МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра Комп’ютерних систем та мереж

СТУДЕНТА КС-131М

Тема: Інйформаційна система підприємтсва на основі об'єктно-орієнтованих баз даних

Виконавець: Владислав ШКУТ

Керівник: Микола ПРОЦЕНКО

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж
Напрям (спеціальність) 123 "Комп'ютерна інженерія"
(шифр, найменування)

З А В Д А Н Н Я
дипломної роботи

Шкута Владислава Анатолійовича
(прізвище, ім'я, по батькові)

1. Тема роботи: Інформаційна система підприємства на основі об'єктно-орієнтованих баз даних
2. Термін виконання проекту (роботи):): з 12.06.2023 до 30.06.2023
3. Вихідні дані до роботи: інформаційна система підприємства на основі об'єктно-орієнтованих баз даних
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) принципи побудови ООБД
2) аналіз використання ООБД в сучасних інформаційних системах
3) формування моделі ООБД
4) висновки

Календарний план

№ п/п	Етапи виконання курсового проєкту	Термін виконання етапів	Примітка
1	Провести аналіз літератури за темою курсового проєкту та аналіз існуючих ООБД, вивчити технічну документацію	27.06.2023	Виконано
2	Проаналізувати архітектуру ООБД, порівняти переваги/недоліки з реляційними моделями	28.06.2023	Виконано
3	Формування моделі ООБД, аналіз технологій для реалізації створеної моделі	29.06.2023	Виконано
4	Оформити звіт з науково-дослідницької практики та супроводжувальну документацію	30.06.2023	Виконано

Дата отримання завдання «27» червня 2023 р.

Керівник науково-дослідної практики _____ Микола ПРОЦЕНКО
(підпис)

Завдання прийняв до виконання _____ Владислав ШКУТ
(підпис студента)

ЗМІСТ

ВСТУП

Іформаційні системи відіграють важливу роль у підприємствах різних галузей, забезпечуючи ефективне управління інформацією та підтримку процесів прийняття рішень. Вони дозволяють підприємствам збирати, зберігати, організовувати та управляти великим обсягом даних. Це включає дані про клієнтів, інформацію про продукти, фінансові записи, дані про запаси та багато іншого. Централізація даних та забезпечення їх точності та доступності надають надійну основу для ефективних бізнес-операцій.

Інформаційні системи автоматизують та оптимізують бізнес-процеси, збільшуючи ефективність та зменшуючи ручні зусилля. Вони допомагають оптимізувати такі дії, як управління записами, обробка замовлень, управління ланцюгом постачання, кадровий облік та фінансові транзакції. Це призводить до покращення продуктивності, зниження витрат та швидкого реагування.

Використання баз даних в інформаційних системах є надзвичайно важливим для ефективного управління та організації великих обсягів даних. Бази даних слугують централізованими сховищами, де інформація зберігається та організовується у структурований формат, що дозволяє легко отримувати, змінювати та використовувати дані. Використання баз даних у інформаційних системах надає наступні переваги:

В першу чергу, бази даних забезпечують структурований підхід до зберігання та організації даних. Вони використовують таблиці з рядками та стовпцями, де різні типи даних зберігаються у відповідних полях. Це дозволяє ефективно керувати та зберігати дані, забезпечуючи їх цілісність та доступність.

По-друге, бази даних сприяють інтеграції даних шляхом поєднання даних з різних джерел у єдину систему. Вони надають засоби для встановлення зв'язків та відношень між різними таблицями. Це дозволяє об'єднувати дані з різних джерел та отримувати повний обсяг інформації для подальшого аналізу та використання.

Третя перевага полягає в потужних можливостях запиту до даних. Бази даних

надають мови запитів, які дозволяють вилучати потрібні дані з бази. Запити дозволяють вибирати конкретні записи за заданими критеріями, виконувати обчислення та здійснювати різні операції з даними. Це надає організаціям можливість ефективно отримувати необхідну інформацію.

Крім того, бази даних забезпечують безпеку даних та контроль доступу. Вони надають механізми автентифікації, авторизації та контролю доступу користувачів, дозволяючи адміністраторам визначати та керувати правами та обмеженнями користувачів. Техніки шифрування можуть бути застосовані для забезпечення конфіденційності та цілісності даних під час передачі та зберігання, забезпечуючи безпеку даних.

Вони також гаранутють цілісність та послідовність даних за допомогою правил валідації та обмежень. Ці правила визначають дозволені значення, відношення та залежності в межах бази даних, забезпечуючи точність, надійність та уникнення неузгодженостей даних. Збереження цілісності даних дозволяє інформаційним системам надавати надійну та достовірну інформацію користувачам.

В сучасному інформаційному суспільстві, де обсяги даних зростають експоненційно, питання ефективного зберігання, обробки та управління інформацією стає важливим завданням для багатьох галузей діяльності. Актуальність теми "Інформаційні системи на основі об'єктно-орієнтованих баз даних" необхідно розглядати в контексті швидкого розвитку технологій та потреби у нових методах обробки даних.

Зростаюча кількість інформації, яку генерують підприємства, наукові установи, та суспільство в цілому, потребує більш ефективних та гнучких інструментів для її управління. В цьому контексті об'єктно-орієнтовані бази даних (ООБД) виступають як потенційно перспективний інструмент для оптимізації роботи з даними.

Об'єм структурованих та неструктурованих даних, які генеруються щодня, зростає на зараз вражаючими темпами. Великі корпорації, соціальні мережі, датчики та інші джерела інформації надають великі потоки даних, що потребують ефективного управління та обробки. За останні роки споживачі даних стали

вимагати не тільки більшого обсягу інформації, але й швидкості її обробки. В інтерактивних додатках, фінансовому секторі, та інших областях, де важливий аспект реального часу, виникає потреба у системах, які можуть надавати дані миттєво.

Розвиток сучасних алгоритмів штучного інтелекту та машинного навчання вимагає доступу до великих обсягів даних для навчання та покращення алгоритмів. Об'єктно-орієнтовані бази даних можуть надати певні переваги у забезпеченні доступу та управлінні необхідними даними для цих систем. Сучасні дані стають все більше комплексними та взаємопов'язаними. Замість простих табличних структур, їхні взаємозв'язки та складні структури стають нормою. ООБД можуть краще справлятися з такими складними відносинами між даними.

Однією з важливих складових сучасних інформаційних систем є забезпечення безпеки даних. ООБД надають можливості для реалізації різних рівнів безпеки та контролю доступу до інформації.Також, вони надають гнучкість в обробці різноманітних типів даних. Здатність працювати як зі структурованими, так і з неструктурованими даними робить їх ефективними для застосувань, де потрібно зберігати та обробляти різнорідні дані.

ООБД дозволяють легко масштабувати систему при зростанні обсягу даних. Це особливо важливо в умовах, коли необхідно обробляти великі обсяги інформації, такі як у великих корпораціях чи обчислювальних центрах. Можна виділити також те, що вони дозволяють зберігати дані в їхньому природному об'єктному контексті, що полегшує розуміння взаємозв'язків між різними елементами даних. Це може бути важливим для семантичного аналізу та використання в додатках штучного інтелекту.

За врахуванням сучасного зростання обчислювальної потужності, використання ООБД стає більш доступним та ефективним. Це розкриває нові можливості для розробки складних інформаційних систем та використання розподілених обчислювань. У світі, де компанії та організації розташовані глобально, здатність працювати в розподіленому середовищі стає ключовою. Це дозволяє забезпечити доступ до даних з будь-якої точки світу та забезпечує їхню

надійність.

Об'єктно-орієнтовані бази даних мають вбудовані механізми для забезпечення цілісності даних. Це стає важливим в аплікаціях, де точність та надійність інформації мають критичне значення, таких як системи управління базами клієнтів чи фінансові системи.

ООБД активно інтегруються з іншими передовими технологіями, такими як мови програмування, фреймворки для розробки програмного забезпечення та інші компоненти інформаційних систем, що дозволяє використовувати їх у різноманітних індустріях. Є можливість ефективно взаємодіяти з іншими типами баз даних. Це дає можливість використовувати їх в комплексі з реляційними, NoSQL та іншими системами зберігання даних для оптимізації певних завдань.

Урахування цих аспектів дозволяє визначити ООБД як потужний інструмент для розв'язання сучасних завдань управління даними в різних сферах від бізнесу та адміністрування до науки та технологій. В подальших розділах роботи буде проведений глибокий аналіз кожного з наведених аспектів з метою визначення практичної застосованості та ефективності використання ООБД у різних сценаріях.

Наукова новизна даної роботи полягає в глибокому дослідженні та систематичному аналізі ролі об'єктно-орієнтованих баз даних в контексті сучасних інформаційних систем. Це дослідження не лише просто висвітлює загальні теоретичні аспекти, але і вдавало вивчає їхні унікальні риси та переваги порівняно з іншими типами баз даних, особливо з реляційними.

Теоретична основа дослідження глибше розкриває принципи та концепції, на яких ґрунтуються об'єктно-орієнтовані бази даних. Акцент робиться на їхньому об'єктно-орієнтованому підході до обробки та організації даних, що віддзеркалюється у використанні концепцій об'єктно-орієнтованого програмування. Це розуміння є важливим для поглибленого аналізу можливостей, які ООБД можуть принести у контексті сучасних інформаційних технологій.

Порівняльний аналіз є важливим аспектом наукової новизни. Розглядання ООБД у контексті порівняння з реляційними базами даних виходить за межі традиційного підходу та дозволяє визначити конкретні сценарії, де ООБД можуть

виявити свою перевагу. Це розширює загальне розуміння використання баз даних в сучасному інформаційному середовищі та визначає ніші для ефективного впровадження ООБД.

Практична орієнтованість дослідження полягає у вивченні конкретних застосувань ООБД в різних галузях та обставинах. Детальний аналіз практичних випадків використання дозволяє виявити їхню реальну ефективність та спростити перехід від теорії до практики. Врахування контексту різних галузей, таких як бізнес, наука чи технології, дозволяє сформулювати конкретні рекомендації щодо впровадження такої технології в різних випадках використання.

Це дослідження сприяє формуванню нового підходу до використання ООБД в інформаційних системах, базуючись не лише на загальній теорії, але і на конкретних даних та практичних сценаріях. Такий підхід є ключовим у контексті стрімкого розвитку сучасних технологій та зростання обсягів даних, де важливо не тільки розуміти принципи, але і здатність ефективно впроваджувати їх у реальних умовах.

Ця робота є значущою для розуміння, як об'єктно-орієнтовані бази даних можуть бути використані у конкретних інформаційних системах, зокрема, у системах управління відносинами з клієнтами (CRM). Поринувши глибше у практичний вимір використання ООБД, особливий акцент буде зроблено на розробці CRM, яка використовує ці бази даних.

Їхнє використання у CRM дозволяє забезпечити більш гнучку та ефективну обробку складних відносин між клієнтами та компанією. Це досягається завдяки об'єктно-орієнтованому підходу, де дані представлені у вигляді об'єктів, що відображають реальні об'єкти або концепції, що спрощує взаємодію з іншими об'єктами та робить систему більш гнучкою до змін.

Важливим аспектом є здатність ООБД зберігати дані в їхньому природному об'єктному контексті. У CRM це може вказувати на можливість більш точно відтворювати взаємозв'язки та інтеракції між клієнтами, їхніми покупками, запитаннями та іншими факторами. Це сприяє створенню більш деталізованих та контекстуально обґрунтованих профілів клієнтів.

Вони також вирішують питання безпеки та цілісності даних у системах CRM.

Механізми вбудованої безпеки дозволяють контролювати доступ до об'єктів даних, забезпечуючи конфіденційність та управління правами користувачів. Це важливо, особливо при роботі з конфіденційною інформацією про клієнтів.

У розробці такої системи на основі об'єктно-орієнтованої бази даних важливо враховувати їхню гнучкість в обробці різноманітних типів даних. Масштабованість дозволяє ефективно взаємодіяти з ростом обсягів інформації, що особливо актуально в CRM, де обробка великої кількості клієнтської інформації є ключовим аспектом.

Інтеграція ООБД з іншими технологіями та системами управління даними ускладнює розширення функціоналу CRM. Сумісність із сучасними технологіями та можливість інтеграції з іншими базами даних дозволяє створювати повноцінні та високопродуктивні інформаційні екосистеми.

Отже, використання об'єктно-орієнтованих баз даних в системах CRM визначається не лише їхнім теоретичним потенціалом, але і конкретними перевагами в практичних застосуваннях. Це дозволяє створювати більш гнучкі, ефективні та надійні CRM-системи, що відповідають сучасним вимогам управління відносинами з клієнтами.

З урахуванням цих факторів, можна визначити, що тема "Інформаційні системи на основі об'єктно-орієнтованих баз даних" має величезний потенціал для вирішення актуальних проблем сучасності та впровадження більш ефективних та гнучких систем управління даними. У подальших розділах роботи буде детально розглянуто та проаналізовано переваги та особливості використання об'єктно-орієнтованих баз даних для розв'язання цих завдань.

РОЗДІЛ 1

РОЛЬ ТА ЗНАЧЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ ПІДПРИЄМСТВ

Інформаційні системи в сучасному бізнесі відіграють ключову роль, ставши необхідним елементом успішного функціонування підприємств. Розуміння та ефективне управління інформацією стали стратегічно важливими завданнями в умовах загостреної конкуренції та стрімкого технологічного розвитку. В цьому контексті системи управління відносинами з клієнтами (CRM) набули особливої ваги для підприємств.

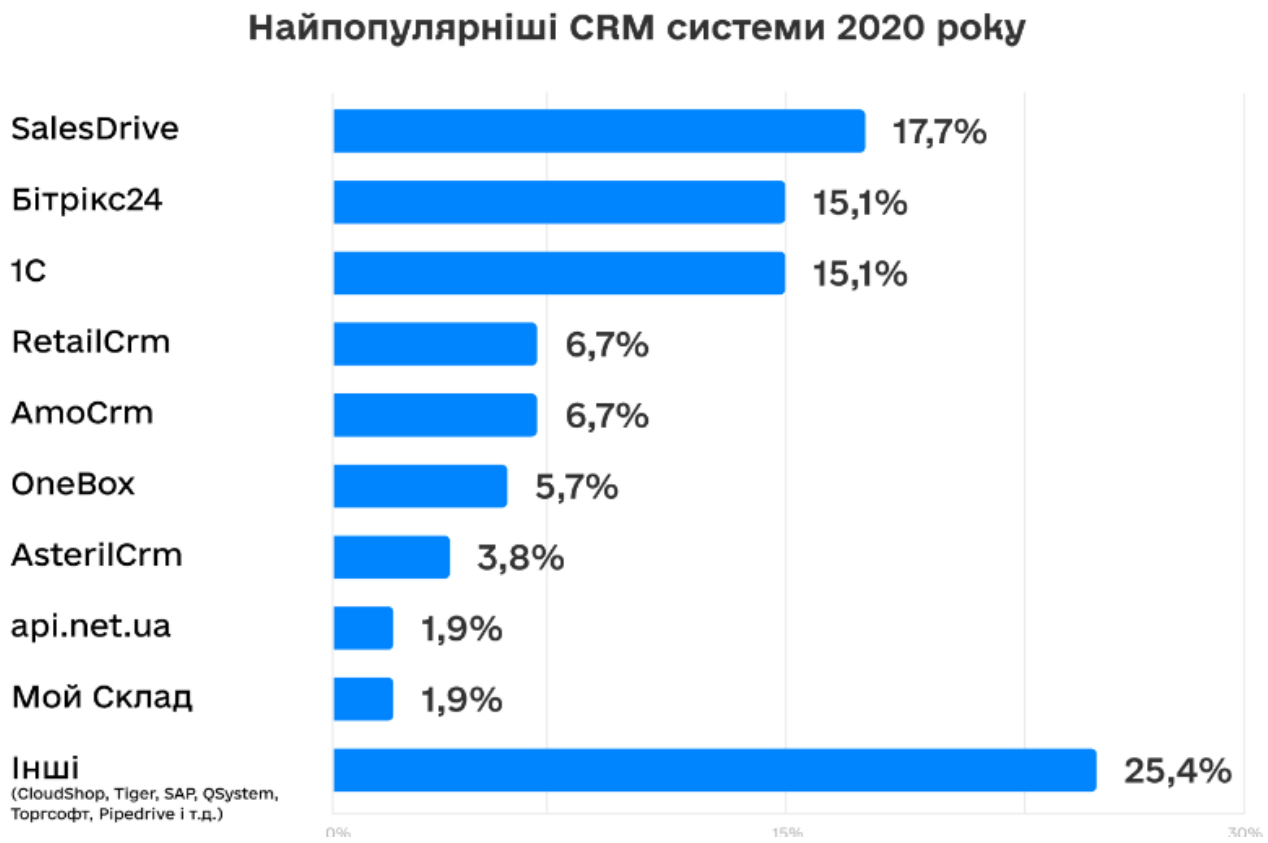
Роль інформаційних систем в підприємстві полягає в тому, щоб забезпечити ефективну обробку, збереження та передачу інформації. Це включає в себе автоматизацію багатьох бізнес-процесів, що полегшує рутинні завдання, зменшує ймовірність помилок та сприяє швидкій прийняттю стратегічних рішень. Інформаційні системи також допомагають підприємствам адаптуватися до змін в оточуючому середовищі, швидко реагувати на нові вимоги ринку та вдосконалювати свою конкурентоспроможність.

Функції які виконують інформаційні системи, в основному це:

- Інформаційні системи створюють ефективні механізми для збору, обробки та зберігання великого обсягу даних, що забезпечує відповідність бізнес-процесів та сприяє прийняттю обґрунтованих стратегічних рішень.
- ІС автоматизують та оптимізують бізнес-процеси, зменшуючи час та витрати на виконання завдань, що поліпшує продуктивність та конкурентоспроможність підприємства. ІС автоматизують та оптимізують бізнес-процеси, зменшуючи час та витрати на виконання завдань, що поліпшує продуктивність та конкурентоспроможність підприємства.
- Завдяки ІС, підприємства можуть здійснювати більш ефективний контроль над своєю діяльністю, мінімізуючи ризики помилок та забезпечуючи точність обліку

даних.

