UNICHECK
by Turnitin

Ім'я користувача:
приховано налаштуваннями конфіденційності

ID перевірки:
1015645767

Дата перевірки:
19.06.2023 14:13:43 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
19.06.2023 14:14:05 EEST

ID користувача:
100011372

Назва документа: **Попович Олег ОК42**

Кількість сторінок: 42 **Кількість слів:** 6890 **Кількість символів:** 51027 **Розмір файлу:** 1.23 MB ID **файлу:** 1015291736

14.5%
Схожість

Найбільша схожість: **3.06%** з **джерелом з Бібліотеки** (ID **файлу:** **1008368650**)

Пошук збігів з Інтернетом не проводився

14.5% Джерела з Бібліотеки | **275** ..... **Сторінка** 44

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | **1**

1 СУЧАСНА КОНЦЕПЦІЯ СТВОРЕННЯ БАЗ ДАНИХ

1.1 Актуальність проблеми розробки бази даних, основні концепції та визначення

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД[1].

Основним завданням бази даних є надійне збереження значних обсягів інформації, відомих як записи даних, і забезпечення доступу до неї для користувачів або прикладних програм. База даних складається з двох основних компонентів: збереженої інформації і системи управління цією інформацією. Для ефективного доступу до даних записи організовуються у вигляді множини фактів або елементів даних.

Дані – це інформація, відомості, показники, необхідні для ознайомлення з ким-, чим-небудь, для характеристики когось, чогось або для прийняття певних висновків, рішень. В базах даних важливу роль відіграє інформація. Інформація — абстрактне поняття, що має різні значення залежно від контексту[4].

Кожна створена база даних має свою предметну область. Предметна область – це необхідний для розробки бази даних об’єкт, який має в собі дані, які будуть зберігатися в базі даних.

Модель даних — абстрактне представлення реального світу, що відображає тільки ті об'єкти, що безпосередньо стосуються програми. Це, як правило, визначає специфічну групу об'єктів, їх атрибутивне значення і відношення між ними[1].

Існують два основні підходи до організації інформаційних масивів: файлова організація та організація у вигляді бази даних. Файлова організація передбачає спеціалізацію та збереження інформації, яка орієнтована, як правило, на одну прикладну задачу, і це здійснюється прикладним програмістом. Цей підхід дозволяє досягти високої швидкості обробки інформації, але має деякі недоліки.

Одним з них є вузька спеціалізація обробних програм і файлів даних, що призводить до надмірної дублювання даних, оскільки ті самі елементи даних зберігаються в різних системах. Крім того, збережена інформація контролюється різними особами, що ускладнює виявлення суперечливостей у даних. Файлова організація також не дозволяє ефективно задовольняти запити користувачів, що охоплюють декілька областей, через відмінності в структурі записів і форматах передачі даних[2]. Різниця між файловою огранізацією та базою даних зображена на Рисунку 1.1.

Рисунок 1.1 - Різниця між файловою організацією та базою даних

Тому виникає необхідність відокремити дані від їхнього опису і створити організацію збереження даних, яка враховуватиме існуючі зв'язки між ними і дозволить використовувати ці дані одночасно для різних застосувань. Вказані причини призвели до появи баз даних. База даних може бути описана як структурна сукупність даних, які знаходяться в активному стані і відображають властивості об'єктів зовнішнього (реального) світу. У базі даних зберігаються не тільки дані, але й їх описи, тому інформація про спосіб зберігання вже не прихована в поєднанні "файл-програма", а явно визначається в базі.

База даних спрямована на інтегровані запити, а не на одну програму, як у випадку файлового підходу, і використовується для задоволення інформаційних

потреб багатьох користувачів. Це дозволяє значно зменшити надлишковість інформації. Перехід від структури бази даних до потрібної структури в програмі користувача здійснюється автоматично за допомогою систем управління базами даних (СУБД).

Рисунок 1.2 - РСУБД

Системи управління базами даних (СУБД) - це програмні інструменти, які дозволяють створювати, заповнювати та опрацьовувати бази даних[3]. Застосування СУБД зображено на Рисунку 1.2. На сьогоднішній день існує велика кількість різноманітних СУБД, доступних у світі.

1.2 Формування та аналіз вимог до бази даних

Для початку розробки бази даних необхідно провести збір вимог. На цьому етапі розробники повинні провести опитування користувачів бази даних, щоб зрозуміти потреби системи та документувати дані та функціональні вимоги.

