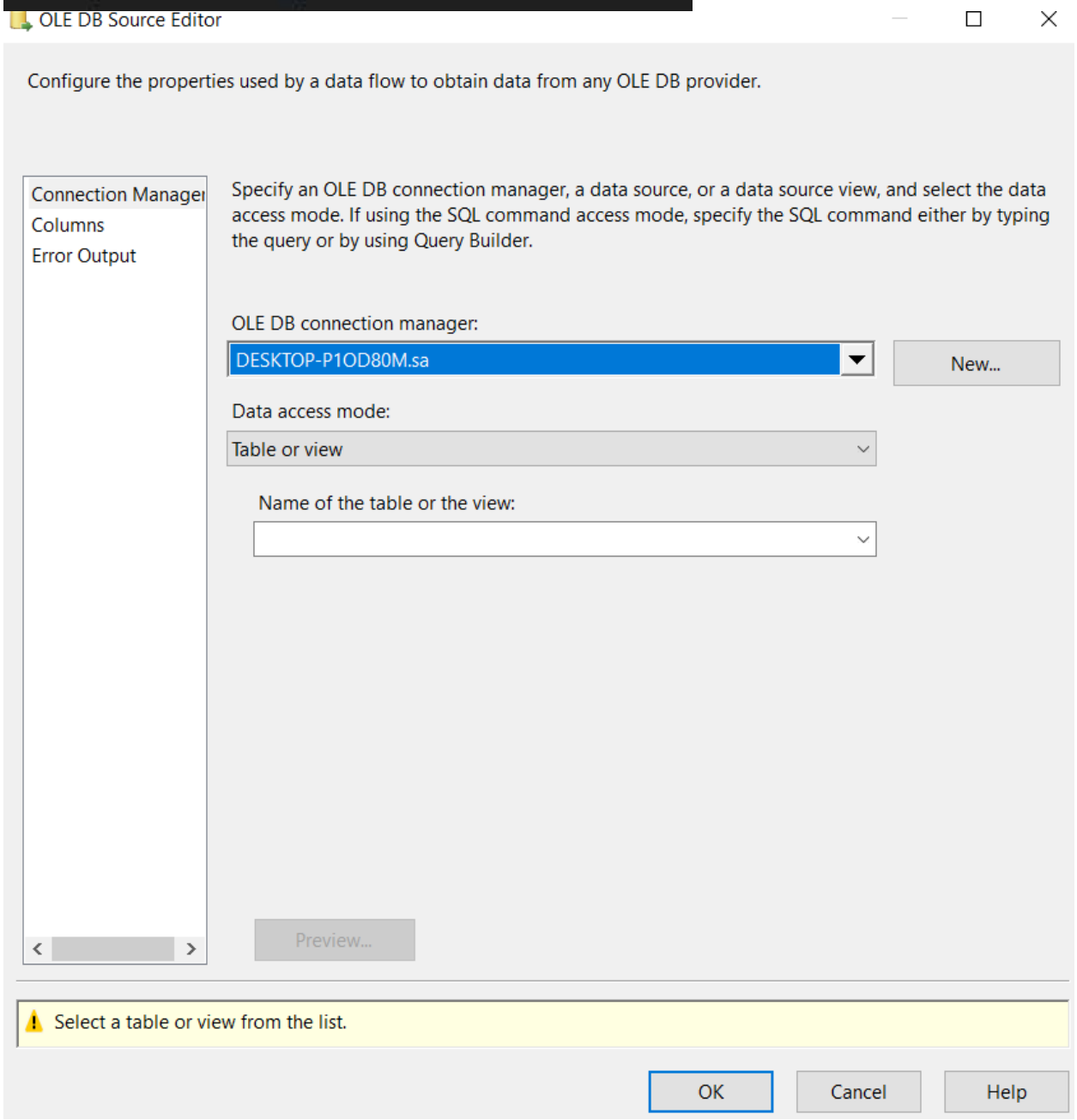
Створення підключення з БД



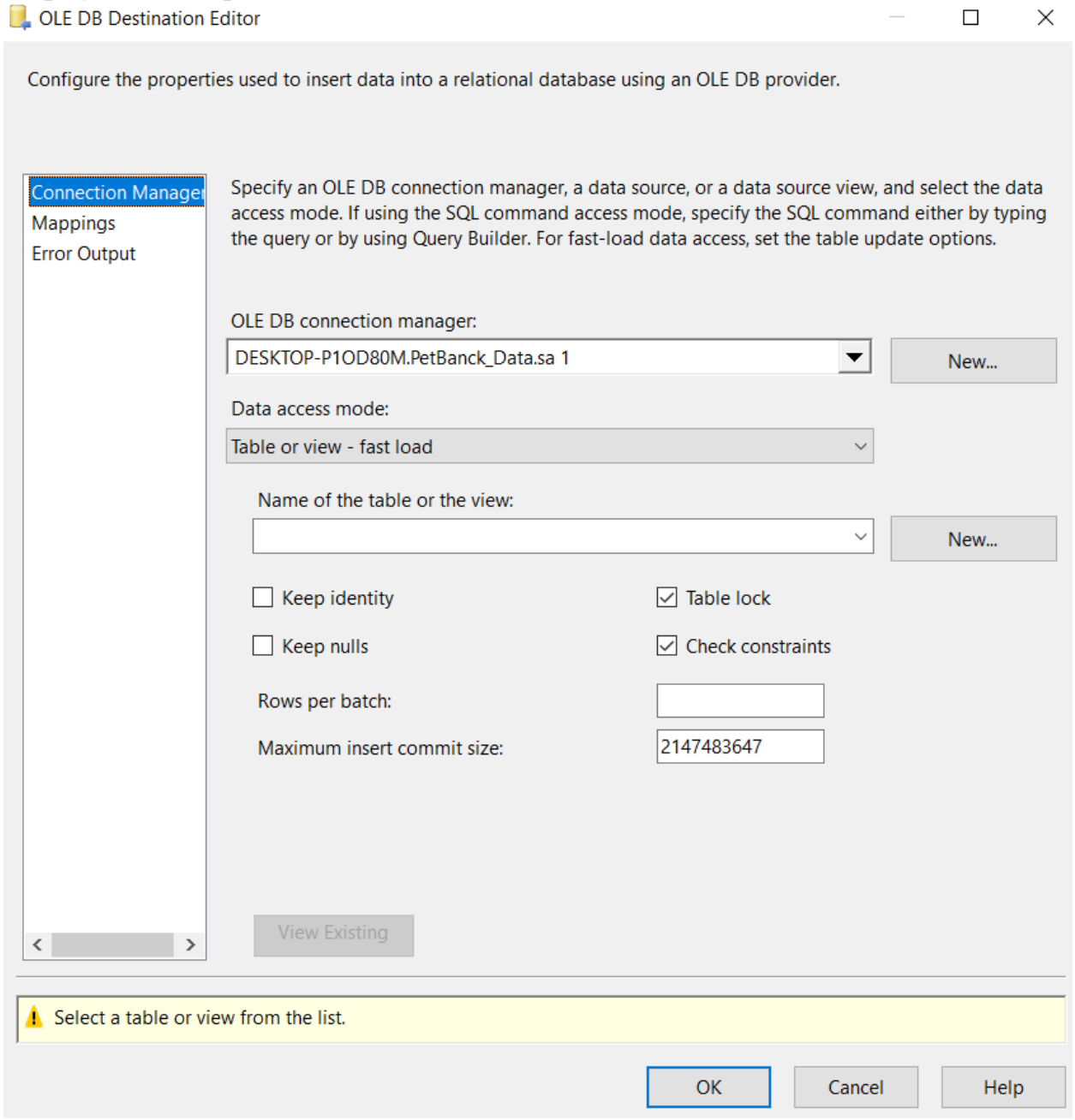