- Інформаційні системи сприяють розвитку електронного бізнесу, забезпечуючи можливість віртуальної торгівлі, електронного маркетингу та збільшення доступності продуктів та послуг.
- Системи управління взаємодією з клієнтами (CRM) дозволяють підприємствам ефективно взаємодіяти з клієнтами, визначаючи їхні потреби та покращуючи якість обслуговування.



Фактори, які сприяли розвитку інформаційних систем, включають рост комп'ютерних технологій, збільшення обсягу даних, глобалізацію бізнесу, розширення систем мережевого зв'язку та зростання потреб у бізнес-рішеннях. Розвиток технологій надав можливість створення та ефективне використання інформаційних систем. Зростання обсягу даних викликало необхідність в системах, що можуть ефективно їх обробляти. Глобалізація бізнесу змусила підприємства використовувати інформаційні системи для оптимізації процесів та взаємодії на

світовому ринку. Розвиток та доступність мережевих технологій сприяли створенню розподілених систем та поліпшили обмін інформацією між підприємствами. Зростання складності завдань бізнесу створило потребу в ефективних інформаційних рішеннях.

У цьому контексті CRM виступає як важливий елемент інформаційної інфраструктури підприємства. CRM спрямована на ефективне управління відносинами з клієнтами, що є стратегічно важливим для будь-якого бізнесу. Системи CRM дозволяють збирати, аналізувати та використовувати дані про клієнтів, щоб створювати персоналізовані підходи та покращувати якість обслуговування. Це сприяє збільшенню лояльності клієнтів, залученню нових та збільшенню прибутковості.

Основне значення CRM для підприємств полягає в тому, що вона допомагає зрозуміти та задовольняти потреби клієнтів на більш осмислений спосіб. За допомогою CRM вдається створити повноцінний профіль кожного клієнта, враховуючи його історію покупок, взаємодій та інші ключові аспекти. Це важливо не лише для ефективного маркетингу та продажів, але і для розробки стратегій розвитку, визначення нових можливостей та вдосконалення продуктів або послуг.

Розвиток інформаційних систем пройшов шлях від простих механізмів обліку до потужних та інтегрованих комплексів, що перетворили спосіб функціонування підприємств та організацій. Початково інформаційні системи використовувались для автоматизації облікових завдань та зберігання обмеженої кількості даних. З часом, з появою персональних комп'ютерів, системи стали більш розповсюдженими та доступними для різних користувачів.

Важливим етапом в розвитку інформаційних систем стала переінженерія бізнес-процесів в середині підприємств. Цей підхід дозволив зрозуміти, які саме завдання можна автоматизувати, а також визначити оптимальний спосіб впровадження технологій для покращення ефективності та контролю.

Поступово з'явилися інтегровані підходи, де інформаційні системи об'єднували різні аспекти бізнесу, такі як фінанси, логістика, та управління персоналом. Це дозволило забезпечити єдність інформаційного середовища та

легше керувати комплексними бізнес-процесами.

Одним із ключових напрямків в сучасному розвитку є перехід до хмарних технологій. Це дозволяє компаніям зосередитися на своїй основній діяльності, перекладаючи технічні питання на зовнішні постачальники хмарних послуг.

Розвиток інформаційних систем також пов'язаний із застосуванням інтелектуальних технологій, таких як штучний інтелект та аналітика великих даних. Це відкриває нові можливості для аналізу та прогнозування, а також дозволяє бізнесу швидше реагувати на зміни в оточуючому середовищі.

Об'єктно-орієнтовані бази даних відіграють критичну роль у розвитку інформаційних систем, привносячи суттєві переваги та трансформуючи спосіб управління та обробки даних. Однією з головних особливостей ООБД є їхній об'єктно-орієнтований підхід до моделювання даних, що дозволяє ефективно вирішувати завдання розробки, управління та оптимізації інформаційних систем. У цьому контексті, розглядати роль ООБД в розвитку інформаційних систем — це визначити, як вони сприяють гнучкості, продуктивності, та забезпечують довгострокову стійкість у різноманітних областях застосування.

Однією з ключових характеристик ООБД є їх здатність представляти дані в об'єктно-орієнтованому стилі. Замість традиційних таблиць та стовпців, як у реляційних базах даних, ООБД використовують об'єкти, які мають властивості та методи. Це дозволяє моделювати реальні об'єкти та їх взаємодію, що значно полегшує відображення реальних бізнес-сценаріїв у структурі бази даних.

Підхід моделювання таких баз даних особливо корисний в розробці складних інформаційних систем, де потрібно моделювати різноманітні ентитети та їх взаємодію. Наприклад, у фінансових системах, де потрібно відслідковувати клієнтські рахунки, торгові операції та фінансові інструменти, ООБД надає можливість створювати зв'язки між цими об'єктами, відтворюючи реальні бізнес-взаємодії.

ООБД забезпечують гнучкий підхід до моделювання даних, що дозволяє легко вносити зміни в структуру бази даних. Це особливо корисно в умовах змінюючогося бізнес-середовища, де вимагається швидка адаптація інформаційних систем до

нових вимог. Така гнучкість забезпечується завдяки можливості додавати нові класи об'єктів, розширювати вже існуючі та легко взаємодіяти з ними. Це дає розробникам та аналітикам змогу швидко реагувати на зміни в бізнес-процесах, не вносячи значних змін в код програми або структуру бази даних.

АНАЛІЗ ОБ’ЄКТНО-ОРІЄНТОВАНИХ БАЗ ДАНИХ

Об'єктно-орієнтовані бази даних є типом систем управління базами даних, які зберігають дані в об'єктно-орієнтованій парадигмі програмування. Вони організовують і керують даними за допомогою об'єктів, які складаються з атрибутів (дані) та методів (функції). Об'єктно-орієнтовані бази даних дозволяють зберігати складні структури даних, включаючи успадкування, інкапсуляцію та поліморфізм. Вони надають засіб для прямого представлення об'єктів реального світу в базі даних, зберігаючи їх взаємозв'язки та поведінку. Цей підхід дозволяє ефективно та інтуїтивно обробляти дані, оскільки він тісно відповідає мовам програмування та моделям, які використовуються у розробці додатків. Ці БД підтримують принципи об'єктно-орієнтованого програмування, такі як абстракція даних, модульність та повторне використання коду. Вони пропонують переваги, такі як гнучкість, можливість розширення та здатність точно моделювати складні області. Також вони підходять для застосунків, які працюють зі складними структурами даних та вимагають ефективного представлення та обробки об'єктів.

Рис. 1.1. Модель даних ООБД.

1.1. Предметна область ООБД

Об'єктно-орієнтовані бази даних можуть вирішувати різноманітні завдання в інформаційній системі. Деякі з завдань, які вони можуть допомогти вирішити, включають:

- Ефективне зберігання та отримання складних структур даних: Об'єктно-орієнтовані бази даних добре підходять для керування складними структурами даних, такими як ієрархічні або мережеві дані. Вони надають ефективні механізми зберігання та отримання цих структур, що дозволяє легко навігувати та отримувати доступ до пов'язаних даних.
- Збереження моделей об'єктно-орієнтованого програмування: Використання об'єктно-орієнтованих баз даних дозволяє безпосередньо зберігати та отримувати об'єкти, як вони визначені в мові програмування. Це дозволяє зберігати парадигму об'єктно-орієнтованого програмування взаємодії між кодом додатку та базою даних.
- Підтримка успадкування та поліморфізму: Об'єктно-орієнтовані бази даних можуть керувати ієрархіями об'єктів та підтримувати концепції успадкування та поліморфізму. Це дозволяє моделювати складні зв'язки та поведінку між об'єктами, забезпечуючи більш гнучке та розширене представлення даних.

Перевагою об'єктно-орієнтованих баз даних над реляційними базами даних є їхня здатність зберігати та керувати складними структурами даних, зв'язками та об'єктами. Об'єктно-орієнтовані бази даних краще підходять для сценаріїв, що передбачають складні структури даних, збереження об'єктів,

динамічну розробку та специфічні для домену додатки. Вони чудово працюють з взаємопов'язаними об'єктами, забезпечують цілісність об'єктів, гнучкість для гнучкої розробки і добре підходять для таких областей, як наукові дослідження, просторовий аналіз даних, мультимедіа та складні симуляції.

ПРОБЛЕМИ ТРАДИЦІЙНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Традиційні інформаційні системи, хоч і є важливою складовою сучасного бізнесу, зіштовхуються із рядом суттєвих проблем, що обмежують їхню ефективність та адаптивність. Визначення цих проблем є кроком до удосконалення інформаційних технологій та забезпечення їхньої більшої відповідності потребам сучасного бізнес-середовища.

Однією з ключових проблем традиційних інформаційних систем є їхній ступінь складності та недостатня гнучкість у розподілених середовищах. Ці системи, розроблені для локальних операцій, стикаються з труднощами підтримки розподіленого обчислення та взаємодії в умовах глобального бізнесу. Ця проблема затримує їхню спроможність адаптуватися до сучасних вимог до глобальної співпраці та швидкості реакції на зміни.

Ще однією проблемою є обмежена масштабованість традиційних систем. Ці системи, які були розроблені для відносно обмеженої кількості користувачів та даних, часто неспроможні ефективно масштабуватися в умовах зростання обсягів інформації та користувачів. За збільшення навантаження система може демонструвати відчутні затримки та втрату продуктивності, що обмежує їхню спроможність вдовольняти потреби швидкозмінюючогося бізнес-середовища.

Також традиційні інформаційні системи стикаються із серйозними викликами в галузі безпеки та конфіденційності. Оскільки дані зосереджені та обробляються централізовано, це створює потенційні точки вразливості для несанкціонованого доступу. Забезпечення безпеки інформації та збереження конфіденційності стає

складною задачею, особливо в умовах зростання кількості хакерських атак та викликів щодо дотримання різноманітних регуляцій щодо захисту даних.

За зростання обсягів даних традиційні системи демонструють свою неефективність у роботі з великими обсягами інформації. Обробка та аналіз великих даних стає надто витратною, що обмежує їхню здатність використовувати цінні висновки, які може надати аналітика великих даних. Така неефективність стає ключовим обмеженням в умовах сучасного бізнесу, де швидкість прийняття рішень та аналіз великих обсягів даних має стратегічне значення.

Нарешті, традиційні інформаційні системи часто не відповідають вимогам гнучкості та адаптивності. Розробка нового функціоналу чи зміни існуючого може виявитися тривіальною задачею, що обмежує їхню здатність систем швидко реагувати на змінюючіся бізнес-вимоги. Умови великого конкурентного тиску та стрімкі зміни вимагають від інформаційних систем більшої гнучкості та швидкості адаптації, ніж можуть надати традиційні підходи.

СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ ТА РЕЛЯЦІЙНІ БАЗИ ДАНИХ

Сучасне інформаційне середовище переповнене об'ємами різноманітних даних, і реляційні бази даних (РБД) стали своєрідним стовпом для зберігання та управління цими даними. Проте, навіть з усією їхньою популярністю, РБД мають свої обмеження та проблеми, що можуть обмежити їхню ефективність та адаптивність в сучасному світі.

Можна виділити наступні проблеми використання реляційних баз даних:

- РБД оптимізовані для роботи з чітко визначеними, структурованими даними у вигляді таблиць. Однак, у світі великих даних, де домінує неструктурована інформація, така як тексти, графіка, аудіо і відео, РБД не завжди ефективні.

Наприклад, розпізнавання образів, аналіз текстів чи обробка великих потоків відеоданих вимагають більш гнучких інструментів. Наприклад, аналіз великих обсягів соціальних медіа-даних, де зазвичай присутні тексти, зображення та відео, може бути ускладненим в РБД.

- При збільшенні об'ємів даних чи потоку користувачів, РБД можуть стати складними в управлінні та масштабуванні. Для бізнесів, які розвиваються, це може призвести до затримок у відповідях на запити та загальної втрати продуктивності.
- Запити, які об'єднують дані з різних таблиць, часто вимагають значного часу для виконання. Складність оптимізації запитів може призводити до витрат часу та ресурсів, особливо в великих РБД.
- Розширення чи підтримка РБД може бути вкрай витратною задачею. Ліцензійні витрати, придбання обладнання та складність адміністрування можуть визначати значний бюджет на технічну інфраструктуру.
- В деяких випадках, де необхідно отримати миттєвий доступ до даних (наприклад, системи реального часу), затримки РБД можуть бути неприйнятними. Приклад: Фінансові установи, які вимагають миттєвого визначення фінансових показників, можуть виявити обмеженість РБД в забезпеченні швидкодії в реальному часі.
- Розширення або зміни в структурі РБД може виявитися трудомістким процесом, особливо коли система вже працює з великим об'ємом даних.
- Великі РБД можуть стикатися з труднощами забезпечення надійності та резервування, що може призвести до втрати даних або недоступності системи. Локальне виключення серверу може викликати тимасову втрату доступу до бази даних та втрату непродуктивного часу.
- У світі глобального бізнесу, де дані розміщуються в різних географічних регіонах, може виникати проблема з ефективністю обробки та звітування. Міжнародні компанії з представництвами в різних крахнах можуть стикатися із викликами синхронізації та обміну даними.

1.2. **Порівняння реляційних та не реляційних баз даних**

Об'єктно-орієнтовані бази даних за останні роки отримали певні досягнення, проте їх розвиток та поширення не настільки широкі, як у реляційних баз даних. Реляційні бази даних переважають на ринку баз даних завдяки своїй зрілості, встановленим стандартам та широкій підтримці та інструментарію.

З іншого боку, вони знайшли своє застосування в специфічних областях, де їх конкретні переваги є цінними. Вони продовжують розвиватись та вдосконалюватись завдяки дослідженням та спеціалізованим виробникам. Проте їх використання часто обмежується певними галузями або індустріями, які потребують складних структур даних, постійності об'єктів або спеціалізованих можливостей запитування.

Загалом, хоча об'єктно-орієнтовані бази даних мають свої переваги та продовжують розвиватись, вони не досягли такого рівня загального використання, як у реляційних баз даних. Вибір між ними залежить від конкретних вимог, складності структур даних та балансу між об'єктно-орієнтованими принципами та перевагами зрілої екосистеми реляційних баз даних.

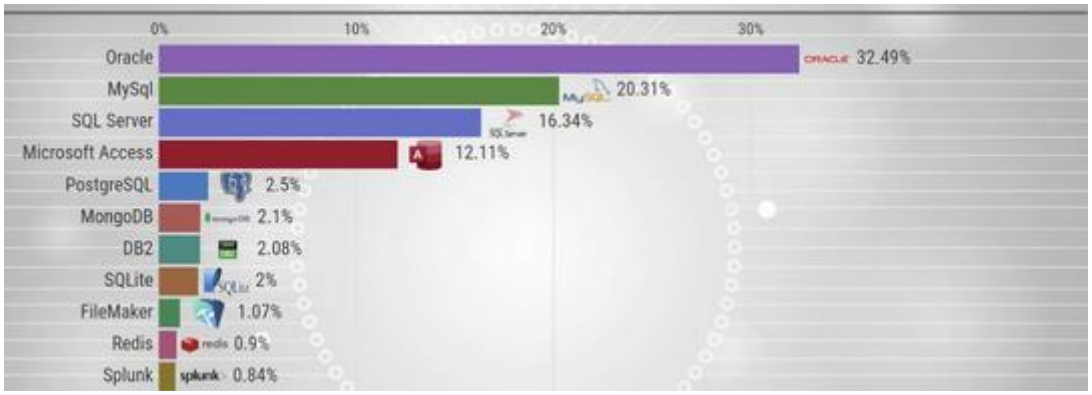


Рис.1.2. Найпопулярніші БД.

Додавання даних до реляційної бази даних зазвичай передбачає вставку рядків у попередньо визначені таблиці, що вимагає дотримання строгих схем та обмежень цілісності даних. Процес включає кілька етапів, таких як перевірка даних, генерація унікальних ідентифікаторів та забезпечення реляційної цілісності. Це може уповільнити процес вставки даних, особливо при роботі зі складними залежностями та великими наборами даних.

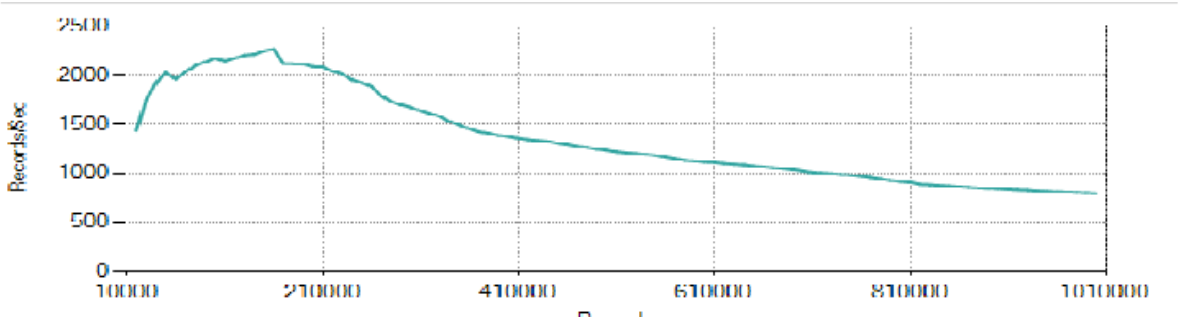


Рис. 1.3. Швидкість операції insert в *SQL Server*.

З іншого боку, додавання даних до об'єктно-орієнтованої бази даних передбачає створення або інстанціювання об'єктів та збереження їх безпосередньо у базі даних. Об'єктно-орієнтовані бази даних призначені для роботи зі складними структурами даних та залежностями, що дозволяє більш гнучке та ефективне вставлення даних. Процес часто відбувається швидше порівняно з реляційними базами даних, оскільки не потрібно виконувати відображення об'єктів на таблиці або забезпечувати реляційні обмеження.