Установлення вимог передбачає обговорення та згоду між усіма користувачами щодо того, які дані потрібно постійно зберігати, а також узгодження значення та інтерпретації елементів даних.

Перелік вимог не повинен включати опис способу обробки даних, а містити лише інформацію про елементи даних, їх атрибути, обмеження та взаємозв'язки[2].

Аналіз даних розпочинається з врахування вимог до даних та створення концептуальної моделі даних. Головною метою аналізу є отримання найбільш повного опису даних, який відповідає вимогам користувача, з урахуванням властивостей даних на різних рівнях.

Концептуальна модель даних відображає загальне формальне представлення даних, яке використовується під час розробки баз даних та сприяє зрозумінню їх структури та значень незалежно від того, як будуть використовуватися дані користувачами або введення цих даних у конкретні IT-середовища. Основна мета концептуальної моделі полягає в описі значень та структури даних, а не в детальному описі їх реалізації [2].

Концептуальна модель даних згодом стає офіційним відображенням того, які дані має містити база даних та які обмеження повинні бути застосовані до цих даних. Модель повинна висловлюватись у термінах, що не залежать від способу, як модель може бути реалізована. Під час аналізу основний акцент робиться на запитаннях: "Що потрібно?" замість "Як це зробити?" [6].

1.3 Логічний дизайн

Початковим етапом розробки бази даних є створення концептуальної моделі даних та логічної схеми, що визначає необхідний тип СКБД (мережева, реляційна, об'єктно-орієнтована). Реляційна концептуальна модель даних може бути використана як вхідна інформація для логічного проектування, що дає можливість отримати детальну реляційну специфікацію та логічну схему таблиць та обмежень, що відповідають опису даних у концептуальній моделі даних. Результатом цього кроку є детальна реляційна специфікація, логічна схема всіх таблиць та обмежень, необхідних для відповідності опису даних у концептуальній моделі даних. На цьому етапі відбувається вибір найбільш наближених таблиць

для подання даних у БД, з урахуванням різних критеріїв проектування, таких як гнучкість для змін, контроль дублікатів та спосіб відображення обмежень [4]. Визначені таблиці логічної схеми визначають, які дані будуть зберігатися і як можна ними маніпулювати в базі даних.

Просте перетворення реляційного подання в таблиці SQL не гарантує, що створена база даних буде мати всі необхідні властивості, такі як повнота, цілісність, гнучкість, ефективність та простота використання. На першому етапі будуть визначені таблиці та обмеження, що відповідають концептуальній моделі даних, і задовольнять вимоги повноти та цілісності, але вони можуть бути жорсткими або не дуже корисними. Таким чином, перший дизайн може бути налаштований та оптимізований для покращення якості бази даних.

По-перше, не обов'язково мати єдину базу даних, щоб задовольнити вимоги користувачів даної концептуальної моделі даних. Існують різні причини для розробки декількох баз даних, такі як необхідність незалежної роботи в різних місцях або відомчий контроль над "своїми" даними. Однак, якщо колекція баз даних містить дублюючі дані, і користувачам потрібно отримати доступ до даних у декількох базах даних, то можливо одна база даних може задовольнити багато вимог, або можуть виникнути проблеми, пов'язані з реплікацією та розповсюдженням даних. Тому ці питання повинні бути ретельно розглянуті.

Другим аспектом проектування та реалізації баз даних є те, що багато різних аспектів залежать від конкретної СКБД, яку використовують. Якщо СКБД вже вибрана до початку проектування, то цей вибір може визначати критерії проектування замість того, щоб очікувати впровадження. Іншими словами, можна враховувати специфічні властивості обраної СКБД в процесі розробки дизайну, а не намагатися пристосувати загальний дизайн до конкретної СКБД [3].

Часто буває так, що один дизайн не може задовольнити всі вимоги до хорошої бази даних одночасно. Тому важливо, щоб проектувальник визначив пріоритетні властивості (зазвичай з використанням інформації з вимог), наприклад, вирішив, чи цілісність важливіша, ніж ефективність, або чи важливіша корисність, ніж гнучкість для даного проекту.