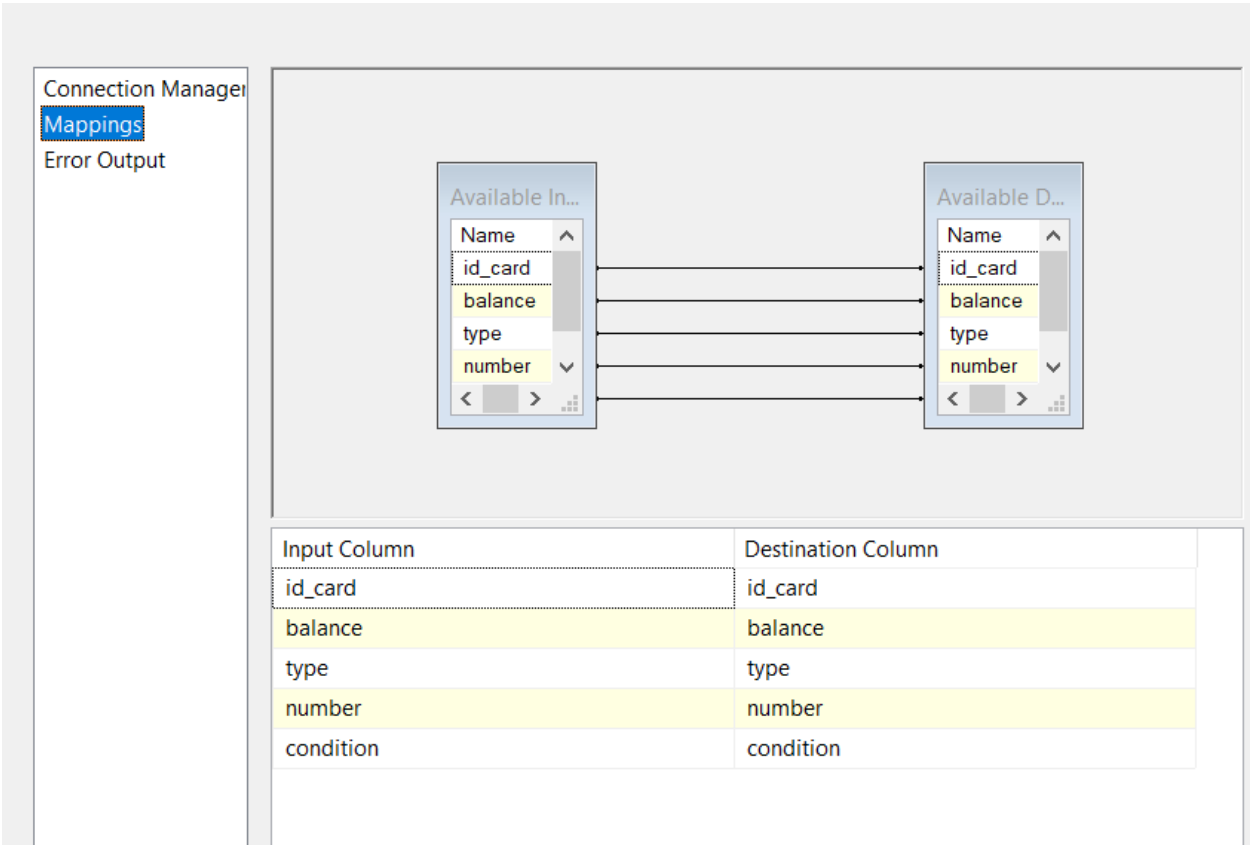
На другому етапі заповнюємо всі ключові поля таблиці-вимірів.