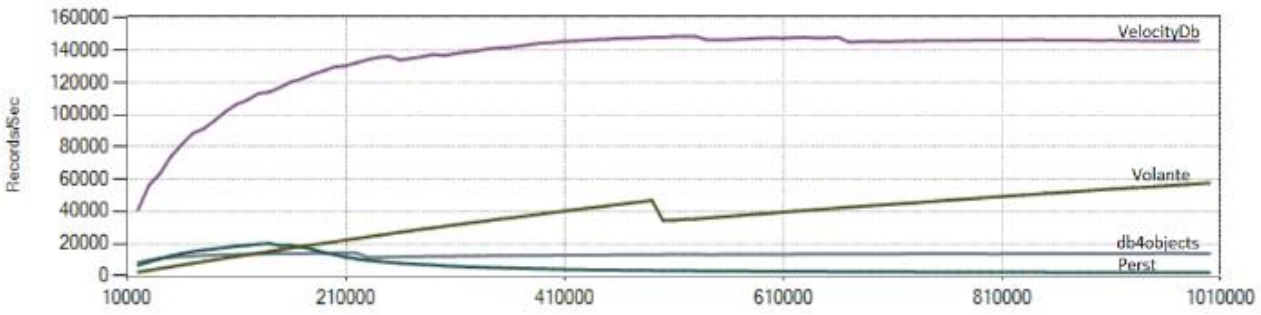


Рис. 1.4. Швидкість операції insert в ООДБ.

Таблиця 1.1.

Критерій	Реляційні БД	Об'єктно-орієнтовані БД	NoSQL БД
Основний принцип	Таблиці з рядками і стовпцями	Об'єкти з методами та атрибутами	Гнучкі схеми даних (ключ-значення, документи, графи)

Схема даних	Статична	Динамічна	Динамічна
Запити	SQL	Об'єктно-орієнтовані мови запитів	Різні, залежно від типу (напр. MongoDB використовує BSON)
Транзакції	Підтримка ACID	Обмежена підтримка ACID	Залежить від системи (часто обмежена або відсутня)
Масштабування	Вертикальне	Горизонтальне	Горизонтальне
Оптимізовано для	Структурованих запитів і звітів	Складних об'єктів і відносин	Великих обсягів неструктурованих даних
Приклади використання	Бухгалтерський облік, управління ресурсами	Системи, що вимагають комплексного представлення даних (напр., CAD)	Великі веб-додатки, Big Data

Підсумовуючи, швидкість додавання даних до об'єктно-орієнтованої бази даних, як правило, є вищою, ніж у реляційної бази даних, завдяки безпосередньому збереженню об'єктів та відсутності складних процесів відображення та обмеження. Однак фактична продуктивність може варіюватися залежно від факторів, таких як дизайн бази даних, стратегії індексації, апаратні ресурси та конкретні операції, що виконуються під час процесу вставки даних.

Також варто зазначити, що розробнику, який буде викорнистовувати базу даних в своїй роботі, буде набагато простіше робити записи, а також витягувати дані з бази. Це обумовлено тим, що йому не доведеться створювати додаткові класи, в які він буде записувати дані з бази, а потім перестворювати об'єкт потрібного йому класу. Це значно пришвидшить його роботу.

Нажаль як і будь-яка інша модель, ООДБ має свої недоліки, а саме:

- Відсутність універсальних стандартів: У порівнянні з реляційними базами даних, об'єктно-орієнтовані бази даних не мають універсальних стандартів, що були б широко прийняті, наприклад, *SQL*.
- Обмежена підтримка складних запитів: Об'єктно-орієнтовані бази даних можуть мати обмежену підтримку для складних запитів, які включають багато залежностей та складних операцій з'єднання.
- Проблеми сумісності: Інтеграція об'єктно-орієнтованих баз даних з існуючими системами або додатками, розробленими для реляційних баз даних, може бути складною, оскільки це може потребувати додаткових зусиль для забезпечення взаємодії між двома моделями даних.
- Дублювання даних та додаткові витрати на зберігання: Об'єктно-орієнтовані бази даних можуть мати більший обсяг зберігання, порівняно з реляційними базами даних.

АРХІТЕКТУРА ОБ’ЄКТНО-ОРІЄНТОВАНИХ БАЗ ДАНИХ

Архітектура об'єктно-орієнтованих баз даних (ООБД) розроблена з урахуванням принципів об'єктно-орієнтованого програмування, що дозволяє ефективно представляти складні дані та взаємозв'язки в базі даних. Основними елементами цієї архітектури є об'єкти, класи, успадкування, поліморфізм та інкапсуляція.

У ООБД, основною одиницею зберігання є об'єкт, який включає в себе як дані, так і методи (функції) для роботи з цими даними. Кожен об'єкт є екземпляром певного класу, який визначає структуру (атрибути) і поведінку (методи) своїх об'єктів. Класи можуть бути організовані в ієрархії, дозволяючи використовувати спадкування.

Упадкування дозволяє класам наслідувати атрибути та методи від їх

батьківських класів. Це сприяє повторному використанню коду та зменшує редундантність. У ООБД, спадкування дозволяє легко модифікувати та розширювати структури даних.

Поліморфізм у ООБД дозволяє об'єктам різних класів оброблятися за допомогою одного і того ж інтерфейсу. Це означає, що методи, визначені в одному класі, можуть бути використані об'єктами іншого класу без зміни коду.

Ікапсуляція є ключовою концепцією ООБД, яка дозволяє обмежити доступ до внутрішніх даних об'єкта. Дані об'єкта захищені від зовнішнього доступу, а всі взаємодії з об'єктом відбуваються через його методи.

ООБД підтримують складні запити, які використовують методи об'єктів для маніпуляцій з даними. Транзакції в ООБД забезпечують надійність та цілісність даних, дозволяючи виконувати декілька операцій як єдину логічну одиницю роботи. Вони також включають механізми індексації для підвищення швидкості доступу до даних. Системи оптимізації запитів автоматично вибирають найефективніший спосіб виконання запитів.

Ця архітектура надає потужний механізм для розробки високофункціональних, гнучких та надійних інформаційних систем, особливо в умовах, де потрібно представити складні взаємозв'язки та ієрархії даних.

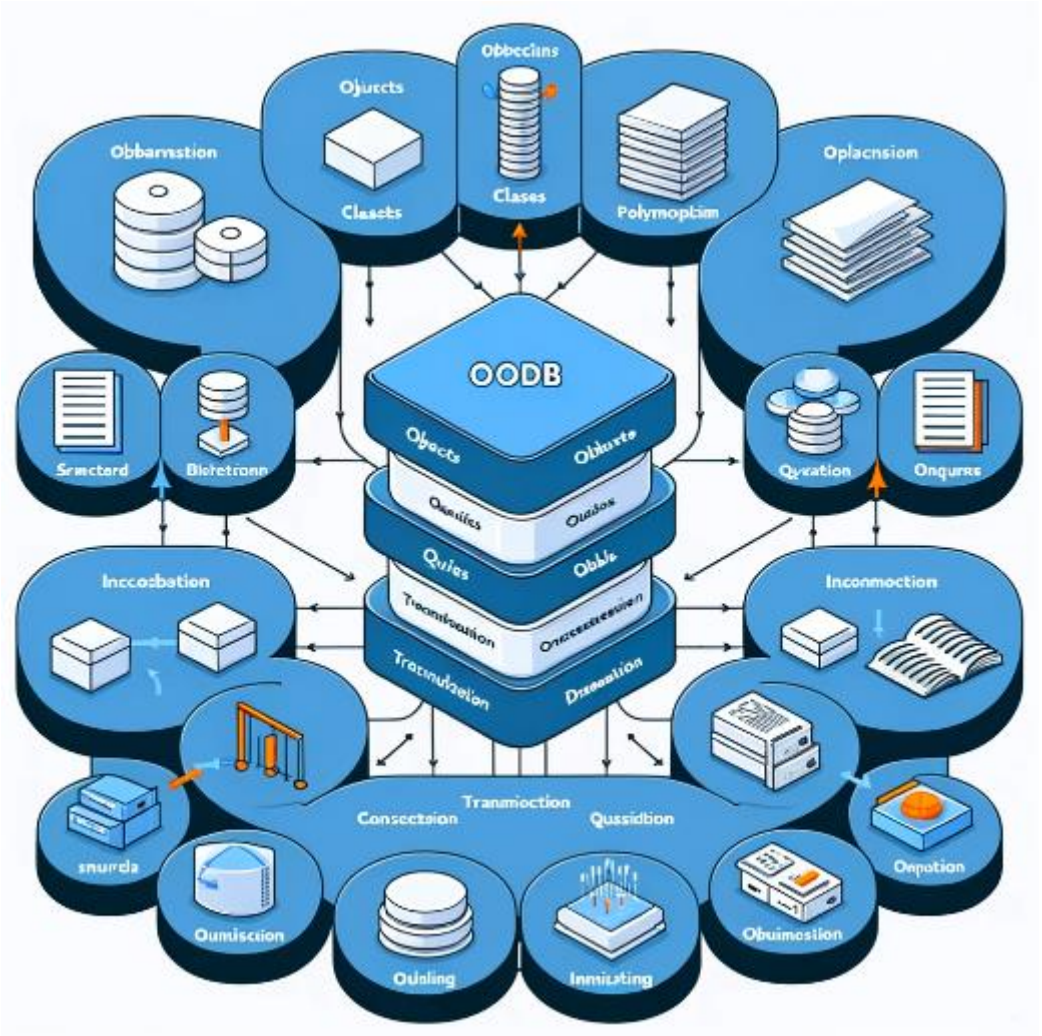


Рис 228. Архітектура ООБД

Переходячи до більш технічних аспектів архітектури об'єктно-орієнтованих баз даних, важливо розглянути такі елементи як обробка запитів, управління транзакціями та забезпечення цілісності даних.

- Обробка запитів: в ООБД, запити часто формулюються з використанням об'єктно-орієнтованих мов запитів, які надають можливість використання об'єктів, їхніх властивостей та методів безпосередньо в запиті. Це дозволяє виконувати складні запити, які включають навігацію по об'єктних відносинах та використання поліморфізму. Механізми оптимізації запитів в ООБД аналізують запит для визначення найбільш ефективного шляху доступу до даних, враховуючи індекси та структуру даних.

- Управління транзакціями: транзакції в ООБД керують послідовностями

операцій з даними для забезпечення цілісності бази даних. Вони підтримують властивості ACID (атомарність, послідовність, ізоляція, довговічність), що є критично важливими для забезпечення надійності та стабільності даних. Атомарність гарантує, що всі операції в транзакції виконуються повністю або не виконуються взагалі. Послідовність забезпечує, що база даних перебуває в послідовному стані перед та після транзакції. Ізоляція запобігає взаємному впливу паралельних транзакцій. Довговічність гарантує, що результати транзакції збережуться, навіть у разі системних збоїв.

- Цілісність даних: ООБД забезпечують механізми для підтримки цілісності даних, включаючи обмеження цілісності, тригери та процедури. Обмеження цілісності визначають правила, яким повинні відповідати дані (наприклад, унікальність ключів, обмеження на значення атрибутів). Тригери - це спеціальні процедури, які автоматично викликаються під час виконання певних операцій з даними (наприклад, при додаванні або оновленні записів). Це дозволяє автоматично підтримувати цілісність та виконувати додаткову обробку.

- Розподілені ООБД**:** Розподілені ООБД є важливим аспектом сучасних інформаційних систем, дозволяючи розподіляти дані між кількома вузлами. Це підвищує доступність, масштабованість та надійність системи. Розподілені ООБД використовують спеціалізовані механізми синхронізації та узгодження даних для забезпечення консистентності між різними вузлами.

Загалом, архітектура об'єктно-орієнтованих баз даних надає потужний інструментарій для розробки комплексних, високоефективних інформаційних систем, особливо в умовах, що вимагають гнучкого представлення та обробки складних структур даних.

Висновки до розділу

У висновку до першого розділу, можна підкреслити кілька ключових аспектів. Архітектура об'єктно-орієнтованих баз даних представляє собою потужний інструмент, що забезпечує гнучкість, масштабованість та високий рівень абстракції для розробки складних інформаційних систем. Характеристики, такі як спадкування, інкапсуляція та поліморфізм, є основою для створення високоадаптивних та легко розширюваних систем, які можуть ефективно використовувати об'єктно-орієнтовану парадигму.

Важливою перевагою об'єктно-орієнтованих баз даних є їх здатність моделювати складні структури даних, що дуже важливо для доменів, де дані мають високий ступінь взаємозв'язків та ієрархій. Це забезпечує ефективне відображення реального світу у вигляді даних, що є важливим для точного та глибокого аналізу. Проте, слід враховувати і недоліки, такі як відносно складніший запит до бази даних та можливі виклики з продуктивністю у порівнянні з традиційними реляційними базами даних. Таким чином, вибір між об'єктно-орієнтованими та реляційними базами даних повинен базуватися на конкретних вимогах проекту та характеристиках даних.

Загалом, архітектура об'єктно-орієнтованих баз даних є важливою для розвитку інформаційних систем, зокрема у сферах, де необхідне глибоке моделювання даних та висока адаптивність системи до змінних вимог користувачів та бізнес-процесів. Вона пропонує унікальний підхід до управління даними, який може бути особливо цінним у складних та динамічних доменах.

РОЗДІЛ 2

МЕТОДИ РЕАЛІЗАЦІЙ ІНФОРМАЦІЙНОЇ СИСТЕМИ

АНАЛІЗ ВИМОГ ДО CRM СИСТЕМ

Аналіз вимог до CRM-системи є важливим кроком у процесі розробки, оскільки він визначає функціональні та нефункціональні вимоги, які повинна задовольняти система. Цей аналіз забезпечує основу для проектування та реалізації системи, зокрема при використанні мови програмування Java.

Функціональні вимоги визначають конкретні завдання та функції, які повинна виконувати CRM-система. До них можуть відноситися:

- Управління клієнтською базою - збір та зберігання інформації про клієнтів, їхні контакти, історію взаємодій.

- Ведення взаємодій з клієнтами - фіксація всіх контактів з клієнтами, включаючи дзвінки, електронні листи, зустрічі.

- Управління продажами - відстеження продажів, ведення історії замовлень, управління продажними воронками.

- Маркетингові кампанії - планування та виконання маркетингових кампаній, аналіз їхньої ефективності.

- Звітність та аналітика - генерація звітів за різними параметрами, аналіз даних для підтримки прийняття рішень.

Нефункціональні вимоги визначають загальні характеристики та стандарти системи, які включають:

Продуктивність - система має ефективно обробляти запити, забезпечуючи швидку відповідь.

Масштабованість - здатність системи адаптуватися до зростаючого числа користувачів та обсягу даних.

Безпека - захист даних клієнтів та забезпечення конфіденційності інформації.

Надійність - забезпечення стабільної роботи системи, мінімізація збоїв та помилок.

Інтеграційні можливості - здатність інтегруватися з іншими системами та програмними рішеннями.

Для кращого розуміння вимог, корисно створити діаграму вимог, яка візуально представляє основні компоненти системи та їх взаємодії.

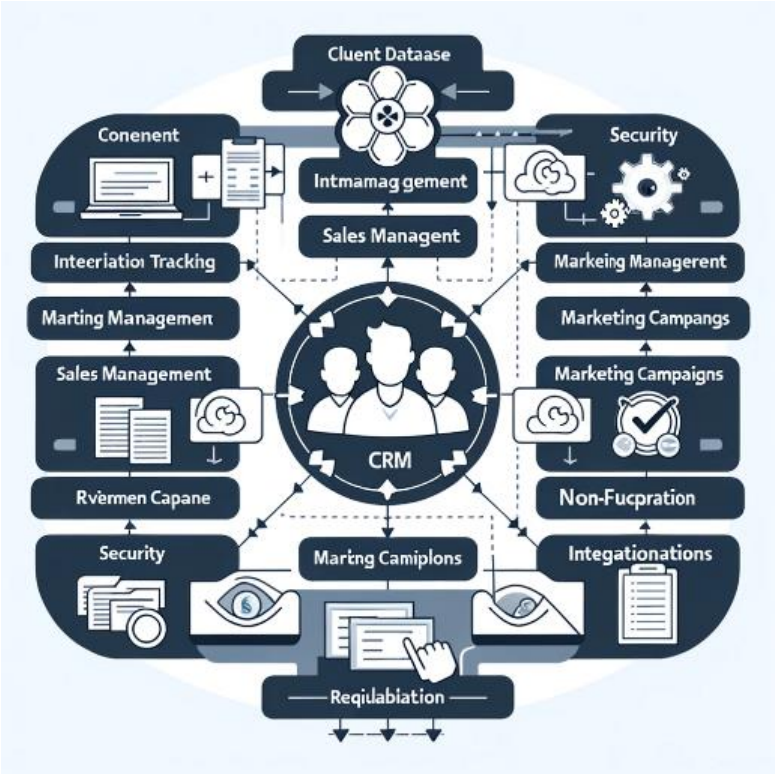


рис. 322 «Діаграма вимог CRM системи»

Вище представлена діаграма, яка візуалізує вимоги до CRM-системи, розробленої на Java. На діаграмі зображено ключові компоненти системи, включаючи управління клієнтською базою, ведення взаємодій з клієнтами, управління продажами, маркетингові кампанії, звітність та аналітику. Також враховано нефункціональні вимоги, як-от продуктивність, масштабованість, безпека, надійність та інтеграційні можливості. Ця діаграма може служити в якості наочного посібника для розуміння структури та ключових властивостей вашої CRM-системи.

Продовжуючи аналіз вимог до CRM-системи, важливо звернути увагу на деталізацію вимог і підготовку до їх реалізації. Це включає визначення конкретних сценаріїв використання, визначення інтерфейсів користувача, а також планування архітектури системи.

Для кожної функціональної вимоги, такої як управління клієнтською базою чи ведення продажів, потрібно розробити детальні сценарії використання. Це допоможе точно визначити, як користувачі будуть взаємодіяти з системою. Наприклад, для функції управління клієнтською базою, сценарій може включати створення нового клієнтського профілю, оновлення існуючих даних, пошук інформації про клієнтів тощо.