6

1.4 Реалізація

На етапі реалізації потрібно створити базу даних згідно з логічною схемою, включаючи збереження даних, захист інформації, зовнішню схему тощо. Успішна реалізація залежить від вибору СКБД, інструментів бази даних та операційної системи. Окрім створення схеми бази даних та встановлення обмежень, є додаткові завдання, такі як введення даних у таблиці, вирішення питань, що стосуються користувачів та їх процесів. Щоб зменшити кількість цих проблем, важливо вибрати СКБД, яка найкраще відповідає нашим потребам [3].

Після створення логічного дизайну потрібно реалізувати базу даних відповідно до визначень, що були створені. Для використання реляційної СКБД, необхідно використання SQL для створення таблиць та обмежень, які відповідають опису логічної схеми, та вибір відповідної схеми зберігання (якщо СКБД дозволяє такий рівень контролю).

Є кілька способів досягнення цієї мети. Один з них полягає в тому, щоб записати відповідні SQL-оператори DDL у файл, який може виконуватися СКБД, для того, щоб існував незалежний текстовий файл, який містить оператори SQL, що визначають базу даних. Інший метод полягає в інтерактивній роботі за допомогою інструменту бази даних, такого як SQL Server Management Studio або Microsoft Access [2] Приклад інтерактивної роботи зображено на Рисунку 1.3.

Рисунок 1.3 - Середовище інтерактивної роботи

Незалежно від механізму, який буде використаний для реалізації логічної схеми, результатом є база даних з таблицями та обмеженнями, яка визначена, але не містить даних для процесів користувача.

1.5 Заповнення таблиць бази даних

Є два способи заповнення таблиць після створення бази даних: використання наявних даних або введення нових даних за допомогою користувацьких програм, призначених для цієї бази даних. Деякі таблиці можуть містити дані з інших баз даних або файли даних. Наприклад, при створенні бази даних для лікарні можна очікувати, що база міститиме записи про персонал, які вже існують. Дані можуть бути отримані від зовнішнього агентства або створені під час введення великих обсягів даних. В таких випадках найбільш практичним є використання засобів імпорту та експорту, які містяться в СКБД.

Зазвичай у СКБД доступні засоби імпорту та експорту даних у різних стандартних форматах. Ці функції також можуть бути відомі як завантаження та розвантаження даних[3]. За допомогою імпорту можна копіювати файли даних безпосередньо в таблицю [6]. Але коли дані зберігаються у форматі файлу, який не можна імпортувати, необхідно підготувати прикладну програму, яка читатиме старі дані, перетворюватиме їх за необхідності та вставлятиме їх в базу даних за допомогою спеціально створеного коду SQL. Передача великої кількості існуючих даних до бази даних називається масовим завантаженням. Масове завантаження даних може містити дуже великі обсяги даних, що завантажуються по одній таблиці за раз. Тому можуть бути засоби СКБД, які можуть відкласти перевірку обмежень до кінця масового завантаження.

1.6 Нормалізація бази даних

Основні причини, з яких база даних може бути недосконалою:

- Надмірність. Дані в БД багаторазово повторюються.
- Потенційна суперечливість (аномалії оновлення). Внаслідок надмірності можна оновити адресу постачальника в одному рядку, залишаючи його незмінним в інших. Отже, при оновленнях необхідно переглядати всю таблицю для знаходження і зміни всіх відповідних рядків.
- Аномалії включення. В базу даних не може бути записаний новий постачальник, якщо продукт, що він постачає не використовується жодним споживачем.
- Аномалії видалення. Проблема виникає при необхідності видалення з бази даних інформації про всі продукти, окремого постачальника. При таких видаленнях можуть бути втрачені відомості про самого постачальника.

Для того щоби уникнути недосконалостей бази даних, які не дозволяють їй успішно функціонувати, виконують нормалізацію таблиць бази даних.

Нормалізація - це процес ефективної організації даних у базі даних. Теорія нормалізації ґрунтується на наявності тієї чи іншої залежності між полями таблиці [2].

Спільнота баз даних встановила нормальні форми як рекомендації для досягнення нормалізації баз даних. Нормалізація бази даних зображена на Рисунку 1.2.

Рисунок 1.2 - Нормалізація бази даних

Ці нормальні форми включають п'ять рівнів - від першої нормальної форми (1НФ), що є найнижчим рівнем, до п'ятої нормальної форми (5НФ). У практиці частіше застосовують 1НФ, 2НФ та 3НФ, а іноді 4НФ. 5НФ є досить рідкісною в практичному застосуванні.