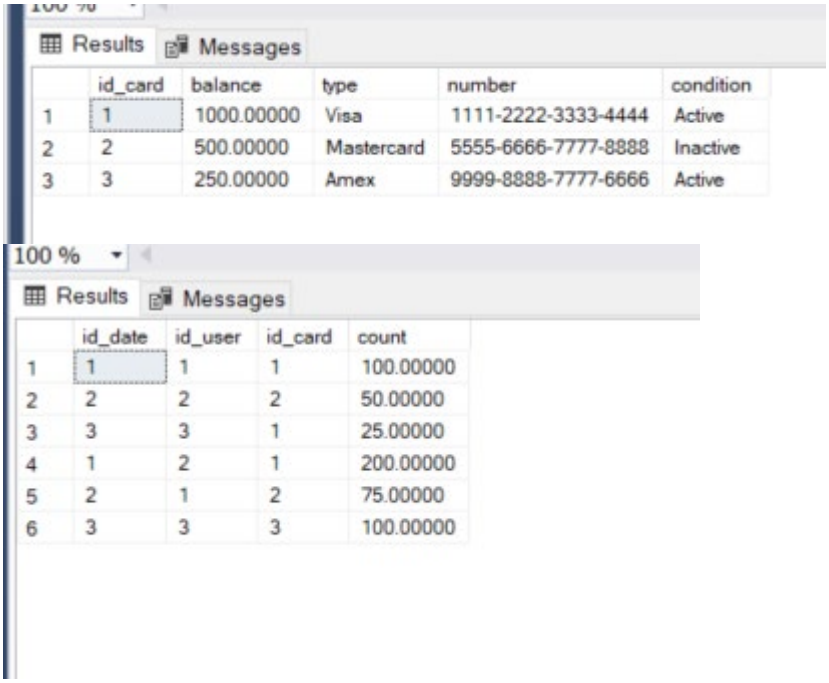


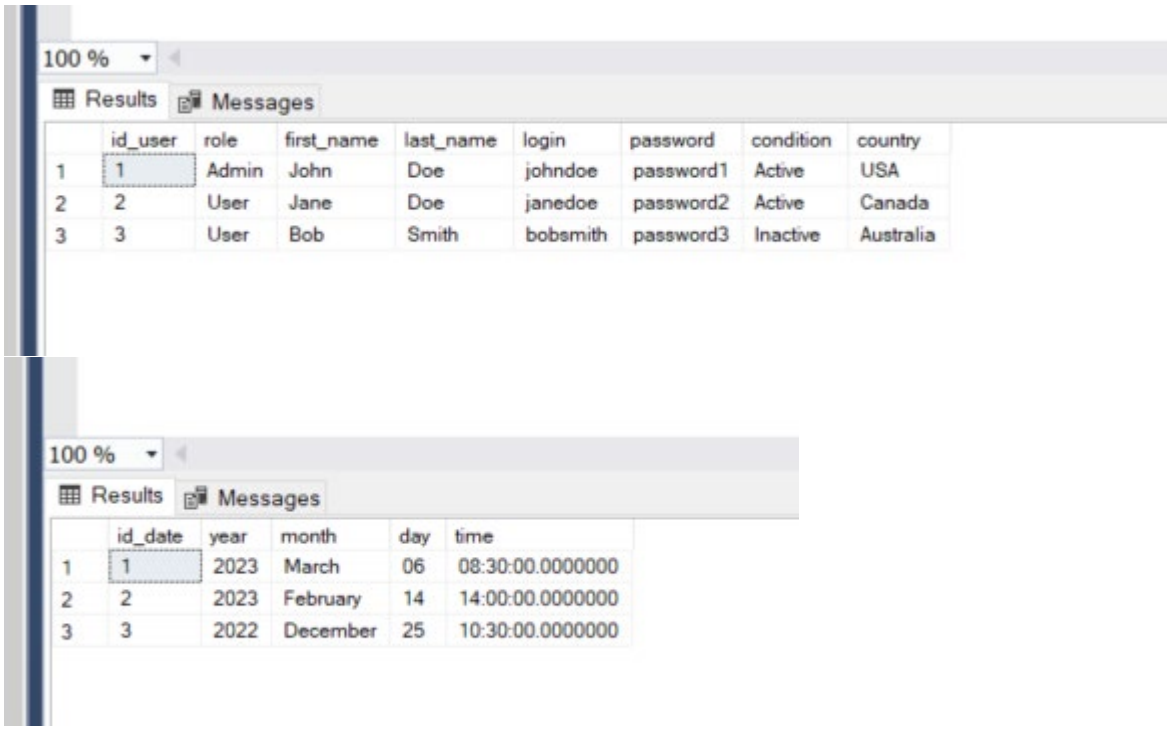
Формуваня вибірки





Заповнені данні





100 %

Results Messages

	id_user	role	first_name	last_name	login	password	condition	country
1	1	Admin	John	Doe	johndoe	password1	Active	USA
2	2	User	Jane	Doe	janedoe	password2	Active	Canada
3	3	User	Bob	Smith	bobsmith	password3	Inactive	Australia

100 %

Results Messages

	id_date	year	month	day	time
1	1	2023	March	06	08:30:00.0000000
2	2	2023	February	14	14:00:00.0000000
3	3	2022	December	25	10:30:00.0000000

Ми отримали готове сховище данних із заповненими даними

4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

4.1. Вибір, обґрунтування інструментарію та спеціальні програмні засоби для розробки програмного забезпечення

Інструментарій та спеціальні програмні засоби для розробки програмного забезпечення (ПЗ) - це набір інструментів, середовищ та програм, які призначені для допомоги розробникам у створенні, тестуванні, аналізі та підтримці програмного забезпечення. Ці засоби надають інженерам зручний спосіб розробки програм та допомагають їм прискорити процес створення програмних продуктів.

Інструментарій розробки ПЗ включає в себе різноманітні комп'ютерні програми та середовища розробки, які дозволяють програмістам писати, тестувати та налагоджувати код, взаємодіяти з системами управління версіями, документувати код, автоматизувати процеси розробки, та інше.

Спеціальні програмні засоби для розробки ПЗ включають в себе бібліотеки, фреймворки, SDK (набори для розробки програм), компілятори, інструменти для аналізу коду, відладчики, а також засоби для тестування, профілювання, моделювання і автоматизації.

Використання інструментарію та спеціальних програмних засобів для розробки ПЗ спрощує та покращує якість роботи розробників, допомагає ефективніше керувати проектами, прискорює процес розробки та знижує можливість помилок у програмах.
Інструментарій та спеціальні програмні засоби для розробки програмного забезпечення (ПЗ) відіграють ключову роль у процесі створення програмних продуктів і дозволяють розробникам досягати наступних важливих цілей:

Підвищення продуктивності: Інструменти розробки надають широкий набір функцій для зручного редагування та написання коду, автозаповнення, відлагодження, підсвічування синтаксису і багато іншого. Вони допомагають розробникам швидше та ефективніше створювати програми.

Забезпечення якості ПЗ: Інструменти аналізу коду, автоматичного тестування та профілювання дозволяють виявляти та виправляти помилки, підвищувати стійкість програм та підвищувати якість коду. Це допомагає запобігати виникненню багів та вад у програмі.