Дизайн інтерфейсу користувача відіграє ключову роль в ефективності CRM-системи. Інтерфейс має бути інтуїтивно зрозумілим, забезпечувати легкий доступ до основних функцій та даних. Важливо також врахувати мобільну адаптивність, щоб користувачі могли ефективно використовувати систему з різних пристроїв.

Архітектура CRM-системи має відображати вимоги до продуктивності, масштабованості та безпеки. Використовуючи Java, можна застосувати багаторівневу архітектуру, що розділяє логіку користувальницького інтерфейсу, бізнес-логіку та доступ до даних. Це не тільки спрощує розробку та тестування, але й забезпечує більшу гнучкість та масштабованість. Інтеграція з об'єктно-орієнтованою базою даних може бути реалізована через JPA, що спростить роботу з даними та підвищить ефективність взаємодії з базою даних.

Необхідно також планувати процеси забезпечення якості, включаючи тестування системи. Використання автоматизованих тестів, які охоплюють різні рівні системи (від модульних тестів до інтеграційного та системного тестування), допоможе забезпечити стабільність та надійність CRM-системи. Це також дасть змогу своєчасно виявляти та усувати помилки, що є критично важливим для бізнес-критичних систем, таких як CRM.

Подібний всебічний підхід до аналізу вимог не тільки забезпечує підготовку ґрунту для ефективного проектування та реалізації CRM-системи, але й сприяє розумінню всіх аспектів її функціонування, від взаємодії з кінцевим користувачем

до бекенд-логіки та інтеграції з іншими системами.

Аналіз цільової аудиторії та бізнес-процесів є ключовим етапом у проектуванні CRM-системи, оскільки він допомагає визначити, хто буде використовувати систему та яким чином вона буде інтегрована в існуючі робочі процеси.

Для ефективного проектування CRM-системи необхідно детально зрозуміти, хто є кінцевими користувачами. Це можуть бути менеджери з продажу, маркетологи, служба підтримки клієнтів, топ-менеджмент та інші співробітники, які взаємодіють з клієнтами або використовують дані про клієнтів для прийняття рішень. Важливо врахувати їхні потреби, рівень технічної компетенції, а також особливості їхньої роботи. Наприклад, менеджери з продажу потребують швидкого доступу до інформації про клієнтів і замовлення, тоді як топ-менеджмент може бути зацікавлений у агрегованих звітах та аналітиці.

Бізнес-процеси, які будуть інтегровані з CRM-системою, мають бути ретельно проаналізовані. Це означає вивчення поточних робочих процедур, потоків даних та взаємодій між різними відділами. Наприклад, як інформація про клієнтів збирається, обробляється та використовується у відділі продажів, маркетингу чи службі підтримки? Які існуючі системи вже використовуються, і як CRM-система може бути інтегрована з ними для оптимізації процесів?

Цей аналіз допоможе зрозуміти, як CRM-система може підвищити ефективність бізнес-процесів, зменшити час на обробку даних і покращити взаємодію з клієнтами. Він також виявить потенційні виклики та обмеження, з якими може зіткнутися система, та допоможе в плануванні необхідних налаштувань та інтеграцій.

Результати цих аналізів відіграють вирішальну роль у процесі проектування CRM-системи, забезпечуючи, що вона буде відповідати реальним потребам користувачів та бізнесу, та буде гнучкою для адаптації до змін у бізнес-процесах. Окрім цього, вони допомагають уникнути непотрібних помилок та витрат, пов'язаних з розробкою недостатньо адаптованої системи.

Необхідно також зосередитися на виявленні конкретних вимог та очікувань

кінцевих користувачів. Це передбачає збір зворотного зв'язку від потенційних користувачів через інтерв'ю, опитування, фокус-групи або аналіз вже існуючих звітів та відгуків. Зібрана інформація допоможе визначити, які функції є критично важливими, а які можуть бути вторинними або не потрібними.

Різні відділи в організації можуть мати унікальні потреби та вимоги до CRM-системи. Наприклад, відділ продажів може шукати інструменти для керування взаємодіями з клієнтами та ведення записів про продажі, тоді як маркетинговий відділ може потребувати більш складних функцій для аналізу даних клієнтів та управління кампаніями. Важливо ідентифікувати ці відмінності та забезпечити, щоб система могла бути налаштована або адаптована для задоволення різноманітних потреб.

CRM-система рідко існує в ізоляції. Вона часто має інтегруватися з іншими бізнес-системами, такими як ERP (Enterprise Resource Planning), системи електронної комерції, облікові системи, і т.д. Аналіз цих інтеграційних точок є важливим для забезпечення плавності бізнес-процесів. Це включає визначення даних, які потрібно передавати між системами, та розробку інтерфейсів для їх обміну.

Під час аналізу бізнес-процесів також важливо врахувати потенційні майбутні зміни в організації. Це може включати розширення компанії, зміни в ринкових умовах, нові продукти або послуги, що можуть вплинути на взаємодію з клієнтами та управління даними. Система повинна бути достатньо гнучкою, щоб адаптуватися до цих змін без необхідності значних переробок або додаткових витрат.

Після завершення аналізу цільової аудиторії та бізнес-процесів створюється документація, яка викладає зібрані вимоги та специфікації. Ця документація стає основою для наступних етапів розробки, включаючи проектування системи, вибір технологій, розробку та тестування. Важливо, щоб ця документація була чіткою, детальною та легко зрозумілою для всіх зацікавлених сторін, включаючи розробників, менеджерів проектів та кінцевих користувачів.

МОДЕЛЬ ДАНИХ ДЛЯ ООБД

Модель даних для об'єктно-орієнтованої бази даних в CRM-системі є фундаментальною складовою, яка визначає, як дані будуть зберігатися, організовані та взаємодіяти між собою. Цей етап вимагає глибокого розуміння бізнес-потреб та бізнес-процесів, а також технічних можливостей об'єктно-орієнтованих баз даних.

Початок розробки моделі даних полягає у визначенні ключових об'єктів, які будуть представлені в базі даних. В контексті CRM, такими об'єктами можуть бути клієнти, контакти, продажі, маркетингові кампанії, звіти тощо. Для кожного об'єкта створюється клас, який визначає його атрибути (наприклад, ім'я, адреса, історія взаємодій для клієнта) та методи, що дозволяють виконувати операції з цими даними.

Важливою частиною моделі даних є визначення відносин між різними об'єктами. Наприклад, відносини можуть включати зв'язки між клієнтами та їхніми замовленнями, між контактами та маркетинговими кампаніями. В об'єктно-орієнтованих базах даних ці відносини часто представлені через використання посилань або вбудованих об'єктів, що дозволяє ефективно моделювати складні взаємозв'язки.

Об'єктно-орієнтовані бази даних дозволяють використовувати наслідування та поліморфізм при проектуванні моделей даних. Це означає, що можна створювати загальні класи (наприклад, клас "Контакт") з базовими атрибутами та методами, які потім можуть бути уточнені в специфічних підкласах (наприклад, "Клієнт" або "Постачальник"). Це сприяє більшій гнучкості та повторному використанню коду.

Під час проектування моделі даних потрібно також враховувати потреби в гнучкості та масштабованості. Модель повинна бути спроектована таким чином, щоб вона могла легко адаптуватися до змін у бізнес-вимогах, а також підтримувати зростання кількості даних та користувачів.

На завершення, необхідно врахувати можливість інтеграції CRM-системи з іншими існуючими системами. Модель даних повинна бути сумісною з даними, які можуть бути імпортовані або експортовані з інших систем, що забезпечує гладке функціонування перехресних бізнес-процесів.

Завершуючи цей розділ, ми маємо чітку та гнучку модель даних, яка є

основою для розробки ефективної та адаптивної CRM-системи, здатної задовольняти поточні та майбутні бізнес-вимоги. Вона повинна бути детально задокументована для забезпечення ясності та послідовності під час наступних етапів розробки.

Оптимізація моделі даних включає переконання в тому, що структура даних ефективно підтримує запити, які будуть на неї виконуватися. Це включає:

- Індексація: визначення ключових атрибутів для індексації, щоб покращити швидкість виконання запитів.

- Нормалізація та Денормалізація: збалансування між нормалізацією даних для уникнення редундантності та денормалізацією для підвищення продуктивності запитів.

- Управління Великими Об'ємами Даних: стратегії для ефективного зберігання та обробки великих об'ємів даних, таких як партіціювання даних.

Інтеграція даних з іншими системами є критично важливою для забезпечення єдності інформаційного простору в організації. Це може включати інтеграцію з системами електронної комерції, бухгалтерськими системами, соціальними мережами тощо. Необхідно визначити формати даних та протоколи для обміну даними, а також механізми перетворення та валідації даних.

Безпека даних у CRM-системі є надзвичайно важливою, особливо з огляду на конфіденційність інформації про клієнтів. Це включає розробку політик доступу до даних, шифрування конфіденційної інформації та впровадження механізмів аудиту для відстеження доступу та змін у даних.

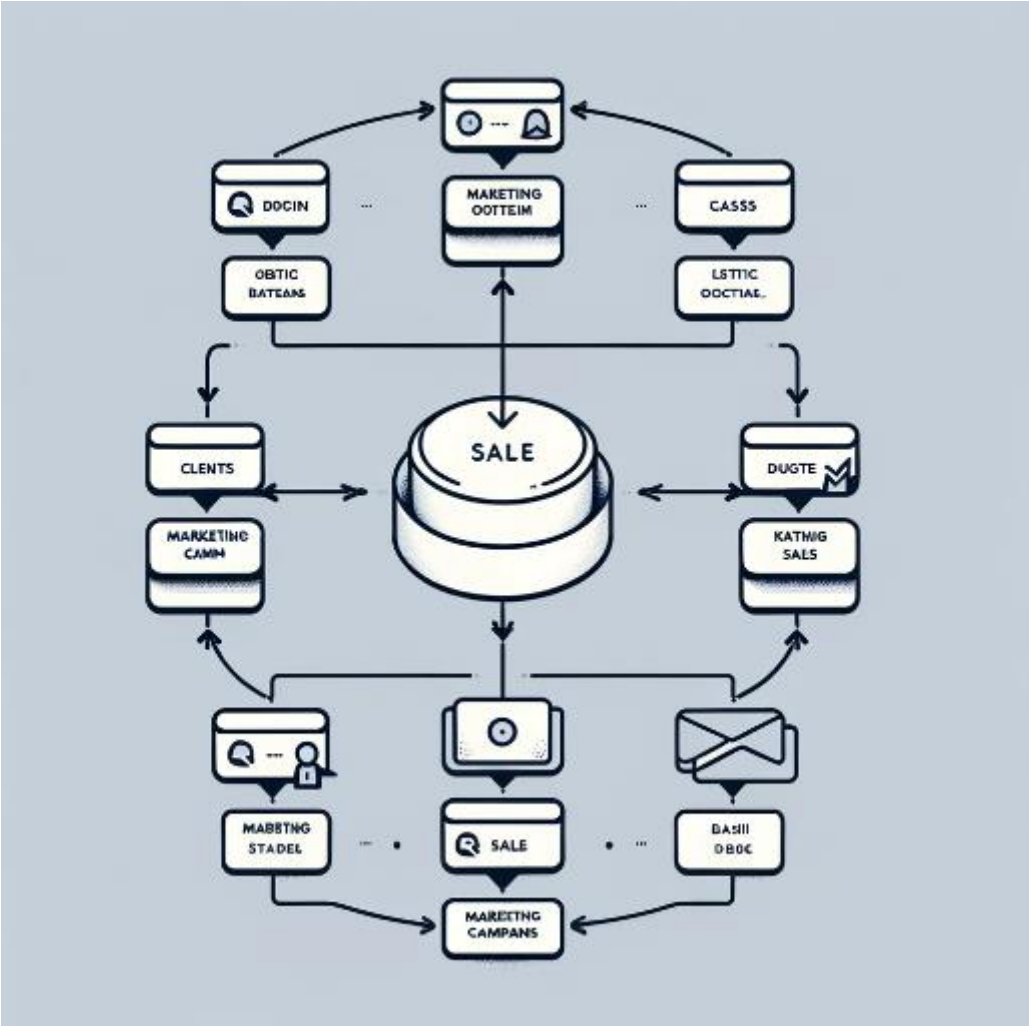


рис 1488 «Модель даних для CRM на основі ООБД»

МЕТОДИ РЕАЛІЗАЦІЇ

При виборі методів реалізації для CRM-системи, основну увагу слід зосередити на виборі підходів програмування, архітектурних шаблонів, а також виборі відповідного середовища розробки. Ось декілька ключових аспектів, які потрібно врахувати:

Об'єктно-Орієнтоване Програмування (ООП)

Оскільки CRM-система буде розроблятися на Java, використання принципів ООП є ідеальним. ООП дозволяє створювати модульні, масштабовані та легко підтримувані системи. Ключові аспекти ООП включають інкапсуляцію, наслідування та поліморфізм, які допомагають в структуруванні коду та

забезпечують ефективне управління даними.

Архітектурні шаблони в програмуванні визначають загальну структуру програмного забезпечення, спрощуючи розробку, підтримку та масштабування систем. Для CRM-системи, особливо розробленої на Java, вибір архітектурного шаблону має вирішальне значення. Ось детальний огляд ключових архітектурних шаблонів:

Модель-Вид-Контролер (MVC)

- Модель - ця частина відповідає за дані та бізнес-логіку. У контексті CRM, модель зберігатиме дані про клієнтів, взаємодії, історію продажів та інші бізнес-процеси.

- Вид - визначає, як дані представляються користувачам. В CRM це можуть бути різні користувацькі інтерфейси для введення та відображення інформації про клієнтів, звітів, аналітики тощо.

- Контролер - взаємодіє з моделлю та видом, обробляючи вхідні дані від користувача, запитуючи дані від моделі та вибираючи відповідний вид для відображення.MVC допомагає у розмежуванні логіки, представлення та бізнес-процесів, що полегшує тестування, підтримку та масштабування додатку.

Сервісно-Орієнтована Архітектура (SOA)

- Сервіси - SOA базується на використанні незалежних сервісів, кожен з яких виконує конкретну бізнес-функцію. Наприклад, у CRM системі може бути сервіс для управління клієнтами, інший для обробки замовлень і так далі.

- Зв'язок між сервісами - зазвичай використовуються веб-сервіси або API для зв'язку між сервісами, що дозволяє легко інтегрувати різні частини системи та забезпечує велику гнучкість.SOA є ідеальним для складних систем, де різні компоненти можуть бути розроблені, розгорнуті та оновлені незалежно один від одного.

Мікросервісна Архітектура

- Мікросервіси: - цей шаблон розбиває додаток на менші, незалежні сервіси, кожен з яких виконує специфічну функцію. Наприклад, в CRM системі може бути

мікросервіс для відслідковування взаємодій з клієнтами, інший для аналітики продажів тощо.

- Незалежність та гнучкість - кожен мікросервіс може бути розроблений, тестований, розгорнутий та масштабований незалежно. Це дозволяє впроваджувати нові функції та оновлення швидко і з мінімальним впливом на інші частини системи.

Використання MVC архітектури у поєднанні з клієнт-сервер моделлю для розробки CRM-системи є стратегічним вибором, який забезпечує чітке розмежування логіки, інтерфейсу користувача та управління даними, а також ефективну взаємодію між різними частинами системи.

Модель клієнт-сервер:

- У моделі клієнт-сервер, клієнт є переднім кінцем CRM-системи, зазвичай реалізованим через веб-інтерфейс або додаток. Клієнтська частина забезпечує користувацький інтерфейс та взаємодію з користувачем, відправляючи запити до сервера та отримуючи відповіді.
- Сервер є бекендом системи, де розміщується бізнес-логіка (часто відповідає компоненту 'Модель' у MVC). Сервер обробляє запити від клієнта, виконує операції з даними, взаємодіє з базою даних та надсилає необхідну інформацію назад клієнту.

Поєднання MVC та клієнт-сервер моделей дозволяє створювати гнучкі, масштабовані та легко підтримувані системи. У CRM-системі це забезпечує чітке відокремлення між логікою обробки даних (сервер) та інтерфейсом користувача (клієнт), що спрощує розробку та тестування різних частин системи. Такий підхід також сприяє кращій безпеці, оскільки бізнес-логіка та дані знаходяться на сервері, що зменшує ризики, пов'язані з безпосереднім доступом до них з клієнтської сторони.

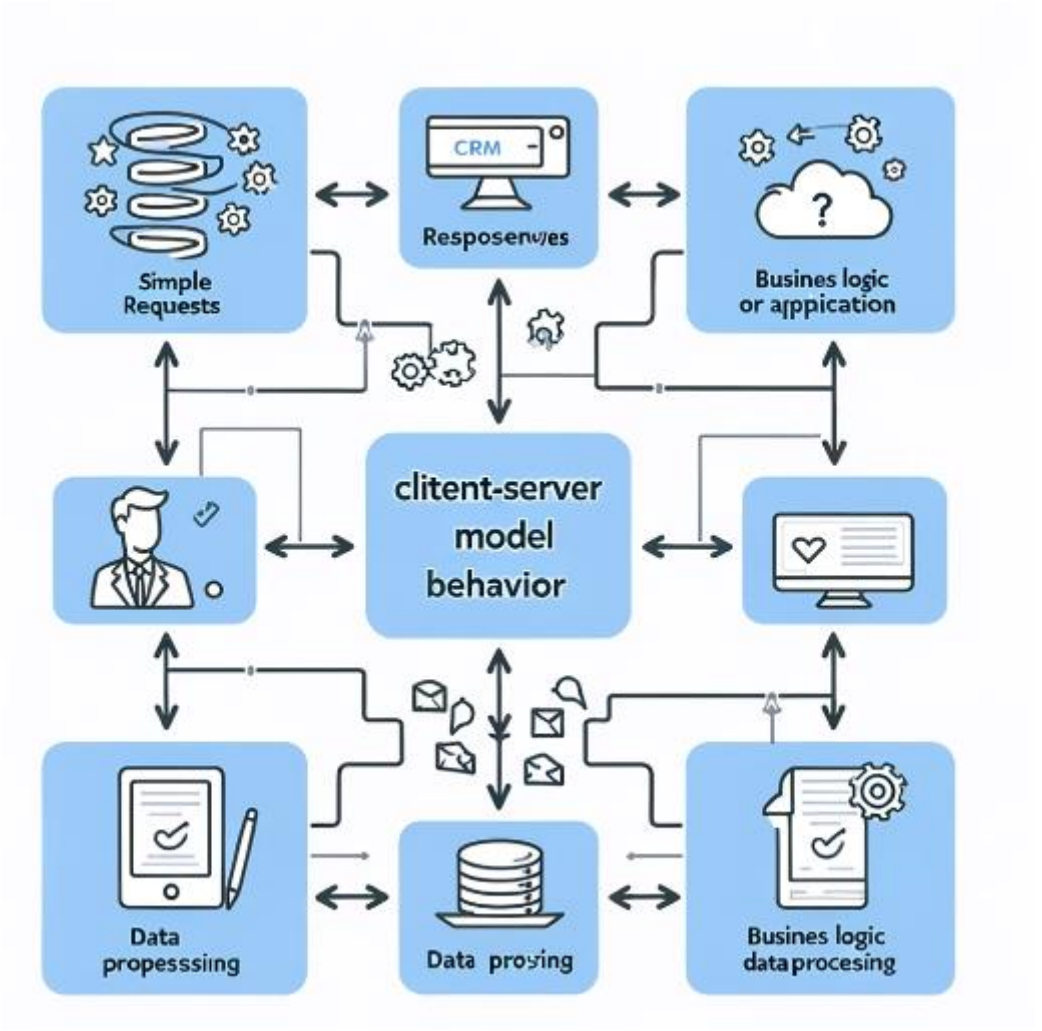


Рис 1337 «Діаграма поведінки моделі клієнт-сервер»

Діаграма, яку ви бачите, демонструє базову поведінку моделі клієнт-сервер в системі CRM. Ця модель складається з двох ключових компонентів: клієнта та сервера, кожен з яких відіграє важливу роль у взаємодії системи.