Перша нормальна форма (1НФ) встановлює основні правила для структури бази даних, а саме:

- Стовпці повинні містити лише атомарні значення
- В таблиці не повинно бути груп даних, які повторюються

Друга нормальна форма (2НФ) повинна мати такі властивості:

- Таблиця знаходиться в 1НФ
- Таблиця не має часткових функціональних залежностей, це можна досягнути двома способами: всі стовпці таблиці є частиною первинного ключа, або таблиця має одностовпцевий первинний ключ.

Неповною функціональною залежністю називається залежність неключового атрибуту від частини ключа, що складається з декількох атрибутів. Повна функціональна залежність передбачає залежність неключового атрибуту від всіх атрибутів одночасно, що входять до складу ключа [6].

Тобто якщо таблиця, з синтетичним первинним ключем, не має складного первинного ключа, то вона завжди знаходиться в 2НФ [4].

Третя нормальна форма (3НФ) встановлює такі правила до таблиць БД:

- Таблиця знаходиться в 2НФ
- Таблиця не має транзитивних залежностей

Узагальнюючи: нормалізація баз даних використовується з метою покращення якості, ефективності зберігання та обробки даних. Вона дозволяє зменшити дублювання даних, зберігати дані у більш структурованому та логічному вигляді, забезпечувати більш точну та надійну інформацію, підвищувати швидкодію та знижувати обсяг пам'яті, що використовується для зберігання даних. Крім того, нормалізація баз даних допомагає зменшити можливість виникнення помилок при зміні даних та полегшує процес їхнього аналізу та звітування.

2 АНАЛІЗ ІСНУЮЧИХ СКБД

2.1 PostgreSQL

PostgreSQL - це об'єктно-реляційна система управління базами даних (СКБД), яка надає розширені можливості для зберігання та обробки даних. Вона була розроблена з відкритим вихідним кодом та доступна на багатьох платформах, таких як Linux, Windows та Mac OS X. PostgreSQL була розроблена на початку 1980-х років у місті Берклі, штат Каліфорнія, як проект по розробці СУБД Ingres. У 1996 році PostgreSQL була випущена як відкритий проект з використанням ліцензії PostgreSQL, що дозволяє вільне використання, модифікацію та розповсюдження [7].

Перша перевага PostgreSQL полягає в тому, що вона є однією з найбільш потужних та надійних СКБД, що доступні на ринку. Вона підтримує велику кількість стандартів SQL та має багато вбудованих функцій для роботи з геоданими, JSON-даними, XML-даними та іншими типами даних. Крім того, PostgreSQL надає можливість розширення за допомогою плагінів, що дозволяє користувачам розробляти свої власні розширення та додавати функціональність до бази даних.

Друга перевага PostgreSQL полягає в його відкритому вихідному коді, що робить його доступним для безкоштовного використання та модифікації. Це дає користувачам можливість адаптувати СКБД під свої потреби та створювати власні рішення на основі PostgreSQL. PostgreSQL дозволяє користувачеві налаштовувати систему шляхом визначення нових функцій, агрегатів, типів, мов, індексів та операторів. Крім того, відкритий код сприяє розвитку великої спільноти розробників, які активно працюють над покращенням та розширенням функціональності PostgreSQL.

Третя перевага PostgreSQL полягає в його підтримці транзакцій та відновлення після збоїв. Вона забезпечує можливість виконання транзакцій зі

стандартними ACID-властивостями. [7] ACID-властивості зображені на Рисунку 2.1.

Рисунок 2.1 - ACID-властивості

- Атомарність (Atomicity) - транзакція розглядається як єдиний логічний блок, всі її зміни або зберігаються повністю, або повністю скасовуються.
- Узгодженість (Consistency) - транзакція переводить базу даних з одного несуперечливого стану (на момент початку транзакції) в інший несуперечливий стан (на момент завершення транзакції). Несуперечливим вважається стан бази даних, коли виконуються всі обмеження фізичної та логічної цілісності бази даних, при цьому допускається порушення обмежень цілісності протягом транзакції, але на момент завершення всі обмеження цілісності, як фізичні, так і логічні, повинні бути дотримані.
- Ізольованість (Isolation) - зміни даних при конкурентних транзакціях ізольовані один від одного на основі системи версій.