Керування версіями і спільною роботою: Системи контролю версій (наприклад, Git) допомагають командам розробників спільно працювати над проектами, відстежувати зміни, об'єднувати їх та вирішувати конфлікти. Інструменти для спільної роботи над кодом дозволяють команді ефективно співпрацювати в реальному часі.

Автоматизація процесів: Інструменти автоматизації дозволяють автоматизувати рутинні завдання та процеси розробки, такі як збірка, розгортання та тестування. Це спрощує та прискорює розробку та знижує ризик помилок.

Підтримка різних мов та платформ: Інструменти розробки підтримують різні мови програмування та платформи, що дозволяє розробникам працювати з різноманітними технологіями та стеками.

Сприяння документації: Багато інструментів дозволяють автоматично створювати документацію до коду, що полегшує розуміння та підтримку програмного продукту.

Підтримка відладки та аналізу коду: Інтегровані відлагоджувачі, аналізатори коду та інструменти для виявлення проблем допомагають розробникам знайти та виправити помилки у коді.

Широкий вибір сторонніх бібліотек і фреймворків: Спеціальні програмні засоби, такі як SDK, фреймворки та бібліотеки, розширюють можливості розробників та допомагають швидко створювати функціональності за допомогою готових рішень.

Загалом, інструментарій та спеціальні програмні засоби для розробки ПЗ роблять процес розробки більш продуктивним, надійним і ефективним, допомагаючи розробникам створювати якісні програмні продукти, вчасно впроваджувати зміни та зменшувати витрати часу та зусиль.

4.1.1. Visual Studio

Розвиток обчислювальної техніки та зростаюча потреба у високоефективних інструментах для розробки програмного забезпечення привели до виникнення численних систем програмування, спрямованих на "швидку розробку". Серед них особливу увагу заслуговують Microsoft Visual Basic та Borland Delphi. В основі таких систем лежить технологія візуального проектування та подійного програмування, яка дозволяє програмістам ефективно створювати програми, спрощуючи генерацію коду та дозволяючи більше уваги приділяти створенню інтерфейсу та обробці подій.

Delphi отримав визнання завдяки успіху та популярності, що викликав бажання компанії Borland розширити методи "швидкої розробки" у сферу професійного програмування. Це призвело до появи Borland C++ Builder.

Borland C++ Builder - це середовище "швидкої розробки", яке використовує мову програмування C++. На сьогоднішній день воно є одним із найпопулярніших інструментів для створення проектів під операційною системою Windows. Система містить близько 200 різних компонентів і вимагає мінімум зусиль для створення готових програм. Швидкодія програм, розроблених за допомогою C++ Builder, значно перевершує аналогічні програми, створені з використанням інших інструментів, таких як MS Visual Basic.

C++ Builder підтримує основні принципи об'єктно-орієнтованого програмування, включаючи інкапсуляцію, поліморфізм та множинне спадкування. Ця система надає високу швидкодію при компіляції та зборці 32-розрядних додатків для сучасних операційних систем, включаючи Windows 95 і Windows NT. Програми, створені з використанням C++ Builder, відзначаються високою оптимізацією та ефективністю використання ресурсів пам'яті.

Окрім цього, C++ Builder має добре розроблену систему взаємодії з різними базами даних, включаючи DBASE, Paradox, Sybase, Oracle, Informix, InterBase, Excel, Access, FoxPro, і Btrieve. Використання Borland Database Engine спрощує роботу з базами даних і надає їй велику прозорість.

Засоби управління проектами, двостороння інтеграція і синхронізація між візуальним та текстовим редагуванням, а також вбудований відладчик, роблять C++ Builder потужним інструментом для розробки.

Написання коду в C++ Builder виявилося дуже зручним завдяки інтегрованому візуальному середовищу розробки (IDE). Інтерфейс IDE дозволяє створювати користувацький інтерфейс програм шляхом простого перетягування та розміщення елементів інтерфейсу на вікні форми. Розробники можуть визначити вигляд, поведінку та функціональність елементів інтерфейсу, що дозволяє швидко створювати візуально привабливі додатки.

Один з ключових плюсів Borland C++ Builder - це велика бібліотека компонентів, які дозволяють створювати різноманітні програми з різними функціональностями. Вона включає компоненти для роботи з графікою, мережами, базами даних, аудіо та багато інших сфер. Це дозволяє розробникам швидко і легко додавати різноманітні функції до своїх програм без необхідності писати код з нуля.

Крім цього, Borland C++ Builder відзначається добре розробленою системою відладки. Розробники можуть використовувати вбудований відлагоджувач для пошуку та виправлення помилок в коді. Він дозволяє крокувати через код, встановлювати точки зупинки, аналізувати змінні та багато іншого. Це робить процес відлагодження більш ефективним та допомагає знайти та виправити проблеми в програмі.

Однією з ключових переваг Borland C++ Builder є його підтримка різних операційних систем, включаючи роботу під Windows 95, Windows NT, а також подальші версії Windows. Це дозволяє розробникам створювати додатки, які працюють на широкому спектрі платформ і версій операційних систем.

Підсумовуючи, Borland C++ Builder став популярним інструментом для розробки програмного забезпечення завдяки своєму візуальному середовищу розробки, багатій бібліотеці компонентів, ефективній системі відладки та підтримці різних операційних систем. Цей інструмент став незамінним для багатьох розробників, які працюють над проектами для платформи Windows.

Напрямок розвитку інструментарію для розробки ПЗ продовжує активно розвиватися. Сучасні розробники мають доступ до різноманітних інструментів, серед яких популярні інтегровані середовища розробки, розширені IDE для конкретних мов та фреймворків, а також різноманітні плагіни, що полегшують та розширюють можливості інструментів розробки.

Один із важливих аспектів розширення функціональності інструментарію - це використання плагінів. Плагіни - це додаткові модулі, які можна інтегрувати в інтегроване середовище розробки для розширення його можливостей. Це дозволяє розробникам використовувати лише ті функції, які їм потрібні, та налаштовувати своє середовище розробки під свої потреби.

4.1.2. Середовище IntelliJ IDEA

IntelliJ IDEA - це інтегроване середовище розробки (IDE), створене компанією JetBrains, спеціалізоване на розробці програмного забезпечення на мові програмування Java та інших мовах, які працюють на віртуальній машині Java (JVM). IntelliJ IDEA є однією з найпопулярніших IDE серед розробників Java та користувачів, які займаються розробкою на цій платформі.