Клієнтська частина є інтерфейсом, через який користувачі взаємодіють з системою. Це може бути веб-інтерфейс або додаток. Клієнт відправляє запити до сервера, наприклад, запит на отримання даних про клієнтів або створення нового замовлення. Ці запити включають у себе всю необхідну інформацію, яка потрібна серверу для обробки.

Сервер обробляє вхідні запити від клієнта. Він відповідає за виконання бізнес-логіки, обробку даних, взаємодію з базою даних, та інші операції, необхідні для обслуговування запитів клієнта. Після обробки запиту, сервер генерує відповідь, яка може містити запитану інформацію, статус виконання операції, або повідомлення

про помилку.

Процес взаємодії між клієнтом та сервером включає кілька кроків:

- Надсилання запиту - клієнт надсилає запит на сервер. Цей запит може бути пов'язаний з отриманням, оновленням, видаленням або створенням даних.
- Обробка запиту сервером - сервер приймає запит, обробляє його з використанням бізнес-логіки та доступу до даних.
- Надсилання відповіді - сервер надсилає відповідь назад клієнту. Ця відповідь може включати результати обробки або підтвердження виконання операцій.
- Обробка відповіді клієнтом - клієнт отримує відповідь від сервера та відображає відповідну інформацію або статус для користувача.

Ця проста діаграма та пояснення демонструють основні принципи роботи моделі клієнт-сервер в CRM-системі, підкреслюючи роль кожного компонента у забезпеченні ефективної та надійної взаємодії.

Інтерфейс користувача в CRM-системі відіграє ключову роль, забезпечуючи ефективну взаємодію між користувачем та системою. Дизайн інтерфейсу повинен бути інтуїтивно зрозумілим та зручним, щоб користувачі могли легко навігувати та використовувати різноманітні функції системи. Важливим аспектом є створення чистого, не перевантаженого дизайну, який зосереджує увагу на найважливіших елементах інтерфейсу та функціональностях.

Функціональні можливості CRM-системи повинні включати управління інформацією про клієнтів, ведення обліку взаємодій, управління продажами та маркетинговими кампаніями, а також аналітику та звітність. Це дозволяє компаніям краще розуміти своїх клієнтів, підвищувати ефективність комунікацій, оптимізувати процеси продажу та маркетингу, а також приймати дані-орієнтовані бізнес-рішення.

Для інтерфейсу користувача важливою є також адаптивність, особливо у випадку мобільних пристроїв та різних розмірів екранів. Це забезпечує комфортне використання системи в будь-яких умовах, що є важливим для сучасних бізнес-процесів, які часто вимагають гнучкості та мобільності.

Інтерактивні елементи, такі як форми для введення даних, графіки,

інтерактивні звіти, повинні бути реалізовані таким чином, щоб користувачам було легко з ними працювати. Це включає забезпечення зрозумілих інструкцій, повідомлень про помилки, допоміжних підказок та забезпечення загальної зручності використання системи.

Нарешті, важливим аспектом є забезпечення безпеки в інтерфейсі користувача. Це означає, що доступ до різних функцій та даних повинен бути регульований залежно від ролі користувача в системі, а також важливо забезпечити захист даних при їх передачі та зберіганні.

Таким чином, розробка інтерфейсу користувача та функціональностей у CRM-системі вимагає ретельного планування та зосередження уваги на зручності, ефективності взаємодії, гнучкості та безпеці, щоб забезпечити високу продуктивність роботи користувачів і досягнення бізнес-цілей.

РЕАЛІЗАЦІЯ ЗАХИСТУ ДАНИХ ТА КОНФІДЕНЦІЙНОСТІ В CRM СИСТЕМАХ

Реалізація захисту даних та конфіденційності в CRM-системах стає особливо актуальною в умовах зростання кіберзагроз та посилення нормативних вимог, як-от GDPR. Ключовим аспектом захисту даних є шифрування, яке гарантує, що чутливі дані залишаються недоступними для несанкціонованого доступу. Шифрування може бути реалізовано на різних рівнях – від шифрування даних, що зберігаються в базі даних (атрестація), до шифрування даних, що передаються (SSL/TLS протоколи для захищеного обміну даними).

Управління доступом є ще одним важливим елементом, який передбачає визначення прав та ролей користувачів у системі. Реалізація моделей контролю доступу, як-от RBAC (Role-Based Access Control), дозволяє забезпечити, що користувачі мають доступ лише до тих даних та функцій, які необхідні для їхньої роботи. Аутентифікація та авторизація користувачів зазвичай виконується за допомогою систем управління ідентифікацією, які можуть включати двофакторну аутентифікацію для додаткового рівня безпеки.

Важливим аспектом є також відповідність нормативним вимогам, як-от GDPR, що вимагає не лише захисту даних, але й надання користувачам можливості контролювати свої дані, включаючи право на доступ, виправлення та видалення своєї інформації. Це передбачає реалізацію механізмів для забезпечення прозорості обробки даних, їх зберігання та обробки згідно з законодавчими вимогами.

Також важливим елементом захисту даних у CRM-системах є регулярне проведення оцінки вразливостей та пенетраційного тестування, щоб ідентифікувати та усунути потенційні слабкі місця в системі. Забезпечення безпеки даних – це неодноразовий процес, що вимагає постійного моніторингу, оновлення та вдосконалення системи безпеки.

Таким чином, реалізація захисту даних у CRM-системах вимагає комплексного підходу, який включає в себе технічні, адміністративні та юридичні аспекти для забезпечення належного рівня безпеки та відповідності сучасним стандартам та нормативним вимогам.

Також важливо підкреслити роль систем аудиту та протоколювання для забезпечення прозорості та відстеження дій всередині системи. Імплементація протоколів аудиту, які фіксують всі операції з даними, включаючи доступ, зміни, видалення та експорт, дозволяє отримати докладний запис дій користувачів та системних процесів. Це не лише допомагає виявляти та реагувати на несанкціоновані або підозрілі дії, але й є важливим для відповідності регуляторним вимогам та забезпечення можливості проведення комплексних аудитів.

Ще одним аспектом, який необхідно врахувати, є захист від зовнішніх загроз та кібератак. Використання вогнестінок, систем виявлення та запобігання вторгнень, а також регулярне оновлення програмного забезпечення та операційних систем допомагають забезпечити захист від зовнішніх атак, таких як віруси, троянські програми та інші види шкідливого ПЗ. Заходи з кібербезпеки повинні бути інтегровані на всіх рівнях CRM-системи, від фронтенду до бекенду, для забезпечення максимального захисту.

Врахування аспектів безпеки на етапі проектування системи, включаючи розробку з безпечним кодом та використання перевірених бібліотек та фреймворків,

також є ключовим. Важливо забезпечити, щоб розробники були обізнані з найкращими практиками безпеки та реалізовували їх у своїй роботі. Освітні програми та тренінги з кібербезпеки для розробників і адміністраторів систем можуть відіграти важливу роль у цьому процесі.

Завершуючи, захист даних у CRM-системі є комплексним завданням, яке вимагає постійної уваги, оновлення та вдосконалення заходів безпеки. Забезпечення конфіденційності, цілісності та доступності даних є фундаментальним для довіри клієнтів та стабільної роботи бізнесу. Тому інтеграція ефективних механізмів захисту даних є критично важливою для успішної реалізації та експлуатації CRM-систем.

При реалізації захисту даних у CRM-системах використовуються різні моделі безпеки, кожна з яких має свої унікальні особливості та відповідні сценарії застосування. Role-Based Access Control (RBAC) є однією з найпопулярніших моделей, оскільки вона дозволяє легко управляти правами доступу на основі ролей у компанії, що спрощує розподіл прав доступу та контроль за ними. Також ця модель допомагає відповідати регуляторним вимогам, оскільки легше контролювати, хто має доступ до яких даних. Однак, RBAC може виявитися недостатньо гнучкою в ситуаціях, де потрібен дуже специфічний доступ, і може стати складною при великій кількості ролей та правил.

Discretionary Access Control (DAC), з іншого боку, пропонує більшу гнучкість, оскільки користувачі можуть самостійно управляти доступом до своїх ресурсів. Ця модель особливо підходить для невеликих організацій, де процеси прийняття рішень зосереджені і не потребують складного управління доступом. Однак, DAC несе в собі ризик неналежного управління доступом та відсутності централізованого контролю, що може призвести до проблем з безпекою.

Mandatory Access Control (MAC) надає найвищий рівень безпеки, забезпечуючи строгий контроль доступу на основі централізованих політик. Це робить MAC ідеальною для високозахищених середовищ, де контроль доступу є критично важливим. Такий підхід може обмежувати оперативність бізнес-процесів та вимагає ретельного планування та налаштування.

Attribute-Based Access Control (ABAC) є найбільш адаптивною моделлю, яка дозволяє здійснювати динамічний контроль доступу на основі атрибутів користувачів, ресурсів та умов. Ця модель підходить для умов, де потрібні деталізовані правила доступу, і забезпечує високий рівень гнучкості. Водночас, ABAC може бути складною у налаштуванні та управлінні та потребує більше ресурсів для реалізації та підтримки.

Кожна з цих моделей безпеки має свої унікальні характеристики, які слід враховувати при виборі підходу до захисту даних у CRM-системах. Важливо знайти баланс між необхідним рівнем безпеки, гнучкістю системи та здатністю до масштабування, щоб вибрана модель відповідала специфіці бізнесу та технічним вимогам.

ІНТЕГРАЦІЯ CRM-СИСТЕМИ З ІНШИМИ КОРПОРАТИВНИМИ СИСТЕМАМИ ТА ДОДАТКАМИ

Інтеграція CRM-системи з іншими корпоративними системами та додатками є ключовим аспектом, що забезпечує цілісність бізнес-процесів та підвищує ефективність управління взаєминами з клієнтами. Інтеграція дозволяє об'єднати різні потоки даних та процеси, які раніше були ізольованими, тим самим забезпечуючи більш глибоке розуміння бізнесу та оптимізацію взаємодії з клієнтами.

ERP (Enterprise Resource Planning) системи, які управляють внутрішніми ресурсами компанії, такими як фінанси, логістика та людські ресурси, при інтеграції з CRM дозволяють синхронізувати дані про клієнтів з внутрішніми операціями, покращуючи планування та прийняття рішень. Це також полегшує доступ до інформації про клієнтів для різних відділів і забезпечує більш узгоджене управління взаєминами з клієнтами.

Рис 1421 «Інтеграція додатків в CRM-системи»

Системи SCM (Supply Chain Management), які займаються логістикою та ланцюгами поставок, при інтеграції з CRM можуть значно покращити управління запасами та замовленнями. Така інтеграція допомагає компаніям ефективніше відповідати на потреби клієнтів, оптимізувати рівні запасів та скоротити час обробки замовлень.

Інші бізнес-додатки, які можуть інтегруватися з CRM, включають маркетингові інструменти, інструменти для обробки великих даних, аналітичні інструменти та рішення для управління проектами. Ця інтеграція надає компаніям можливість ефективно використовувати дані для маркетингових кампаній, аналітики клієнтських даних та управління проектами, що залучають клієнтів.

Інтеграція вимагає глибокого розуміння процесів та даних обох систем, включаючи способи обміну даними та інтеграції функціоналу. Це може передбачати

використання API, проміжного програмного забезпечення, а також ретельне планування для забезпечення безперебійної та ефективної роботи інтегрованих систем.

Таким чином, інтеграція CRM з іншими корпоративними системами та додатками є важливим кроком у створенні цілісної інформаційної системи, яка може значно підвищити ефективність бізнес-процесів та якість обслуговування клієнтів.

Інтеграція CRM-системи з іншими корпоративними системами вимагає ретельного планування та виконання, особливо з огляду на сумісність та синхронізацію даних між системами. Для забезпечення ефективної інтеграції необхідно враховувати не тільки технічні аспекти, але й бізнес-процеси, які ці системи підтримують.

Один із ключових аспектів інтеграції – це забезпечення консистентності даних. Коли системи обмінюються даними, важливо гарантувати, що інформація оновлюється в реальному часі або з мінімальною затримкою, щоб уникнути несумісності та помилок. Це може вимагати реалізації складних механізмів синхронізації та контролю за версіями даних.

Іншим важливим фактором є забезпечення безпеки під час обміну даними між системами. Це включає захист від несанкціонованого доступу та забезпечення конфіденційності даних під час їх передачі. Використання шифрованих каналів з'єднання, таких як SSL/TLS, та реалізація механізмів аутентифікації та авторизації для доступу до API, є стандартними практиками в таких випадках.

Розгляд різних архітектурних підходів до інтеграції також є ключовим. Наприклад, використання ESB (Enterprise Service Bus) або інших інтеграційних платформ може надати гнучкість і підтримку для інтеграції великої кількості різноманітних систем. Однак, такі підходи можуть бути складнішими та дорожчими в реалізації та підтримці.

Завершуючи, інтеграція CRM з іншими корпоративними системами та додатками є складним, але критично важливим завданням, що вимагає комплексного підходу та врахування як технічних, так і бізнес-аспектів. Правильно виконана інтеграція може значно підвищити ефективність роботи системи,

покращити обробку та аналіз даних, а також сприяти більш ефективному прийняттю рішень у бізнесі.

Вибір технологій для реалізації CRM-системи є вирішальним фактором, який впливає на функціональність, гнучкість, масштабованість та вартість проекту. При виборі технологій слід враховувати кілька ключових аспектів, включаючи специфіку бізнес-процесів, потреби у масштабуванні, вимоги до безпеки даних, а також інтеграцію з іншими системами та додатками.

Як правило, основою CRM-системи служить база даних, яка забезпечує зберігання, організацію та обробку великих обсягів даних. Вибір бази даних повинен базуватися на таких параметрах, як продуктивність, масштабованість, надійність та легкість інтеграції. Серед популярних варіантів можна розглядати SQL-орієнтовані системи, такі як MySQL або PostgreSQL, а також NoSQL-рішення, наприклад MongoDB або Cassandra, які можуть бути кращим вибором для неструктурованих даних та горизонтального масштабування.

Для розробки бекенду CRM-системи важливо вибрати мову програмування та фреймворки, які забезпечать потрібний рівень гнучкості, продуктивності та легкості підтримки. Java, .NET, Python або Node.js є популярними виборами залежно від вимог проекту та досвіду команди розробників. Фреймворки, такі як Spring для Java або Django для Python, можуть спростити розробку, пропонуючи готові компоненти та бібліотеки.

Для фронтенду CRM-системи важливо забезпечити привабливий, інтуїтивно зрозумілий та відповідний інтерфейс. Технології, такі як HTML5, CSS3 та JavaScript, є стандартом для веб-розробки. Фреймворки, як-от React, Angular або Vue.js, можуть надати додаткову гнучкість та продуктивність при створенні динамічних веб-інтерфейсів.

Не менш важливою є підтримка мобільних пристроїв, оскільки доступ до CRM-системи зі смартфонів та планшетів є критично важливим для сучасного бізнесу. Використання адаптивного дизайну або створення нативних мобільних

додатків для iOS та Android може забезпечити необхідну мобільність та зручність для користувачів.

Нарешті, важливо врахувати аспекти інтеграції з іншими системами, включаючи ERP, SCM та інші корпоративні додатки. Вибір технологій, які підтримують стандартні інтерфейси для інтеграції, такі як REST або SOAP API, може спростити цей процес.

В цілому, вибір технологій для реалізації CRM-системи повинен базуватися на ретельному аналізі поточних та майбутніх бізнес-потреб, можливостей команди розробників та бюджетних обмежень. Правильно вибрані технології забезпечать стабільну, ефективну та гнучку систему, готову відповідати на виклики сучасного ринку.

Продовжуючи обговорення технологічного стеку для реалізації CRM-системи, важливим аспектом є розуміння взаємодії між різними технологіями та вибір найбільш підходящих інструментів для кожного компоненту системи.

Після вибору бази даних, яка задовольняє потреби в продуктивності, масштабованості та надійності, наступним кроком є вибір технологій для розробки серверної частини. Тут рішення залежить від мови програмування та фреймворків, які найкраще відповідають вимогам та ресурсам проекту. Наприклад, Java може бути вибрана за свою стабільність і велику екосистему, тоді як Python - за свою гнучкість та широкі можливості для роботи з даними.