- Стійкість (Durability) - PostgreSQL дбає про те, що результати успішних транзакцій гарантовано зберігаються на жорсткому диску незалежно від збоїв апаратної частини.

Крім того, PostgreSQL має вбудований механізм відновлення після збоїв, який дозволяє автоматично відновлювати базу даних до попереднього стану.

Четверта перевага полягає в її можливості масштабування та роботи з великими обсягами даних. Вона може працювати з базами даних розміром до терабайт та підтримує паралельну обробку запитів, що дозволяє прискорити роботу з великими обсягами даних.

П’ята перевага це безпека. Безпека даних також є найважливішим аспектом будь-якої СУБД. У PostgreSQL вона забезпечується 4 рівнями безпеки [7]:

- PostgreSQL не можна запустити під привілейованим користувачем - системний контекст.
- Шифрування трафіку між клієнтом та сервером за допомогою SSL, SSH - мережевий контекст.
- Складна система аутентифікації на рівні хоста або IP-адреси/підмережі. Система аутентифікації підтримує паролі, зашифровані паролі, Kerberos, IDENT та інші системи, які можуть підключатися за допомогою механізму підключення аутентифікаційних модулів [7].
- Деталізована система прав доступу до всіх об'єктів бази даних, яка разом із схемою забезпечує ізоляцію назв об'єктів для кожного користувача. Для цього PostgreSQL надає багату та гнучку інфраструктуру.

Нарешті, шоста перевага PostgreSQL полягає в її підтримці для різних мов програмування, таких як C, C++, Java, Python, Ruby та інші [7]. Це дає можливість розробникам використовувати PostgreSQL у своїх проектах та легко інтегрувати її з іншими системами.

Незважаючи на багато переваг, PostgreSQL також має деякі недоліки, зокрема:

- Вона може бути складною для новачків. Оскільки PostgreSQL є дуже гнучкою та розширюваною СУБД, вона може вимагати багато зусиль для освоєння та розуміння її функціоналу.
- Вона може бути менш продуктивною в порівнянні з іншими СУБД. PostgreSQL має багато функцій та можливостей, що можуть призвести до певного зниження продуктивності в порівнянні з менш функціональними СУБД.
- Потребує великої кількості пам'яті. PostgreSQL може вимагати значної кількості пам'яті для ефективної роботи з дуже великими базами даних або складними запитами.
- Обмежена підтримка кластеризації. PostgreSQL має обмежену підтримку кластеризації, що може зробити її менш підходящою для великих проектів з високими вимогами до доступності та масштабованості.

Враховуючи всі переваги та недоліки PostgreSQL, можна зробити висновок, що ця СКБД є однією з найкращих для зберігання та обробки даних. Вона має потужний функціонал, відкритий вихідний код, підтримку транзакцій та відновлення після збоїв, можливості масштабування та роботи з великими обсягами даних, а також підтримку для різних мов програмування. З цими перевагами PostgreSQL може задовольнити потреби різних користувачів, від невеликих стартапів до великих корпорацій.

2.2 SQLite

SQLite - це вбудовувана реляційна система управління базами даних (РСУБД), яка працює на різних платформах, включаючи Linux, Windows та MacOS. Вона відрізняється від інших РСУБД тим, що базується на одному файлі бази даних, який зберігається локально на пристрої [8]. Такий підхід до збереження даних дозволяє SQLite бути компактнішою та швидшою в порівнянні з іншими СУБД, а також не вимагає наявності окремого серверу баз даних.

Завдяки своїм властивостям, SQLite використовується:

- на сайтах з низьким і середнім трафіком;

- у локальних однокористувацьких, мобільних додатках або іграх, які не призначені для масштабування;
- в програмах, які часто виконують прямі операції читання/запису на диск;
- в додатках для тестування бізнес-логіки.

SQLite не потребує адміністрування та працює на мобільних пристроях, ігрових консолях, телевізорах, безпілотних літальних апаратах, камерах, мультимедійних системах автомобілів та інших пристроях. Ця СКБД використовується в багатьох програмах, таких як Firefox, Chrome, Safari, Skype, XnView, AIMP, Dropbox, Viber та багато інших.