Ця IDE пропонує широкий спектр функціональностей, які полегшують процес розробки, включаючи автоматичне доповнення коду, вбудовану систему контролю версій, підтримку різних фреймворків і бібліотек, аналіз коду, налаштування інструменти для роботи з базами даних та багато інших корисних можливостей.

IntelliJ IDEA також має велику кількість плагінів та розширень, які дозволяють налаштовувати середовище розробки під конкретні потреби проекту. Крім того, вона підтримує розробку великих і складних проектів, що робить її ідеальним вибором для підприємств та розробників, які працюють з великим обсягом Java-коду.

IntelliJ IDEA є крос-платформеною IDE, що означає, що ви можете використовувати її на різних операційних системах, включаючи Windows, macOS і

Linux. Її інтерфейс дуже користувач-орієнтований і інтуїтивний, що полегшує роботу з нею для розробників на будь-якому рівні навичок.

Це одна з найпопулярніших інтегрованих середовищ розробки (IDE) для мов програмування Java, а також підтримує багато інших мов, таких як Kotlin, Groovy, Scala, JavaScript, і багато інших. IntelliJ IDEA також підтримує велику кількість різних плагінів, які допомагають розробникам поліпшити їхній досвід роботи з IDE. Нижче наведені діні про деякі з найпопулярніших плагінів для IntelliJ IDEA.

Git Integration: Цей плагін додає підтримку системи контролю версій Git безпосередньо в IDE. Ви можете використовувати його для коміту, відкату, синхронізації та інших операцій, що пов'язані з Git, прямо з IDE.

Maven: Цей плагін надає підтримку для управління проектами, що використовують Apache Maven. Він допомагає імпортувати проекти, створювати залежності, виконувати цільові завдання і багато інших операцій, пов'язаних з Maven.

Gradle: Як і плагін Maven, цей плагін надає підтримку для системи збірки і управління залежностями Gradle. Ви можете працювати з Gradle-проектами, налаштовувати завдання, виконувати збірку і інші операції з IDE.

Spring Framework Support: Цей плагін спрощує розробку додатків на базі Spring Framework. Він надає підказки коду, автозаповнення конфігурацій Spring, інтеграцію з Spring Boot і багато інших корисних функцій.

Kotlin: Якщо ви працюєте з мовою програмування Kotlin, цей плагін допоможе вам покращити роботу з IDE, надаючи підсвічування синтаксису, автозаповнення і інші функції, специфічні для Kotlin.

Tomcat and Jetty Integration: Ці плагіни дозволяють запускати та налагоджувати веб-додатки на вбудованих веб-серверах, таких як Apache Tomcat і Eclipse Jetty, безпосередньо з IDE.

Code Quality and Analysis Tools: Інші популярні плагіни включають підтримку для різноманітних інструментів для аналізу коду, таких як FindBugs,

Checkstyle, PMD, і інших. Вони допомагають підвищити якість вашого коду та знайти потенційні проблеми.

Database Tools: IntelliJ IDEA також має плагіни для роботи з базами даних, такі як MySQL, PostgreSQL, Oracle та інші. Вони дозволяють підключатися до баз даних, виконувати запити SQL, відлагоджувати SQL-код і багато інших операцій.

Lombok: Цей плагін підтримує аннотації Lombok, які допомагають скоротити кількість бойлерплейту коду в Java-проектах.

Key Promoter X: Цей плагін призначений для навчання користувачів гарячим клавішам в IntelliJ IDEA. Він показує сповіщення з реком

JUnit: Цей плагін надає підтримку для тестування з використанням JUnit. Він дозволяє створювати, виконувати та аналізувати тести прямо з IDE.

Docker Integration: Якщо ви працюєте з Docker-контейнерами, цей плагін надає інтеграцію для створення, запуску та керування Docker-контейнерами безпосередньо з IntelliJ IDEA.

SonarLint: Цей плагін дозволяє використовувати SonarQube для аналізу якості вашого коду. Він надає звіти та поради щодо поліпшення якості вашого проекту.

Liquibase Integration: Даний плагін допомагає вам керувати міграціями баз даних з використанням Liquibase, надаючи інтегрований інтерфейс для визначення та виконання міграцій.

CamelCase Plugin: Цей плагін допомагає автоматично перетворювати тексти на CamelCase або інші стилі запису, що корисно при роботі зі змінними, методами та ідентифікаторами.

Rainbow Brackets: Цей плагін розфарбовує дужки та інші парні символи різними кольорами, що полегшує розуміння вкладених структур в коді.

CodeGlance: Даний плагін додає міні-карту коду на бічну панель, що дозволяє швидко переміщатися по великому файлу та відслідковувати місцезнаходження в ньому.

String Manipulation: Цей плагін надає різні корисні операції для обробки рядків, такі як перетворення регістрів, видалення пробілів, кодування та декодування рядків і багато інших.

Grep Console: Цей плагін дозволяє налаштовувати вивід консолі в IntelliJ IDEA за допомогою регулярних виразів, що полегшує пошук та аналіз логів.

FindBugs-IDEA: Даний плагін надає підтримку для інструменту FindBugs, який аналізує Java-код на виявлення потенційних помилок і проблем.

Це лише невелика частина доступних плагінів для IntelliJ IDEA, існують десятки і сотні інших, які вас можуть зацікавити в залежності від ваших потреб та проекту. Встановлювати плагіни можна прямо з магазину плагінів IntelliJ IDEA або завантажити їх з офіційного сайту розробників.

4.1.3. Apache Maven

Для збірки проекту у цьому я використовую Apache Maven.

Apache Maven - це інструмент для управління проектами та залежностями в області розробки програмного забезпечення. Він забезпечує автоматизований процес збірки, залежностей та публікації програмного коду. Apache Maven став популярним інструментом у світі розробки програмного забезпечення завдяки своїм перевагам:

Він вимагає, щоб проекти мали специфічну структуру каталогів і файлів, що полегшує роботу з ними та робить їх більш передбачуваними.

Maven автоматично виконує збірку проекту, включаючи компіляцію, створення JAR-файлів, генерацію документації, виконання тестів тощо.

Maven дозволяє визначити залежності проекту і автоматично завантажувати необхідні бібліотеки з централізованих репозиторіїв або локальних ресурсів.