Фронтенд, або клієнтська частина CRM-системи, вимагає використання технологій, які забезпечать привабливий та інтуїтивно зрозумілий інтерфейс. Технології, як-от HTML5, CSS3 та JavaScript, є основою для будь-якого веб-інтерфейсу, а фреймворки, такі як React, Angular чи Vue.js, пропонують додаткову гнучкість для створення динамічних та відгукових інтерфейсів.

Мобільна підтримка також є критичною складовою, оскільки все більше користувачів використовують мобільні пристрої для доступу до CRM-систем. Розробка нативних додатків для iOS та Android або використання адаптивного веб-дизайну забезпечує доступність CRM-системи на різних пристроях.

Загалом, вибір технологій для реалізації CRM-системи має враховувати баланс

між технічними можливостями, вимогами бізнесу, а також легкістю розробки та підтримки. Важливо також забезпечити, що обрані технології дозволять системі адаптуватися до майбутніх змін та вимог, що є ключовим для довгострокового успіху будь-якого IT-проекту.

2.1. Об’єктно-орієнтовані середовища розробки

Середовища розробки з орієнтацією на об'єктно-орієнтоване програмування (ООП) забезпечують набір інструментів, фреймворків та мов програмування, які підтримують принципи та практики ООП. Вони дозволяють створювати, проектувати та підтримувати програмні системи з використанням об'єктно-орієнтованих методологій. Середовища розробки включають мови програмування, бібліотеки класів, фреймворки, інтегровані середовища розробки, інструменти налагодження та тестування, інструменти документації та співпраці.

Вони дозволяють розробникам ефективно створювати складні програмні рішення, використовуючи об'єктно-орієнтовані підходи. Також, вони забезпечують потужність і гнучкість для моделювання реальних об'єктів, створення класів, спадкування, поліморфізму та інших принципів ООП. Використання таких середовищ дозволяє розробникам писати чистий, структурований і легко зрозумілий код, що сприяє підтримці, розширенню та змінам в програмному забезпеченні.

Крім того, середовища розробки з орієнтацією на об'єктно-орієнтоване програмування підтримують широкий спектр інструментів і функціональності, таких як компоненти користувацького інтерфейсу, доступ до баз даних, мережеві можливості та робота з файлами. Це дозволяє розробникам зосередитися на високорівневій логіці програми, не вдаючись в деталі низькорівневої реалізації.

На рисунку нижче ми можемо побачити найпопулярніші мови програмування, які підтримують парадигму ООП:

Рис. 2.1. Найпоплярніші об’єктно-орієнтовані мови програмування

Загалом, в нових проектах все частіше використовується *Python*, але так як раніше дуже багато проектів було реалізовано на базі *Java* та *C++*, тому популярність саме *Java* все ще випереджає *Python*.

В подальшій роботі буде використовуватись саме *Java*, так як вона має певні переваги, а семе:

- Незалежність від платформи: Програми на *Java* можуть працювати на будь-якій платформі з встановленою *Java Virtual Machine (JVM),* що робить його мовою незалежною від платформи. Після компіляції коду *Java* можна виконувати на різних операційних системах без будь-яких змін.

- Надійність та стабільність: *Java* має сувору перевірку типів під час компіляції, обробку виключень та автоматичне керування пам'яттю (сбірка сміття), що сприяє надійності та стабільності програм на *Java*. Це допомагає уникнути поширених помилок програмування та підвищує стабільність програм.

- Безпека: *Java* має вбудовані механізми безпеки, які допомагають захистити програми від поширених вразливостей безпеки. Вона надає "пісочницю" для виконання ненадійного коду, запобігаючи зловживанням та забезпечуючи безпеку

як системи користувача, так і додатка.

- Мультипотоковість та конкурентність: *Java* підтримує мультипотоковість, що дозволяє розробникам створювати багатопотокові програми, які можуть виконувати кілька завдань одночасно. Ця можливість є важливою для створення масштабованих та реагуючих програм, які можуть ефективно використовувати сучасні багатоядерні процесори.

Для Java двома найкращими середовищами розробки є *Eclipse* та *IntelijIDEA*. Ми будемо використовувати саме *IntelijIDEA*, тому що це потужне інтегроване середовище розробки (*IDE*), яке має широкий спектр функціональних можливостей, таких як інтелектуальне автодоповнення коду, високорівневий аналіз коду, зручна навігація та ефективні засоби налагодження. Інтерфейс *IDE* є зрозумілим користувачу, а також підтримується різні фреймворки та технології, що робить його популярним серед розробників. Завдяки великому набору плагінів та потужній інтеграції з системою збірки, *IntelliJ IDEA* надає ефективне та продуктивне середовище для розробки програмного забезпечення.

2.2.Об’єктно-орієнтовані бази даних, *MongoDB*

Розглянемо 2 найпопулярніші не реляційні бази даних, а семе *MongoDB* та *OracleNoSQL*.

MongoDB - це популярна документно-орієнтована база даних з відкритим вихідним кодом, яка призначена для зберігання, пошуку та управління неструктурованими даними. Вона використовує гнучку модель документів, де дані зберігаються в *JSON*-подібних документах з динамічними схемами. *MongoDB* пропонує високу масштабованість і продуктивність, що дозволяє здійснювати горизонтальне масштабування на декількох серверах. Вона також надає широкі можливості запитів, індексування та підтримку різних мов програмування.

MongoDB широко використовується в сучасних веб-додатках, системах управління контентом та аналітиці в режимі реального часу.

Oracle NoSQL Database - це розподілене сховище ключів-значень, оптимізоване для високої продуктивності, масштабованості та доступності. Вона призначена для обробки великих обсягів неструктурованих і напівструктурованих даних. *Oracle NoSQL Database* забезпечує автоматичне розділення та реплікацію даних, гарантуючи їх довговічність та відмовостійкість. Вона підтримує гнучкі моделі даних і дозволяє здійснювати транзакції *ACID*. Ця база даних підходить для таких випадків використання, як профілі користувачів, управління сеансами, каталоги продуктів і рекомендаційні системи.

Для побудови інформаційної системи буде використовуватися саме *MongoDB,* тому що вона краще інтегрується з розподіленими середовищами розробки. Вона є документо-орієнтованою, а це означає, що БД складається з колекцій, в якій містяться різноманітні документи. Однією з переваг зберігання документів, є те, що в кожному документі може бути різна кількість параметрів, через це не буде такої проблеми, як в реляційних базах даних, коли в таблиці можуть залишатися пусті поля. Нижче наведена модель *MongoDB*:

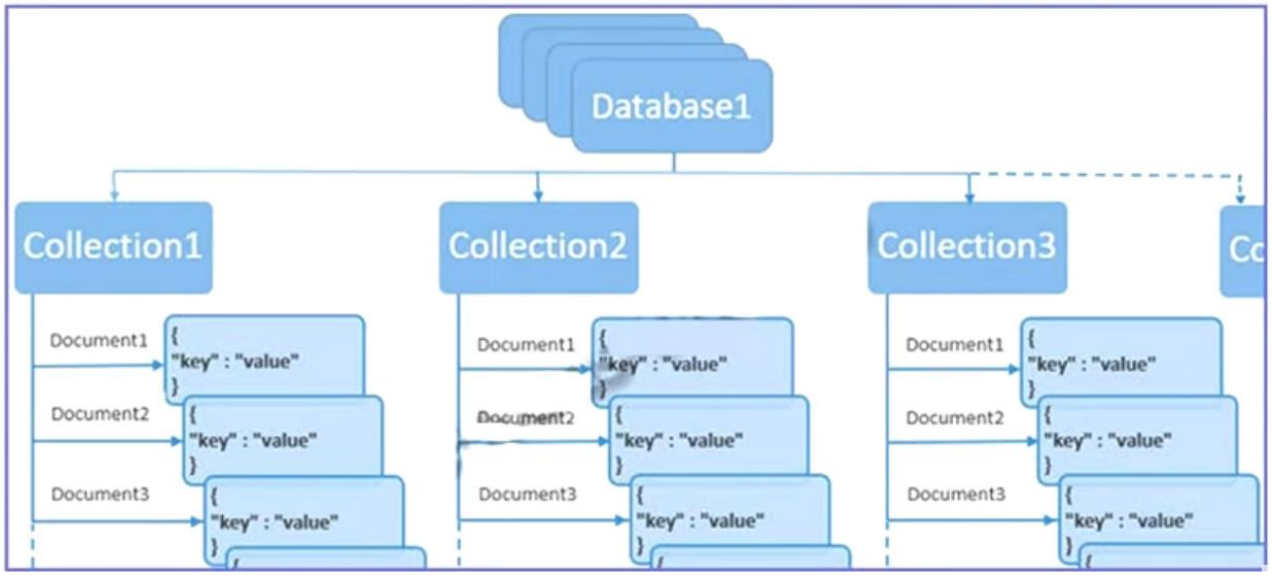


Рис.2.2. Модель документо-орієнтованої бази даних

Встановити дану базу даних можна за посиланням - *https://www.mongodb.com/try/download/community*

Для управління БД можна використовувати або зручний графічний інтерфейс, або ж скористатися терміналом.

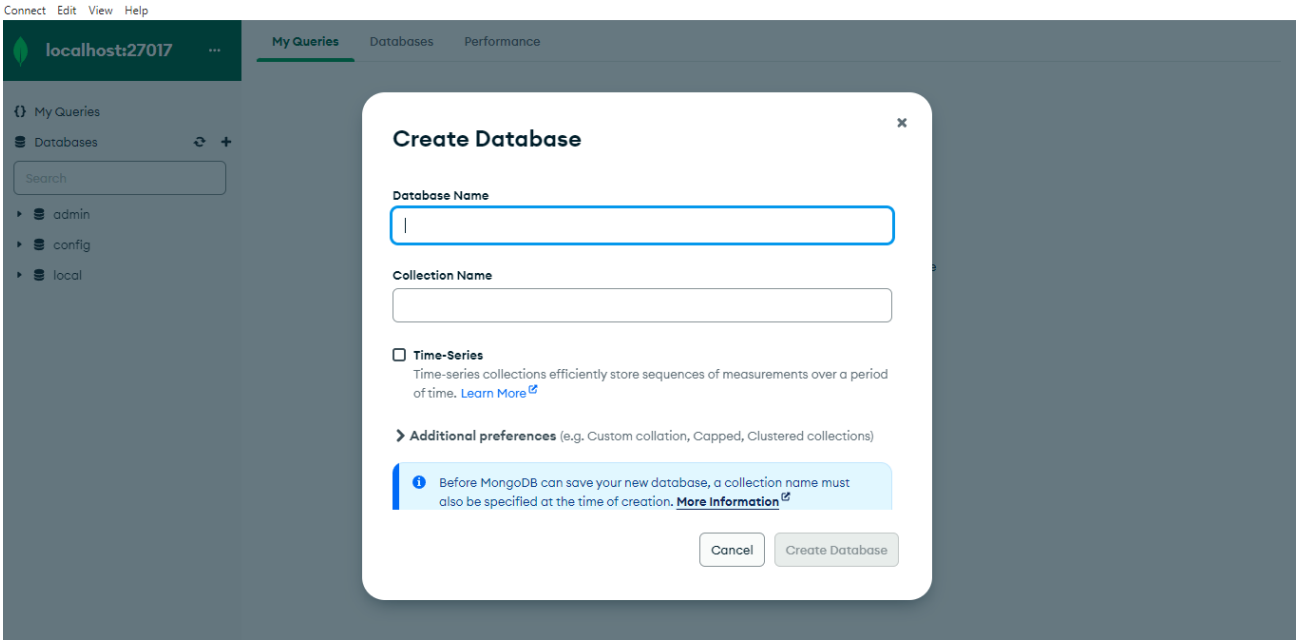


Рис. 2.3. Графічний інтерфейс *MongoDB*

Для зміни поточної БД або для її створення достатньо використати команду в терманалі:

use Database_Name

Для створення колекції потрібно використати метод, який має наступний вигляд:

db.createCollection(name, options)

де *name* – ім'я для колекції, а *options*(необов'язковий параметр) зберігає опції

індексації та ліміту пам’яті.

Операції *CRUD* (створення, читання, оновлення та видалення) в *MongoDB* відносяться до основних операцій, які можна виконувати з даними.

Операція створення включає вставку нових документів у колекцію *MongoDB*. Ви можете створити новий документ, надавши необхідні поля даних та їх значення. *MongoDB* автоматично присвоює унікальний ідентифікатор, відомий як *ObjectID*, кожному документу.

Методи *insert()* та *save()* дозволяють додавати нові документи в колекцію. Також, якщо викликати вставку на неіснуючій колекції, то вона буде створена і в неї буде додано цей документ.

Щоб знайти потрібний документ слід виконати метод *find()*. Також можна опціонально додати виклик методу *pretty()*, що зробить вивід красивішим.

В mongo є два методи для оновлення документів: *update()* та *save()*. в той час як *update()* оновлює існуючий запис, *save()* його заміняє.

Для видалення використовується метод *remove()*. Він приймає два параметра: критерій для вибірки, що буде видалена та прапор justOne (видалити лише перший збіг з вибірки). Щоб видалити колекцію потрібно викликати метод:

db.Collections_Name.drop()

Висновки до розділу

У другому розділі дипломної роботи, було досліджено ключові компоненти та процеси, необхідні для створення ефективної та функціональної системи. Вибір Java та MongoDB для реалізації CRM-системи був зумовлений їхньою високою продуктивністю, гнучкістю та здатністю ефективно обробляти великі обсяги даних, що є важливими факторами для бізнесу, зайнятого у сфері обслуговування дронів.

Протягом другого розділу ми розглянули структуру та архітектуру CRM-системи, зосередившись на таких ключових аспектах, як аналіз вимог до системи, цільової аудиторії, бізнес-процесів, а також моделювання даних у MongoDB. Це дало нам змогу глибше зрозуміти, як система повинна взаємодіяти з користувачами, обробляти дані та відповідати на потреби бізнесу.

Ми також детально розглянули різні підходи до індексації та оптимізації запитів в MongoDB, що є критично важливим для підтримання високої продуктивності системи. Ефективне використання індексів та оптимізація запитів дозволяє CRM-системі швидко обробляти запити та надавати актуальну інформацію, необхідну для прийняття рішень у бізнесі.

На закінчення, ми визначили та описали технологічний стек для реалізації CRM-системи, з акцентом на використання Java та MongoDB. Ця комбінація технологій не лише забезпечує потужний інструментарій для розробки, але й гарантує високу гнучкість та масштабованість системи, що є важливим для адаптації до змінних умов бізнес-середовища та технологічних інновацій.

Завершуючи другий розділ, можна стверджувати, що розробка CRM-системи на базі Java та MongoDB відкриває широкі можливості для компаній, які прагнуть до оптимізації взаємодій з клієнтами та ефективного управління бізнес-процесами, особливо в такій технологічно розвиненій галузі, як обслуговування дронів.

МОДЕЛЬ ДАНИХ У MONGODB ДЛЯ CRM-СИСТЕМИ

У наступному розділі буде представлена детальна модель даних MongoDB для нашої CRM-системи, розглядаючи ключові сутності, такі як Clients, Projects, Tickets, SwUpdates та DroneModels. Кожна сутність має відіграти певну роль у забезпеченні ефективного управління відносинами з клієнтами, розробкою продуктів, технічною підтримкою, а також у відстеженні та оновленні технічних аспектів дронів.

Entity	Key Fields	Description	Usage
Clients	clientId, name, email	Дані про клієнтів	Взаємодії з клієнтами
Projects	projectId, clientReference, status	Інформація про проекти	Управління проектами
Tickets	ticketId, issueDescription, resolutionStatus	Запити на підтримку	Обробка та реагування на запити
SwUpdates	updateId, version, changeLog	Оновлення ПО	Відстеження релізів ПО
DroneModels	modelId, modelName, specifications	Моделі дронів	Технічні характеристики дронів

Сутність Clients в об'єктно-орієнтованій базі даних MongoDB має ключове значення, оскільки вона відіграє центральну роль у CRM-системі, забезпечуючи управління та відстеження інформації про клієнтів. Як документ-орієнтована база даних, MongoDB ідеально підходить для зберігання гнучких схем даних, що є особливо корисним для динамічної інформації, яка може змінюватися або розширюватися з

часом.

Структура сутності Clients:

– clientId (String): Унікальний ідентифікатор клієнта, який використовується для відстеження та посилання на конкретного клієнта в системі. Це поле є ключовим для зв'язування клієнта з іншими сутностями, такими як Projects чи Tickets.

– name (String): Повне ім'я або назва компанії клієнта. Ця інформація використовується для персоналізації взаємодій з клієнтом та може включати як ім'я контактної особи, так і назву організації.

– email (String): Електронна адреса клієнта, яка використовується як основний канал комунікації. Це важливий атрибут для відправки оновлень, звітів або маркетингових матеріалів.

– phone (String): Контактний номер телефону клієнта. Це поле може містити декілька номерів для різних відділів або контактних осіб.

– organization (String): Назва організації клієнта, якщо вона відрізняється від name. Це поле дозволяє зберігати структуровані дані про компанію клієнта.

– interactionHistory (Array): Масив, що містить історію взаємодій клієнта з компанією. Кожен елемент масиву може включати дату взаємодії, тип (зустріч, телефонний дзвінок, електронний лист тощо) та короткий опис.

Сутність Clients в MongoDB дозволяє гнучко взаємодіяти з даними клієнтів, надаючи можливість легко оновлювати, запитувати та аналізувати цю інформацію. Оскільки MongoDB не вимагає жорсткої схеми даних, поля можуть бути додані, видалені або змінені з часом без необхідності переструктурування всієї бази даних, що надає високий рівень адаптивності.

Додатково, масив interactionHistory в Clients відкриває широкі можливості для аналізу взаємодій з клієнтами. Кожен запис у цьому масиві може бути використаний для відстеження історії спілкування, оцінки ефективності комунікаційних стратегій та планування майбутніх взаємодій. Ця інформація є важливою для розуміння

потреб клієнтів та оптимізації бізнес-процесів.

Поле email та phone служать не тільки для прямих контактів, але й можуть бути інтегровані з системами масової розсилки та автоматизованого маркетингу. Це дозволяє створювати цільові кампанії та забезпечувати консистентну комунікацію з клієнтами.