SQLite має кілька переваг порівняно з іншими системами управління базами даних:

SQLite відзначається високою швидкістю роботи. Це досягається завдяки особливостям архітектури, яка дозволяє швидко здійснювати операції зчитування даних. Компоненти СКБД вбудовані безпосередньо в додаток і викликаються в межах того ж самого процесу, що дозволяє забезпечити швидший доступ до них порівняно з взаємодією між різними процесами.

Зберігання даних в одному файлі. База даних складається з таблиць, зв'язків між ними, індексів та інших компонентів. У SQLite вони зберігаються в одному файлі бази даних, який знаходиться на тому ж пристрої, що і програма. Щоб уникнути помилок під час роботи з файлом, він блокується для сторонніх процесів перед записом. Раніше це призводило до того, що дані в базу міг записувати тільки один процес одночасно. Але в нових версіях це проблему вирішується налаштуванням режиму роботи СКБД.

Мінімалізм. Творці SQLite використовують принцип "мінімального повного набору". З усіх можливостей SQL в ній є найбільш необхідні. Тому SQLite відрізняється малим розміром, простотою рішень та легкістю адміністрування. Для покращення базової функціональності можна використовувати сторонні програмні засоби та розширення.

Надійність. Код на 100% покритий тестами. Це означає, що перевірений кожен компонент ПО. Тому SQLite вважається надійною СУБД з мінімальним ризиком непередбачуваної поведінки.

Нульова конфігурація. Перед використанням СУБД не потрібна складна настройка або тривала установка. Для вирішення більшості завдань можна користуватися нею "з коробки", без установки додаткових компонентів.

Малий розмір. Повністю налаштований SQLite з усіма налаштуваннями займає менше 400 Кб. Якщо використовувати СУБД без додаткових компонентів, розмір можна зменшити до 250 Кб. Він залежить тільки від кількості завантаженої інформації. Незважаючи на малий розмір, SQLite підтримує більшість функцій стандарту SQL2 та має ряд власних [7].

Доступність. SQLite перебуває в загальнодоступному використанні. На її використання немає правових обмежень, а власником вважається громадськість. Можна відкривати, переглядати та змінювати вихідний код встановленого ПО.
Кросплатформність. СУБД підходить для UNIX-подібних систем, MacOS та Windows.

Автономність. Система незалежна від стороннього ПО, бібліотек або фреймворків. Щоб додаток з базою даних на SQLite працював, не потрібні додаткові компоненти. Крім того, не обов'язково мати доступ до Інтернету: вся база даних зберігається на пристрої, і дані можна отримати локально.

SQLite, як і будь-яке інше програмне забезпечення, має свої недоліки:

- Недостатня масштабовність. SQLite може працювати з невеликими базами даних, але не є оптимальним вибором для великих проектів, де потрібно обробляти значні обсяги даних.
- Менша продуктивність порівняно з іншими СУБД. SQLite може виконувати запити повільніше, ніж деякі конкуруючі СУБД, зокрема тих, що спеціалізуються на великих обсягах даних.
- Обмежена підтримка багатопоточності. SQLite підтримує багатопоточність, але не є найкращим вибором для проектів, де потрібно обробляти багато запитів одночасно.

- Недостатня безпека при одночасному доступі. SQLite не підтримує блокування рівня таблиці, тому може виникати проблема з безпечним доступом до бази даних, якщо декілька користувачів одночасно працюють з базою даних.
- Недостатня підтримка деяких типів даних. SQLite не підтримує деякі типи даних, які підтримуються іншими СУБД, такі як типи JSON або XML.

Загалом, SQLite - це потужна та надійна вбудовувана РСУБД, яка є ідеальним вибором для проектів з обмеженими ресурсами та для розробки мобільних додатків. Хоча вона має деякі обмеження, SQLite все ж залишається популярним вибором серед розробників завдяки своїм перевагам та широкому спектру застосувань.

2.3 Oracle Database

Oracle Database - це одна з найпопулярніших та найбільш потужних систем керування базами даних (СКБД), яка була розроблена корпорацією Oracle в 1977 році. Oracle Database є однією з найбільш складних та розвинених СКБД, яка має багато можливостей та функцій, що дозволяють їй працювати з великими обсягами даних та підтримувати високий рівень надійності та продуктивності.