Підтримує версіювання, дозволяючи встановити конкретну версію бібліотеки або плагіну.

Підтримує різноманітні плагіни для виконання різних завдань, такі як збірка, тестування, деплоймент на сервер тощо. Користувачі можуть створювати свої власні плагіни.

Підтримує різні типи проектів, включаючи Java, C++, .NET, веб-додатки, Android, тощо.

Maven може інтегруватися з іншими популярними інструментами розробки, такими як Eclipse, IntelliJ IDEA, Jenkins тощо.

Загалом, Apache Maven робить процес розробки програмного забезпечення більш організованим і ефективним, допомагаючи розробникам керувати проектами та їх залежностями.

Це важливий інструмент для розробників програмного забезпечення, що впливає на всі аспекти процесу створення програм та забезпечує їх надійність та ефективність. Використання Maven дозволяє стандартизувати структуру проекту, спростити збірку, додати потрібні бібліотеки, ефективно виконувати тести та публікувати результати роботи.

Завдяки Maven, розробники можуть керувати залежностями проекту, що робить процес розробки більш передбачуваним та менш вразливим до помилок. Це також допомагає встановити конкретну версію бібліотеки або плагіну, що особливо корисно для забезпечення сумісності та стабільності проекту.

Maven підтримує широкий спектр плагінів, які дозволяють виконувати різноманітні завдання, від збірки та тестування до деплойменту на сервер. Розробники також можуть створювати власні плагіни, розширюючи функціональність Maven для конкретних потреб їх проекту.

Незалежно від типу проекту - чи це Java, C++, .NET, веб-додаток або мобільний додаток - Maven надає засоби для ефективного управління ним. Важливою особливістю Maven є його можливість інтеграції з іншими інструментами розробки, такими як Eclipse, IntelliJ IDEA та Jenkins, що полегшує використання Maven у різних екосистемах.

Загалом, використання Apache Maven робить розробку програмного забезпечення більш організованою та продуктивною, надаючи розробникам потужний інструмент для керування проектами та їх залежностями.

4.1.4. Microsoft SQL Server

SQL Server є продуктом від Microsoft, який використовується для керування та роботи з базами даних. Ця реляційна система управління базами даних спроектована для зберігання та обробки даних в табличному форматі, де дані представлені у вигляді рядків та стовпців. Вона підтримує реляційну модель, дозволяючи створювати складні зв'язки між даними в різних таблицях.

SQL Server надає можливості високої доступності для забезпечення доступності даних навіть під час відмов та аварій. До цих можливостей входять реплікація, кластеризація та дзеркальна реплікація. SQL Server також включає в себе засоби безпеки, такі як автентифікація та авторизація, для захисту даних від несанкціонованого доступу.

Ця система підтримує мову запитів SQL для взаємодії з базами даних, дозволяючи створювати, зчитувати, оновлювати та видаляти дані. Вона також надає інструменти для оптимізації запитів, підвищення продуктивності та взаємодії з іншими продуктами Microsoft, такими як Azure, Power BI та інші.

Загалом, SQL Server використовується для управління та обробки даних у різних галузях, включаючи корпоративний бізнес, веб-розробку, аналіз даних та інше. Цей продукт надає широкі можливості для розробки та управління базами даних без використання пунктів.

Основні функції SQL Server включають:

- Зберігання даних: SQL Server використовуєся для створення баз даних, в яких можна зберігати структуровані дані. Ці дані

представлені у вигляді таблиць з рядками та стовпцями, що дозволяє легко організувати та зберігати інформацію.

- Мова запитів SQL: SQL Server підтримує мову запитів SQL (Structured Query Language), що дозволяє виконувати різноманітні операції з даними. За допомогою SQL можна створювати, зчитувати, оновлювати та видаляти дані з бази даних.
- Безпека даних: SQL Server надає різні механізми для захисту даних. Вони включають автентифікацію та авторизацію, шифрування даних, аудит доступу та інші засоби безпеки.
- Вища доступність: SQL Server пропонує можливості для забезпечення вищої доступності даних за допомогою різних технологій, таких як реплікація, кластеризація та дзеркальна реплікація.
- Оптимізація запитів: SQL Server включає оптимізатор запитів, який допомагає покращити продуктивність та швидкість виконання запитів до бази даних.
- Інтеграція: SQL Server легко інтегрується з іншими продуктами Microsoft, такими як Microsoft Azure, Power BI, SharePoint та інші.
- Резервне копіювання та відновлення: SQL Server надає засоби для створення резервних копій баз даних та відновлення даних у випадку аварійного втрати інформації.
- Автоматизація завдань: За допомогою SQL Server Agent можна автоматизувати запуск завдань та розподіл їх на різні компоненти системи.
- Засоби аналізу даних: SQL Server має можливості для аналізу даних та створення звітів, що допомагає в здійсненні бізнес-аналізу.
- Підтримка мов програмування: Ви також можете використовувати різні мови програмування для розробки додатків, які взаємодіють з SQL Server, включаючи C#, Java, Python та інші.

Загалом, SQL Server є потужною системою управління базами даних, яка надає багато функцій та засобів для ефективної роботи з даними в різних галузях і діяльностях без використання пунктів.

Загальна структура програмного забезпечення

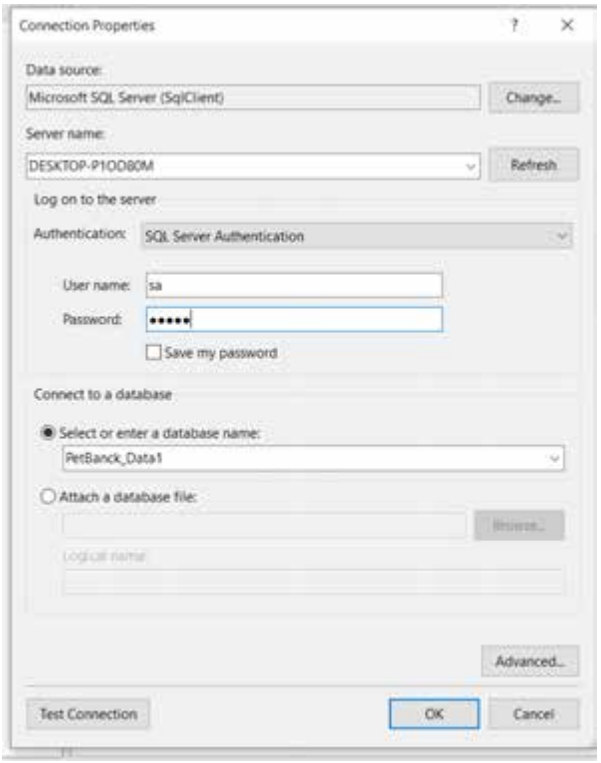
З точки зору задач, які необхідно виконати, програмне забезпечення можна поділити на дві частини:

1) програмне забезпечення, що забезпечує систему онлайн банкінгу;
2) аналіз даних, програмне забеспечення аналітичного блоку.