Інформація про organization може бути використана для сегментації клієнтської бази, аналізу ринкових тенденцій та адаптації продуктів та послуг до потреб різних організацій. Це також сприяє кращому розумінню ринкового положення компанії та її взаємодії з різними сегментами клієнтів.

Враховуючи гнучкість MongoDB, Clients може бути легко розширена додатковими полями, такими як preferences, feedbacks, або contractDetails, для подальшого поглиблення клієнтських даних. Це робить сутність Clients не просто сховищем статичної інформації, а динамічним інструментом для управління відносинами з клієнтами, надаючи значний потенціал для бізнес-аналітики та стратегічного планування.

Сутність Projects в MongoDB CRM-системі відіграє ключову роль у відстеженні та управлінні проектами, пов'язаними з розробкою та підтримкою програмного забезпечення для цивільних дронів. Оскільки MongoDB дозволяє створювати гнучкі схеми даних, ця сутність може бути адаптована для задоволення специфічних потреб бізнесу.

Структура сутності Projects:

- projectId (String): Унікальний ідентифікатор проекту, який є основним ключем для відстеження різних проектів у системі. Цей ідентифікатор використовується для зв'язування проектів з іншими сутностями, такими як Tickets та SwUpdates.
- clientReference (String): Посилання на clientId з сутності Clients, що дозволяє встановити зв'язок між проектами та клієнтами. Це забезпечує зручне відстеження того, які проекти виконуються для конкретних клієнтів.
- projectName (String): Назва проекту, що відображає його основну мету

або сферу розробки. Назва проекту важлива для швидкого ідентифікування та класифікації проектів.

- startDate та endDate (Date): Дати початку та завершення проекту. Ці поля дозволяють відстежувати тривалість проектів та планувати ресурси та зусилля.
- budget (Number): Бюджет проекту. Контроль за бюджетом є важливим для управління фінансовими ресурсами компанії та оцінки рентабельності проектів.
- status (String): Поточний статус проекту, наприклад, "В розробці", "На тестуванні", "Завершено". Статус допомагає відслідковувати прогрес проекту та інформувати зацікавлені сторони.

Сутність Projects відіграє центральну роль у координації та управлінні різними аспектами проектів у компанії. Вона дозволяє керівникам проектів, розробникам та іншим учасникам отримувати повний огляд поточного стану проектів, контролювати відхилення від плану та вносити корективи в процеси.

Можливість зв'язування проектів з конкретними клієнтами через clientReference сприяє кращому розумінню вимог клієнтів та налаштуванню проектів згідно з цими вимогами. Це також полегшує комунікацію з клієнтами та надає їм персоналізовані оновлення про стан виконання проектів.

Інформація про дати початку та завершення, бюджет та статус проекту є важливою для ефективного планування ресурсів, фінансового управління та відстеження загальної продуктивності. Використання MongoDB дозволяє легко оновлювати та розширювати ці дані, що є особливо корисним у динамічному та швидкозмінному середовищі розробки ПО.

Сутність Tickets у MongoDB CRM-системі відіграє важливу роль у процесі управління запитами на підтримку від клієнтів. Ця сутність забезпечує централізоване місце для відстеження, аналізу та вирішення проблем, які клієнти можуть зустріти під час використання програмного забезпечення для дронів.

Структура сутності Tickets:

- ticketId (String): Унікальний ідентифікатор тікету, що дозволяє легко відстежувати та керувати конкретним запитом.
- projectId (String): Ідентифікатор проекту, до якого відноситься тікет. Це забезпечує зв'язок між запитами на підтримку та конкретними проектами в Projects.
- clientReference (String): Посилання на clientId з сутності Clients, що вказує на клієнта, який подав тікет.
- issueDescription (String): Детальний опис проблеми або запиту, представлений клієнтом. Ця інформація є ключовою для розуміння та вирішення питання.
- submissionDate (Date): Дата подання тікету, яка важлива для відстеження часу реакції та ефективності обслуговування клієнтів.
- resolutionStatus (String): Поточний статус тікету, наприклад, "В обробці", "Вирішено", "Закрито". Це поле допомагає керувати процесом обробки та відповіді на запити.

Сутність Tickets є невід'ємною частиною CRM-системи, оскільки вона дозволяє ефективно обробляти та управляти взаємодіями з клієнтами, пов'язаними з підтримкою та обслуговуванням. Вона допомагає команді підтримки швидко реагувати на запити, відстежувати їх статус та аналізувати типові проблеми та запити клієнтів.

Інтеграція сутності Tickets з Projects та Clients надає змогу створити 360-градусний огляд взаємодій з клієнтами, забезпечуючи глибоке розуміння контексту кожного запиту. Це дає можливість не тільки вирішувати окремі запити, але й оцінювати загальну задоволеність клієнтів та їх досвід використання продукту.

Окрім цього, Tickets може інтегруватися з системами сповіщень або автоматичного відповідача, що значно підвищує швидкість та ефективність обробки запитів. Автоматизація процесів, таких як первинна класифікація тікетів та надання стандартних відповідей на часті запити, може значно знизити час відгуку та підвищити задоволеність клієнтів.

Сутність SwUpdates в MongoDB CRM-системі відіграє критичну роль у

відстеженні та управлінні оновленнями програмного забезпечення для цивільних дронів. Ця сутність забезпечує централізований доступ до інформації про всі версії ПО, оновлення та їх вплив на різні моделі дронів, що є важливим для забезпечення стабільності та безпеки роботи дронів.

Структура сутності SwUpdates:

- updateId (String): Унікальний ідентифікатор оновлення, який дозволяє легко ідентифікувати та відстежувати конкретне оновлення.
- projectId (String): Ідентифікатор проекту, якому належить оновлення. Це забезпечує зв'язок між оновленням ПО та конкретними проектами, над якими працює компанія.
- releaseDate (Date): Дата випуску оновлення. Це дозволяє відстежувати історію випусків оновлень та планувати майбутні релізи.
- version (String): Версія оновлення ПО. Відстеження версій є ключовим для розуміння еволюції ПО та вирішення питань сумісності.
- changeLog (String або Array): Детальний опис змін, внесених в оновленні. Це може включати виправлення помилок, додавання нових функцій, поліпшення безпеки та інші важливі оновлення.

Сутність SwUpdates відіграє важливу роль у забезпеченні прозорості та контролю над процесом розробки та випуску оновлень ПО. Вона допомагає розробникам, тестувальникам та інженерам підтримки відстежувати актуальність ПО, забезпечувати сумісність з різними моделями дронів та оперативно реагувати на виявлені проблеми.

Використання MongoDB для цієї сутності дозволяє легко організувати та зберігати великі обсяги даних про оновлення. Можливості агрегації та фільтрації даних в MongoDB сприяють швидкому пошуку та аналізу оновлень, дозволяючи здійснювати комплексний аналіз впливу оновлень на продукти та послуги.

Інформація з changeLog є особливо цінною для забезпечення якості та безпеки продуктів. Відстеження змін дозволяє виявляти можливі ризики, оцінювати ефективність внесених поліпшень та забезпечувати відповідність вимогам

стандартів та регуляцій.

Сутність DroneModels у MongoDB CRM-системі є важливою для управління інформацією про різні моделі цивільних дронів, з якими працює компанія. Ця сутність дозволяє компанії зберігати детальні специфікації, відомості про сумісність з програмним забезпеченням та іншу технічну інформацію, необхідну для розробки, тестування та підтримки ПО для дронів.

Структура сутності DroneModels:

- modelId (String): Унікальний ідентифікатор моделі дрона, який використовується для ідентифікації та відстеження різних моделей.
- modelName (String): Назва моделі дрона, яка допомагає швидко ідентифікувати конкретну модель серед широкого асортименту.
- specifications (String або Embedded Document): Технічні специфікації моделі, включаючи такі параметри, як розмір, вага, час польоту, радіус дії, типи сенсорів тощо. Ці дані можуть бути структуровані як вбудований документ для забезпечення кращої організації.
- compatibleSoftwareVersions (Array): Список версій програмного забезпечення, сумісних з даною моделлю дрона. Це поле допомагає забезпечити правильну сумісність ПО та запобігти потенційним проблемам під час експлуатації.
- notes (String): Додаткові примітки або важливі відомості про модель, такі як особливі інструкції, рекомендації щодо використання або історія оновлень.

Сутність DroneModels є фундаментальною для роботи компанії, яка розробляє ПО для дронів, оскільки вона дозволяє зосередитися на конкретних моделях та їх особливостях. Відстеження технічних специфікацій та сумісності з програмним забезпеченням є важливим для гарантування, що розроблене ПО буде ефективно функціонувати з різними моделями дронів.

Використання MongoDB для цієї сутності забезпечує необхідну гнучкість у зберіганні та організації різноманітної технічної інформації. Можливість розширювати та оновлювати дані моделей без необхідності перепроектування всієї бази даних є ключовим аспектом для динамічного сектору, яким є розробка ПО для

дронів.

Сутність DroneModels також може бути інтегрована з іншими сутностями, такими як Projects та SwUpdates, для забезпечення цілісного погляду на процес розробки та підтримки ПО. Це дозволяє розробникам та менеджерам проектів краще планувати роботу, враховуючи специфічні вимоги та характеристики різних моделей дронів.

Завдяки гнучкості та масштабованості MongoDB, DroneModels може легко адаптуватися до нових вимог ринку, включаючи впровадження нових технологій, оновлення характеристик та розширення асортименту дронів. Це робить сутність надзвичайно важливою для підтримки інновацій тазбереження конкурентоспроможності в галузі розробки ПО для дронів.

Окрім технічних характеристик, сутність DroneModels може бути використана для аналізу трендів та вподобань на ринку дронів. Через поля, як-от notes та compatibleSoftwareVersions, можливо збирати детальну інформацію про попит на певні моделі, проблеми сумісності, які виникають при розробці, та відгуки користувачів щодо певних моделей дронів. Це надає цінний внесок у стратегічне планування розвитку продукту та інновації.

Інтеграція DroneModels з іншими сутностями, такими як Projects та SwUpdates, дозволяє розробникам та технічним спеціалістам бачити повну картину роботи з конкретними моделями дронів. Наприклад, вони можуть аналізувати, які моделі дронів найчастіше використовуються у проектах, які версії ПО сумісні з цими моделями, та які оновлення ПО були найефективнішими.

Крім того, DroneModels може бути інтегрована з зовнішніми джерелами даних, наприклад, з базами даних постачальників або з інформаційними системами ринкових аналітиків, що дозволяє компанії відстежувати новітні тенденції у використанні дронів та швидко адаптуватися до змін у ринкових умовах.

СТРУКТУРА ДОКУМЕНТІВ, ІНДЕКСАЦІЯ ТА ОПТИМІЗАЦІЯ ЗАПИТІВ

Для ілюстрації структури документів та колекцій у MongoDB для CRM-системи компанії, важливо розглянути, як конкретно дані організовані та як вони взаємодіють у рамках бази даних. Ось детальний опис структури кожної колекції:

Документи в колекції Clients містять всю необхідну інформацію про клієнтів. Типовий документ у цій колекції може виглядати так:

```
{
  "_id": ObjectId("..."),
  "clientId": "C1001",
  "name": "John Doe",
  "email": "johndoe@example.com",
  "phone": "123-456-7890",
  "organization": "Doe Drones",
  "interactionHistory": [
    {
      "date": "2023-01-01",
      "type": "Email",
      "details": "Запит на інформацію про нові моделі дронів"
    },
```

Рис 332.

Кожен документ у колекції Projects представляє окремий проект, над яким працює компанія. Приклад документа:

```
{
  "_id": ObjectId("..."),
  "projectId": "P2002",
  "clientReference": "C1001",
  "projectName": "Розробка навігаційного ПЗ для дронів",
  "startDate": "2023-02-01",
  "endDate": "2023-08-01",
  "status": "В розробці"
}
```

Рис 32424.

Ця колекція використовується для відстеження запитів на підтримку від клієнтів. Приклад структури документа:

```
{
  "_id": ObjectId("..."),
  "ticketId": "T3003",
  "projectId": "P2002",
  "clientReference": "C1001",
  "issueDescription": "Проблема з оновленням ПЗ дрона",
  "submissionDate": "2023-03-10",
  "resolutionStatus": "В обробці"
}
```

Рис 2342432

Документи в цій колекції містять інформацію про оновлення програмного забезпечення. Приклад:

```
{
  "_id": ObjectId("..."),
  "updateId": "U4004",
  "projectId": "P2002",
  "releaseDate": "2023-04-15",
  "version": "v2.1",
  "changeLog": "Додано підтримку нових GPS модулів"
}
```

Рис 324234

Кожен документ у цій колекції представляє специфікації та інформацію про модель дрона. Приклад документа:

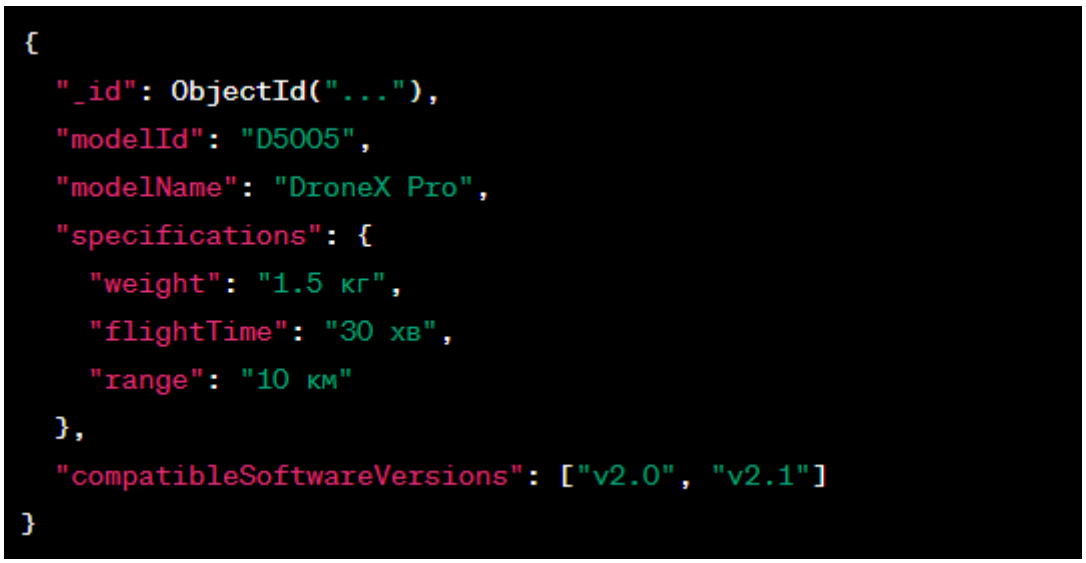


Рис. 23423

Ці колекції відображають основну структуру даних CRM-системи і є фундаментом для її функціонування. MongoDB дозволяє легко керувати цими документами, забезпечуючи гнучкість у внесенні змін, швидкий доступ до даних та ефективне відстеження взаємодій з клієнтами та управління проектами.

Розгляд підходів до індексації та оптимізації запитів в MongoDB для CRM-системи, що обслуговує дрони, є критично важливим для забезпечення високої продуктивності та ефективності системи. Індексація в MongoDB виконує ключову роль у підтримці швидкості та ефективності запитів, особливо коли мова йде про великі обсяги даних та складні запити, що типові для CRM-систем.

Індексація може бути наступною:

– Стандартна індексація:

MongoDB автоматично створює унікальний індекс на поле _id для кожної колекції, що забезпечує швидке відстеження та пошук документів. Додаткові індекси можуть бути створені на часто використовувані поля, наприклад, clientId в Clients або projectId в Projects, для оптимізації запитів, які використовують ці поля.

– Композитні індекси:

Композитні індекси в MongoDB можуть бути створені для оптимізації запитів, які використовують кілька полів. Наприклад, індекс, що включає clientId та startDate

в Projects, може покращити продуктивність запитів для відстеження проектів певного клієнта за певний період.

– Індекси для текстового пошуку:

Для поліпшення пошуку за текстовими полями, такими як issueDescription в Tickets, можуть бути використані текстові індекси. Це полегшує виконання складних запитів на пошук за ключовими словами.

– Індекси для сортування:

Індекси, оптимізовані для сортування, важливі для запитів, які включають операції sort(), наприклад, при сортуванні проектів за startDate або тікетів за submissionDate.

Оптимізація Запитів:

– Використання $explain для аналізу запитів:

MongoDB надає оператор $explain, який може бути використаний для аналізу виконання запитів. Це дозволяє зрозуміти, які індекси використовуються, та оцінити ефективність запитів.

– Оптимізація плану виконання запитів:

Аналізуючи плани виконання, можна визначити, чи потрібно модифікувати індекси для забезпечення кращої продуктивності, особливо для складних запитів з декількома умовами та операціями.

– Пагінація результатів:

Для запитів, що повертають великі набори даних, використання пагінації (наприклад, через limit() та skip()) може значно знизити навантаження на систему та покращити час відгуку.

– Використання агрегацій:

Агрегаційні пайплайни в MongoDB можуть бути оптимізовані для ефективного об'єднання, фільтрації та перетворення даних, що особливо важливо для аналітичних запитів та звітів.

Завдяки цим підходам до індексації та оптимізації запитів, CRM-система

забезпечує високу продуктивність та надійність у роботі з даними, що є важливим для компаній, що займаються обслуговуванням та розробкою ПО для дронів.

У контексті саме поточної системи, оптимізація запитів іноді може вимагати відступу від стандартних підходів, особливо коли мова йде про обробку великих масивів даних або реалізацію складних бізнес-логік. Наприклад, вибірка даних про всі інтеракції певного клієнта з системою може включати запити до декількох колекцій, як-от Clients, Projects, і Tickets. В таких випадках, використання агрегаційних пайплайнів дозволяє ефективно об'єднувати та фільтрувати дані з різних джерел, оптимізуючи час виконання запитів.