Архітектура Oracle Database базується на клієнт-серверній моделі, де клієнти взаємодіють з сервером, щоб отримати доступ до даних. Сервер Oracle Database складається з трьох основних компонентів: сесійного процесу, процесу сервера та процесу фонового завдання. Сервер має вбудовану підтримку мови SQL, яка є стандартом для роботи з даними в СКБД [9].

Однією з найбільш важливих особливостей Oracle Database є її можливості використання в багатокористувацьких середовищах з великою кількістю користувачів та високими обсягами даних. Система підтримує розподілену архітектуру, що дозволяє розподіляти дані на кілька серверів та кластерів для забезпечення більшої надійності та масштабованості. Крім того, Oracle Database має вбудовані засоби для моніторингу та оптимізації продуктивності системи, такі

як діагностика проблем та планування операцій, що дозволяють забезпечувати стабільну та швидку роботу системи.

Oracle Database має багато можливостей та функцій, що дозволяють їй використовуватися в різних галузях та веб-проектах. Система підтримує багато типів даних, включаючи структуровані, напівструктуровані та некеровані дані, що дозволяє використовувати її в різних сценаріях [9].

Вона є однією з найбільш потужних та надійних систем керування базами даних (СКБД) з багатьма перевагами:

- Надійність та стійкість: Oracle Database є дуже стійкою та надійною системою, яка може обробляти великі обсяги даних без втрати продуктивності та надійності.
- Масштабованість: Система може легко масштабуватися в залежності від зростання потреб користувачів та даних. Oracle Database підтримує розподілені бази даних, кластеризацію та інші технології, які дозволяють розширювати масштаб системи.
- Швидкість та продуктивність: Oracle Database може обробляти великі обсяги даних з високою швидкістю, що дозволяє користувачам отримувати доступ до даних швидко та ефективно.
- Розширені можливості: Oracle Database має багато функцій та можливостей, що дозволяють користувачам працювати з даними різних типів та форматів. Система має вбудовану підтримку мови SQL, а також інші інструменти та сервіси для роботи з базами даних.
- Захист даних: Oracle Database забезпечує високий рівень захисту даних та конфіденційності користувачів. Система має різні механізми захисту, такі як шифрування даних та автентифікацію користувачів [9].

Якщо виділити тільки головні переваги Oracle Database то це - її висока надійність та стабільність. Система має вбудований механізм відновлення після збоїв та аварій, що дозволяє забезпечувати високий рівень доступності даних та продуктивності системи. Крім того, Oracle Database має вбудовану систему

захисту даних, що дозволяє захищати конфіденційні дані від несанкціонованого доступу та зламів.

Незважаючи на те, що Oracle Database є однією з найпопулярніших та найбільш потужних систем керування базами даних, вона також має кілька недоліків.

Висока вартість: Oracle Database є однією з найбільш дорогих СКБД на ринку. Це може бути значним обмеженням для менших компаній та стартапів, які не можуть собі дозволити витратити значну суму на встановлення та підтримку системи.

Складність: Oracle Database є дуже складною системою, що вимагає від операторів високого рівня знань та досвіду. Для налагодження та належної експлуатації системи необхідно мати спеціаліста зі знанням архітектури, SQL та інших технологій, що використовуються в Oracle Database.

Великі вимоги до апаратного забезпечення: Oracle Database вимагає відносно потужного апаратного забезпечення та пам'яті для ефективної роботи з великими обсягами даних. Це може бути додатковим фактором, що підвищує вартість встановлення та підтримки системи.

Закритість: Oracle Database є закритою системою, що використовує пропрієтарні технології. Це може бути обмеженням для компаній, які працюють з відкритим кодом та віддають перевагу вільному програмному забезпеченню.

Проблеми з безпекою: Oracle Database може бути під ударом хакерів та зловмисників, що може привести до витоку конфіденційної інформації та порушення безпеки даних. Однак, Oracle постійно вдосконалює свої методи захисту та патчі для запобігання таким проблемам [9].

Загалом, Oracle Database є однією з найбільш потужних та надійних систем керування базами даних, яка має багато можливостей та функцій для роботи з великими обсягами даних та високими навантаженнями. Система є популярною серед великих корпорацій та організацій, які потребують надійного та швидкого доступу до великих обсягів даних.

2.4 MySQL

MySQL - це одна з найпопулярніших систем керування базами даних (СКБД), що була розроблена в 1995 році компанією MySQL AB та придбана компанією Oracle Corporation у 2010 році [5]. MySQL використовується в багатьох веб-проектах та додатках, що підтримуються інтернет-компаніями та організаціями.