4.2.1 Задачі аналізу даних онлайн банкінгу

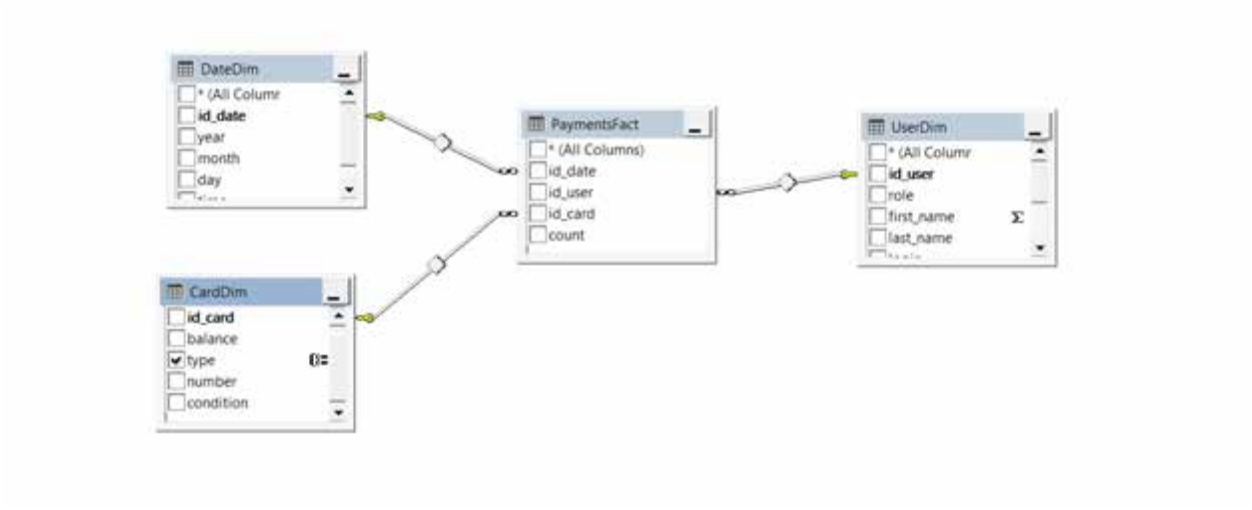
Задачі аналізу даних можна виконати використовуючи OLAP технології, що ґрунтуються на збережені даних у СД.

Використовуючи службу репорті будуємо звіт для свого проекту для цього потрібно виконати наступні етапи:



Створюємо джерело даних – СД

За допомогою Майстра звітів створимо звіт, що представляє дані за типами боржників та їх середній борг в розрізі підприємств, що надають послуги в різних містах.



Отримуємо звіт по типу карток у вигляді секторної діаграми.

Проаналізувавши дану статистику ми можемо бачити який тип картки є найбільш

популярним також для більшої наочності приведена секторна діаграма.
З цих даних випливає що картка типу Visa є найбільш популярною на другому місці MasterCard і на третьому Amex. Дана інформація допоможе менеджерам формувати нові пропозиції для користувачів випливаючи з того що пропозиції представлені в карті Visa на сьогоднішній день є найбільш релевантними для користувачів Це допоможе сформувати нові пропозиції які будуть ще вигіднішими

Кількість платежів в розрізі часу і країни

	Україна	Польща	Литва	Іспанія
Січень 2023	243	143	165	56
Лютий 2023	556	124	134	72
Березень 2023	438	213	134	65
Квітень 2023	434	211	96	67
Травень 2023	532	132	98	43
Червень 2023	342	123	110	76
Липень 2023	642	272	87	56
Серпень 2023	344	213	87	72
Вересень 2023	765	128	120	53
Жовтень 2023	986	195	100	52
Листопад 2023	-	-	-	-
Грудень 2023	-	-	-	-

Також ми генеруємо другий звіт який допоможе нам побачити з якої країни проведено найбільша кількість платежів ми маємо чотири колонки: Україна Польща Литва та Іспанія а також місяці протягом 2023 року. Із цієї статистики ми бачимо скільки платежів було проведено у кожній локації кожного місяця та

проаналізувавши дану статистику можна зробити висновок що Україна є найбільш популярною локацією Польща та Литва ділять між собою друге місце та на третьому місці Іспанія в якій нещодавно відкрився наш банк.

Цих два статистичних представлення допоможуть аналітику та менеджменту компанії банку зробити правильні Висновки та сформувати нові пропозиції у вигляді карток бонусних надбавок які будуть корисними для користувачів що в свою чергу приведе до більшої популярності банку та ефективності його роботи.

ВИСНОВКИ

У процесі виконання цієї роботи на тему “Система аналізу цифрових платежів” було досліджено і проаналізовано систему аналітики онлайн бакінгу. Були застосовано технології OLAP та Data Mining, що дало змогу створити аналітичний модуль який надає можливість менеджеру і аналітику формувати нові пропозиції для клієнтів на основі статистики у розрізі країни і часу. Створено програмне забезпечення моніторингу параметрів мікроклімату тепличного комплексу.

Отримали систему аналізу. Ця система забезпечує своєчасне отримання статистики, а також можливість отримання інформації в реальному часі та за певний часовий період.

Отже, в результаті розробки магістерської роботи отримали систему аналітики, шляхом побудови сховища даних та розгорнутання куба, а також аналізу роботи онлайн банку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Теоретичні відомості про системи підтримки прийняття рішень [Електронний ресурс] – Режим доступу:
http://www.studyport.net/lib/sistemi-pidtrimki-prijnyattya-rishen/

2. Плескач В.Л. Інформаційні системи і технології на підприємствах / Плескач В.Л., Затонацька Т.Г. – Київ: 2011.