Крім того, важливим аспектом оптимізації запитів є регулярний моніторинг та аналіз використання ресурсів бази даних. Це включає відстеження часу відгуку запитів, використання пам'яті та CPU, а також частоту запитів до певних колекцій. Цей аналіз допомагає виявити "вузькі місця" в системі та вчасно вносити корективи, наприклад, переглядаючи індексацію або реорганізуючи структуру даних.

Особливу увагу слід приділити оптимізації запитів, пов'язаних з аналітикою та звітністю. Звіти можуть вимагати комплексних запитів до бази даних, що інколи включають агрегацію великих обсягів даних. Використання оптимізованих агрегаційних пайплайнів та інтелектуальне кешування результатів може значно підвищити продуктивність таких запитів.

Нарешті, важливо забезпечити, що система має гнучкість для адаптації до змін у бізнес-процесах компанії. Це може включати додавання нових типів даних, зміну схеми документів, або впровадження нових типів запитів, що вимагає регулярного перегляду та оновлення індексації та стратегій оптимізації.

Завдяки цим підходам, MongoDB забезпечує ефективне та гнучке управління даними в CRM-системі, підтримуючи високий рівень продуктивності та адаптивності до вимог бізнесу.

АРХІТЕКТУРА СИСТЕМИ, СТЕК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Архітектура CRM-системи, базується на клієнт-серверній моделі, яка забезпечує міцну основу для ефективного взаємодії та управління даними. Ця модель дозволяє розділити відповідальності між клієнтським інтерфейсом (front-end) та серверною логікою (back-end), забезпечуючи високий рівень модульності, гнучкості та масштабованості.

Клієнтська Частина (Front-End):

– Клієнтська частина розроблена з використанням сучасних веб-технологій, таких як HTML5, CSS3, і JavaScript фреймворків, таких як React або Angular. Це забезпечує інтуїтивно зрозумілий та відгуковий інтерфейс, який дозволяє користувачам легко взаємодіяти з системою, виконувати запити, переглядати звіти, та управляти даними.

– Клієнтська частина взаємодіє з сервером через визначений API, зазвичай реалізований за допомогою REST або GraphQL. Це забезпечує стандартизований спосіб запитів та отримання даних, що дозволяє легко інтегрувати різні сервіси та джерела даних.

Серверна Частина (Back-End):

– Серверна частина відповідає за обробку запитів від клієнтів, виконання бізнес-логіки, інтеграцію з базою даних та зовнішніми сервісами. Розроблена з використанням мов програмування, таких як Java, Python або Node.js, та відповідних фреймворків.

– MongoDB використовується як основна система управління базами даних, забезпечуючи гнучкість, високу продуктивність та легкість масштабування. Дані організовані у вигляді документів та колекцій, що дозволяє ефективно обробляти великі обсяги даних, характерних для CRM-систем.

– Бекенд надає API для взаємодії з клієнтською частиною, а також

інтегрується з різними зовнішніми сервісами, такими як системи електронної пошти, інструменти аналітики, сервіси хмарного сховища тощо.

– Система може бути реалізована з використанням мікросервісної архітектури, де окремі компоненти системи (наприклад, управління користувачами, обробка запитів, аналітика) розроблені як незалежні мікросервіси. Це забезпечує високий рівень гнучкості, оскільки кожен мікросервіс може бути розгорнутий, оновлений, та масштабований незалежно від інших.

Інтеграція MongoDB у CRM-систему, розроблену на Java, вимагає ретельного підходу до підключення бази даних, обробки даних та взаємодії з нею, особливо у випадку системи, призначеної для компанії, яка займається обслуговуванням дронів. MongoDB використовується як основна база даних завдяки її гнучкості, продуктивності та масштабованості, що ідеально підходить для динамічної природи даних у CRM-системах.

Для підключення MongoDB до Java-проекту необхідно включити залежності MongoDB Java Driver у файлі залежностей проекту, такому як pom.xml для Maven. Створення підключення до MongoDB вимагає конфігурації параметрів з'єднання, включаючи адресу сервера, порт, ім'я бази даних та параметри автентифікації. Для взаємодії з базою даних створюється сесія, через яку виконуються запити до бази, включаючи створення, читання, оновлення та видалення даних.

Ключовою частиною є розробка методів для виконання CRUD операцій, використовуючи MongoDB Java Driver, що дозволяє ефективно маніпулювати даними. Ці методи можуть включати виконання запитів на пошук, додавання нових документів, оновлення існуючих, та видалення документів з бази даних. Обробка запитів та відповідей, включаючи обробку помилок і виключних ситуацій, є важливою для забезпечення стабільності та надійності системи. Оптимізація запитів, зокрема використання індексів, агрегації даних, та пагінації, є необхідною для ефективного виконання запитів.

Для створення коду підключення бази даних MongoDB до проекту Java в IntelliJ IDEA та реалізації CRUD операцій для сутностей CRM-системи, можна

використати наступні кроки та код. Для початку підключимо залежність для Maven.

```
<dependencies>
    <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongodb-driver-sync</artifactId>
        <version>4.2.3</version>
    </dependency>
</dependencies>
```

Рис. 2132

Після чого створимимо клас для управління підключеннями та CRUD операціями:

```
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;
import org.bson.Document;
public class MongoDBConnector {
   private MongoDatabase database;
   public MongoDBConnector() {
   MongoClient mongoClient =
MongoClients.create("mongodb://localhost:27017");
      database = mongoClient.getDatabase("crmDB");
   }
  public MongoCollection<Document> getCollection(String collectionName) {
      return database.getCollection(collectionName);
   }
   public void createDocument(String collectionName, Document doc) {
```

```
        MongoCollection<Document> collection = getCollection(collectionName);
        collection.insertOne(doc);
    }
        public Document readDocument(String collectionName, Document query) {
        MongoCollection<Document> collection = getCollection(collectionName);
        return collection.find(query).first();
    }
        public void updateDocument(String collectionName, Document query,
Document update) {
        MongoCollection<Document> collection = getCollection(collectionName);
        collection.updateOne(query, new Document("$set", update));
    }
        public void deleteDocument(String collectionName, Document query) {
        MongoCollection<Document> collection = getCollection(collectionName);
        collection.deleteOne(query);
    }
}
```

У цьому коді MongoDBConnector - це клас для управління підключенням до MongoDB. Методи createDocument, readDocument, updateDocument та deleteDocument демонструють основні CRUD операції для роботи з колекціями. Вони використовують клас Document з бібліотеки MongoDB для створення та маніпулювання документами в базі даних.

Після цього можна створити окремі класи або методи для кожної з сутностей CRM-системи (Clients, Projects, Tickets, SwUpdates, DroneModels), використовуючи MongoDBConnector для взаємодії з відповідними колекціями бази даних.

Для прописування сутностей у базу даних у проекті, який використовує MongoDB та Java, потрібно реалізувати моделі даних, які відображають сутності CRM-системи, та методи для взаємодії з цими даними. Нижче я наведу приклад того, як можна структурувати ці моделі та реалізувати основні CRUD операції.

Кроки для Реалізації Сутностей:

– Створення класів моделей:

– Для кожної сутності (наприклад, Client, Project, Ticket) створюється відповідний клас у Java, який відображає поля, необхідні для цієї сутності.

– Використання анотацій для MongoDB:

– Якщо використовується ORM (Object-Relational Mapping) бібліотека, така як Spring Data MongoDB, можна використовувати анотації для вказівки властивостей моделей, які мають бути збережені в базі даних.

– Реалізація CRUD Операцій:

– Реалізація методів для створення, читання, оновлення та видалення даних для кожної сутності.

Після створення класів моделей для кожної сутності, таких як Client, Project, Ticket, наступним кроком є визначення відносин між цими сутностями. Наприклад, клас Project може мати поле, що вказує на clientId, асоціюючи проект з конкретним клієнтом. Це дозволяє легко отримувати інформацію про всі проекти, пов'язані з певним клієнтом. Також важливо реалізувати логіку валідації даних у цих моделях для забезпечення коректності та цілісності даних, що зберігаються в базі даних. Для цього можна використовувати вбудовані можливості бібліотеки ORM або реалізувати власну логіку перевірки в моделях. Всі ці кроки сприяють створенню міцної основи для роботи CRM-системи, що дозволяє ефективно обробляти та управляти даними клієнтів та проектів.

```java
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "clients")
public class Client {
    @Id
    private String id;
    private String name;
    private String email;
    // Інші поля та конструктори, геттери, сеттери

    // Конструктор, геттери та сеттери
}

@Document(collection = "projects")
public class Project {
    @Id
    private String id;
    private String name;
    private String clientId;
    // Інші поля та конструктори, геттери, сеттери

    // Конструктор, геттери та сеттери
}
```

Рис.

Аналогічно для інших сутностей.

Для реалізації класу в рамках Spring Framework, який відповідатиме за керування CRM-системою, ми можемо створити сервісний клас, що використовується для бізнес-логіки та взаємодії з базою даних. Цей клас буде включати методи для управління даними клієнтів, проектів, заявок тощо. Оскільки Spring Framework дотримується принципів інверсії контролю та впровадження залежностей, клас буде інтегровано з іншими компонентами системи через ін'єкцію залежностей.

import org.springframework.stereotype.Service;

import org.springframework.beans.factory.annotation.Autowired;

import com.example.crm.repository.ClientRepository;

import com.example.crm.model.Client;

```
@Service
public class CrmManagementService {
private final ClientRepository clientRepository;
@Autowired
 public CrmManagementService(ClientRepository clientRepository) {
      this.clientRepository = clientRepository;
 }
  public Client getClientById(String id) {
      return clientRepository.findById(id).orElseThrow(() -> new
ResourceNotFoundException("Client not found"));
   }
   public Client createClient(Client client) {
      return clientRepository.save(client);
    }
}
```

У цьому прикладі CrmManagementService є сервісним класом, який відповідає за управління клієнтами у CRM-системі. Клас марковано анотацією @Service, яка вказує Spring, що цей клас є кандидатом для створення bean'у (компоненту Spring). Залежності, такі як ClientRepository, ін'єктуються через конструктор, що забезпечує розділення відповідальності та легше тестування.

Методи getClientById, createClient та інші демонструють реалізацію типових CRUD операцій. Вони використовують ClientRepository для взаємодії з базою даних і можуть включати додаткову логіку, таку як валідацію даних або обробку винятків.

Цей клас може бути розширений для включення додаткових методів для управління іншими сутностями системи, такими як проекти або заявки, забезпечуючи централізоване та єдине місце для бізнес-логіки CRM-системи.

Висновки до роботи

У ході цієї роботи була розглянута тема розробки інформаційної системи на основі об'єктно-орієнтованої бази даних (ООБД) для компанії, яка займається обслуговуванням дронів. Робота включала в себе детальний аналіз архітектури ООБД, переваг і недоліків використання цієї технології, а також розгляд питань інтеграції ООБД в інформаційні системи та розвитку інформаційних систем з часом.

У ході дослідження було виявлено, що використання об'єктно-орієнтованої бази даних може принести значні переваги в розробці CRM-системи для обслуговування дронів. ООБД дозволяють зберігати і керувати даними в більш структурованому та ефективному способі, а також спрощують роботу з складними об'єктами та взаємозв'язками між ними.

За допомогою об'єктно-орієнтованої бази даних можливо покращити продуктивність та масштабованість інформаційної системи, а також забезпечити більшу гнучкість та можливість розширення в майбутньому. Важливо враховувати інтеграцію ООБД в існуючу інфраструктуру та забезпечити безпеку даних.

У роботі була розглянута модель ООБД та її вплив на розробку інформаційної системи. Детальний аналіз архітектури ООБД дозволив підкреслити її роль у забезпеченні ефективного функціонування CRM-системи.

Загалом, результати дослідження свідчать про можливість успішного використання об'єктно-орієнтованої бази даних у розробці інформаційних систем. Вона може сприяти покращенню роботи компанії, оптимізувати процеси та забезпечити зручний інтерфейс для користувачів. Таким чином, інформаційна система на основі ООБД може бути важливим інструментом для підвищення конкурентоспроможності компанії в сучасному бізнес-середовищі.

У цьому дослідженні було розглянуто роль та значення об'єктно-орієнтованих баз даних (ООБД) в розробці інформаційної системи (ІС) для обслуговування дронів. ООБД виявилися важливими компонентами, що дозволяють забезпечити не тільки ефективну організацію та зберігання даних, але й гнучкість та розширюваність системи з плином часу.

ООБД спрямовані на моделювання об'єктів та відносин між ними, де кожен об'єкт може мати свої властивості та методи. Ця підхід робить інформаційну систему більш абстрактною та гнучкою, дозволяючи легко додавати нові об'єкти та змінювати їх структуру без необхідності редагувати велику кількість коду. Модель ООБД може бути створена з використанням класів, спадкування, поліморфізму та інших об'єктно-орієнтованих концепцій.

Однією з ключових переваг використання ООБД є можливість використовувати транзакції для забезпечення цілісності та консистентності даних. Це особливо важливо в інформаційних системах, які обробляють великі обсяги даних та вимагають точного керування доступом до них. Багаторівнева архітектура ООБД дозволяє відокремлювати даних від програмного коду, що спрощує розробку та підтримку системи.

Щодо недоліків ООБД, слід відзначити, що вони можуть бути вимогливими до ресурсів, особливо при обробці великої кількості даних або при великому обсязі одночасних запитів. Також розробка системи на основі ООБД може вимагати додаткового часу та зусиль для створення відповідної моделі даних та взаємозв'язків між об'єктами.

Інтеграція ООБД в інформаційну систему може бути складним завданням, особливо якщо вже існують інші системи з реляційними базами даних. Проте, існують різні технології та підходи для забезпечення взаємодії між ООБД та іншими джерелами даних.

Загалом, ця дослідницька робота відкрила можливості використання об'єктно-орієнтованих баз даних у розробці CRM-систем. Вони можуть забезпечити гнучкість, швидкість та ефективність в обробці та зберіганні даних, що робить їх важливим інструментом у сфері авіаційного обслуговування. Дана робота є вагомим внеском у розвиток області інформаційних систем та ООБД, і може служити основою для подальших досліджень та реалізацій в цій галузі.

Далі, розглядаючи перспективи розвитку інформаційних систем на основі об'єктно-орієнтованих баз даних, важливо відзначити, що ця технологія не стоїть на

місці. З плином часу ООБД стають більш потужними та функціональними, що відкриває нові можливості для розробки інформаційних систем. Вдосконалення алгоритмів оптимізації, підтримка розподілених систем та можливість інтеграції з сучасними технологіями штучного інтелекту та аналізу даних роблять ООБД потужним інструментом для вирішення різноманітних завдань.

Інтеграція ООБД в інформаційні системи також відкриває шлях для розширення функціональності та підвищення продуктивності. Можливість здійснювати аналіз даних в режимі реального часу, прогнозування та оптимізація рішень, а також підтримка різних форматів даних дозволяють інформаційним системам стати більш адаптивними та конкурентоспроможними на ринку.

У світлі зростаючого обсягу даних та зростаючих вимог до їх обробки та зберігання, використання ООБД стає більш актуальним та необхідним. Вони дозволяють зберігати структуровану та навіть напівструктуровану інформацію в більш організованому та ефективному способі.

У заключенні, в даній науковій роботі було проведено детальний аналіз використання об'єктно-орієнтованих баз даних в розробці CRM-системи для обслуговування дронів. Виявлені переваги та недоліки цієї технології, а також визначено її потенціал для майбутнього розвитку інформаційних систем. ООБД є потужним інструментом, що може забезпечити ефективну роботу інформаційних систем та сприяти подальшому розвитку сфери обслуговування дронів.

У даній дослідницькій роботі для розробки інформаційної системи на основі об'єктно-орієнтованої бази даних було обрано мову програмування Java та базу даних MongoDB. Вибір Java обґрунтовувався його великою популярністю в сфері розробки програмного забезпечення, високою надійністю та переносимістю коду між різними платформами. Ця мова програмування також надає зручний інструментарій для роботи з об'єктами та обробки даних, що робить її ідеальним вибором для розробки системи на основі об'єктно-орієнтованої бази даних.

MongoDB була обрана як база даних через її особливості, спрямовані на збереження та обробку напівструктурованих даних, які часто зустрічаються в

області обслуговування дронів. Вона володіє гнучкістю та можливістю швидкого масштабування, що важливо для обробки великої кількості даних, які генеруються дронами.

У рамках дослідження були детально описані та продемонстровані сутності та їхні зв'язки в розробленій системі. Використовуючи об'єктно-орієнтований підхід, було розроблено модель даних, що включала в себе сутності, такі як "Дрон", "Клієнт", "Замовлення" та інші, а також визначено їхні взаємозв'язки. Ця модель дозволила ефективно організувати та зберігати дані про обслуговування дронів, а також виконувати операції з ними у системі.

У цілому, обрані технології та модель даних дозволили створити ефективну та гнучку інформаційну систему для обслуговування дронів на основі об'єктно-орієнтованої бази даних. Ця система може ефективно обробляти та аналізувати дані, забезпечувати високу доступність та масштабованість, і готова до подальшого розвитку та розширення функціональності у вимогам сучасного ринку обслуговування дронів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. К. Рамамурті, Г. Хелд: *Object-Oriented Database Systems: Concepts and Architectures*, 1993. – 280с.

2. Документація *MongoDB* [Електронний ресурс] – Режим доступу: *https://docs.mongodb.com/*

3. Ян Леннокс: *Object-Oriented Database Design Clearly Explained*, 2003. – 384с.

4. Документація *Oracle NoSQL Database* [Електронний ресурс] – Режим доступу: *https://docs.oracle.com/cd/NOSQL/html/index.html*

5. Вільяма Г. Гейтса: *Object-Relational Database Development: A Plumber's Guide*, 1996. – 512с.

6. Клайв Небел: *Object Databases: An Introduction*, 2008. – 260с.

7. Майкл В. Мані: *Object-Relational DBMSs: The Next Great Wave*, 1996. – 216с.

8. Дуглас К. Барроуса: *The Object Database Handbook: How to Select, Implement, and Use Object-Oriented Databases*, 1996. – 352с.

9. Документація *Neo4j Graph Database* [Електронний ресурс] – Режим доступу: *https://neo4j.com/docs/*