3. Ситник В.Ф. Інформаційні системи і технології / Ситник В.Ф. – Київ: 2002.

4. Кельдер Т.Л. Інформаційні системи та технології / Кельдер Т.Л – ЗДУ, 2002.

5. Студенткий портал. І нформаційні системи і технології обліку [Електронний ресурс] – Режим доступу до ресурсу:
http://studbase.com/manuals/12/12/

6. Теорія економічних інформаційних систем [Електронний ресурс] – Режим доступу до ресурсу:
http://www.kgau.ru/istiki/teis/index.html

7. Пасічник В.В. Організація баз даних та знань / Пасічник В.В., Резніченко В.А. – К.: Видавнича група BHV, 2006.

8. Д.О. Ладичук Бази даних геоінформаційних систем. Навчальний посібник / Д.О. Ладичук, В.І. Пічура – Херсон, 2006.

9. Н. Б. Шаховська. Сховища даних / Шаховська Н. Б., Пасічник В. В. – Видавництво «Магнолія 2006», Львів – 2008.

10. EasyCode. Програмування, легко про складне [Електронний ресурс] – Режим доступу до ресурсу:
http://easy-code.com.ua/2012/08/redaguvannya-data-mining-modelej-komerciya-rizne-statti/

11. В.В. Войтенко. C/C++ : теорія та практика / В.В. Войтенко, А.В.Морозов – Житомир, 2003.

12. Корпорація Microsoft [Електронний ресурс] – Режим доступу до ресурсу:

https://www.microsoft.com/Ukraine/News/Issues/2004/08/SQLServer.mspx

13. FastReport для C++Builder 1-6 [Електронний ресурс] – Режим доступу до ресурсу:

http://gedemin.googlecode.com/svn/trunk/Gedemin/FastReport/README.UKR

14. Т.А.Павловская. C/C++. Программирование на языке високого уровня / Т.А.Павловская. – Питер: 2002.

15. Полубояров В.В. Використання MS SQL Server Analysis Services 2008 для побудови сховищ даних / Полубояров В.В – М: 2008

16. Закон України «Про охорону праці» [Електронний ресурс] − Режим доступу:

http://zakon4.rada.gov.ua/laws/show/2694-12

17. Жидецький В.Ц. Основи охорони праці / Жидецький В.Ц. – Львів: Афіша, 2002.

18. Запорожець О.І. Основи охорони праці / Запорожець О.І.,Протоєрейський О.С.,Франчук Г.М.,Боровик І.М. – К.:Центр учбової літератури,2009.

ДОДАТОК А

Сторінок - 2

Код створення сховища даних

```
USE PetBanck_Data1
GO

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'UserDim' AND type_desc='USER_TABLE')

DROP TABLE UserDim
GO
CREATE TABLE UserDim (
id_user INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[role] NVARCHAR(20) NOT NULL,
[first_name] NVARCHAR(20) NOT NULL,
[last_name] NVARCHAR(20) NOT NULL,
[login] NVARCHAR(20) NOT NULL,
[password] NVARCHAR(20) NOT NULL,
[condition] NVARCHAR(20) NOT NULL,
[country] NVARCHAR(20) NOT NULL
);

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'CardDim' AND type_desc='USER_TABLE')

DROP TABLE CardDim
GO
CREATE TABLE CardDim (
id_card INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[balance] DECIMAL(10,5) NOT NULL,
[type] NVARCHAR(20) NOT NULL,
[number] NVARCHAR(20) NOT NULL,
[condition] NVARCHAR(20) NOT NULL
);

IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'DateDim' AND type_desc='USER_TABLE')

DROP TABLE DateDim
GO
CREATE TABLE DateDim (
id_date INT IDENTITY(1,1) NOT NULL PRIMARY KEY ,
[year] CHAR(4) NOT NULL,
[month] NVARCHAR(20) NOT NULL,
[day] NVARCHAR(20) NOT NULL,
[time] TIME NOT NULL
);
```

```
IF EXISTS (SELECT name FROM sys.objects

WHERE name = 'PaymentsFact' AND type_desc='USER_TABLE')

DROP TABLE PaymentsFact
GO
CREATE TABLE PaymentsFact (
id_date INT NOT NULL,
id_user INT NOT NULL,
id_card INT NOT NULL,
[count] DECIMAL(10,5) NOT NULL,
FOREIGN KEY (id_date) REFERENCES DateDim(id_date) ON DELETE CASCADE,
FOREIGN KEY (id_user) REFERENCES UserDim(id_user) ON DELETE CASCADE,
FOREIGN KEY (id_card) REFERENCES CardDim(id_card) ON DELETE CASCADE
);
GO
```

```
USE PetBanck_Data1

INSERT INTO UserDim ([role], [first_name], [last_name], [login], [password], [condition],
[country])
VALUES
('Admin', 'John', 'Doe', 'johndoe', 'password1', 'Active', 'USA'),
('User', 'Jane', 'Doe', 'janedoe', 'password2', 'Active', 'Canada'),
('User', 'Bob', 'Smith', 'bobsmith', 'password3', 'Inactive', 'Australia');

-- Insert test data into CardDim table
INSERT INTO CardDim ([balance], [type], [number], [condition])
VALUES
(1000.00, 'Visa', '1111-2222-3333-4444', 'Active'),
(500.00, 'Mastercard', '5555-6666-7777-8888', 'Inactive'),
(250.00, 'Amex', '9999-8888-7777-6666', 'Active');

-- Insert test data into DateDim table
INSERT INTO DateDim ([year], [month], [day], [time])
VALUES
('2023', 'March', '06', '08:30:00'),
('2023', 'February', '14', '14:00:00'),
('2022', 'December', '25', '10:30:00');

-- Insert test data into PaymentsFact table
INSERT INTO PaymentsFact (id_date, id_user, id_card, [count])
VALUES
(1, 1, 1, 100.00),
(2, 2, 2, 50.00),
(3, 3, 1, 25.00),
(1, 2, 1, 200.00),
(2, 1, 2, 75.00),
(3, 3, 3, 100.00);
```