Рисунок 2.2 - Архітектура MySQL

Архітектура MySQL базується на клієнт-серверній моделі, де клієнти взаємодіють з сервером, щоб отримати доступ до даних. Архітектура MySQL зображена на Рисунку 2.2

Сервер MySQL включає в себе ряд компонентів, таких як MySQL сервер, мережевий протокол, інтерпретатор мови SQL, бібліотеку клієнта та інші. Клієнти можуть звертатися до сервера через різні інтерфейси, такі як командний рядок, графічний інтерфейс користувача та веб-сервер [10].

MySQL підтримує багато можливостей, що роблять її популярним у світі веб-розробки. Сервер MySQL підтримує транзакції з використанням механізму блокування рівня рядка, що забезпечує високу рівень ізоляції та конкурентної обробки запитів. Сервер також має вбудовану підтримку кластеризації, що дозволяє об'єднувати декілька серверів MySQL в один, забезпечуючи високий рівень доступності та масштабованості [10].

Одна з головних переваг MySQL - це відкритість та безкоштовність, що робить її доступною для використання для будь-якої організації або користувача. MySQL також має велику спільноту користувачів, яка забезпечує підтримку та відповіді на запитання щодо використання системи. MySQL має подвійне ліцензування. Вона може розповсюджуватися відповідно до умов ліцензії GPL. Але за умовами GPL, якщо якась програма використовує бібліотеки MySQL, то вона теж повинна розповсюджуватися за ліцензією GPL. Проте це може розходитися з планами розробників, не бажаючих відкривати текстів своїх програм. Для таких випадків передбачена комерційна ліцензія компанії MySQL AB, яка також забезпечує якісну сервісну підтримку. В разі використання та розповсюдження програмного забезпечення з іншими вільними ліцензіями, такими як BSD, MIT та інші, MySQL дозволяє використання бібліотек MySQL за ліцензією GPL [10].

Крім того, MySQL є дуже ефективною системою, що може оброблювати великі обсяги даних з високою швидкістю. Система має вбудований кешувальний механізм, що дозволяє збільшувати швидкість обробки запитів за рахунок збереження часто використовуваних даних в оперативній пам'яті.

Серед основних переваг MySQL відзначають наступні:

- Масштабованість. MySQL може підтримувати роботу БД значних розмірів, що підтверджують її реалізації в Yahoo!, Google,HP, Associated Press. Згідно документації, що додається до MySQL, деякі БД, що використовуються компанією MySQL AB (розробником MySQL), зберігають до 50 млн. записів [10].

- Переносимість. MySQL працює на різних платформах, серед яких Unix, Linux, Windows, OS/2, Solaris, Mac OC. Окрім того, MySQL працює на різних платформах.
- До MySQL можна одержувати доступ із будь-якої точки Internet кільком користувачам одночасно. MySQL має цілий ряд програмних інтерфейсів додатків (Application Programming Interface –API), які дозволяють встановлювати з'єднання з MySQL із додатків, написаних на таких мовах як C, C++, Perl, PHP, Java, Python [5].
- Безпека. MySQL має систему контролю доступу до даних, забезпечує шифрування даних при передаванні.
- Швидкість функціонування.
- Зручність експлуатації. MySQL досить зручно встановлюється та реалізується, легко адмініструється.
- Відкритий код.

Однак, наявні й певні недоліки MySQL, зокрема, обмежена підтримка високорівневих операцій та складних запитів, порівняно з іншими СКБД, такими як Oracle або PostgreSQL. Також MySQL не має вбудованих засобів реплікації даних, що може стати проблемою при розвитку великих проектів.

У підсумку, MySQL є однією з найпопулярніших СКБД, яка забезпечує ефективну обробку даних та високу швидкість роботи. Вона має велику спільноту користувачів та безкоштовну ліцензію, що робить її доступною для використання для будь-якої організації або користувача. Також MySQL має підтримку транзакцій та кластеризації, що забезпечує високий рівень доступності та масштабованості. Однак, необхідно враховувати недоліки системи, такі як обмежена підтримка високорівневих операцій та складних запитів, а також відсутність вбудованих засобів реплікації даних.

3 РЕЛЯЦІЙНА МОДЕЛЬ БД РІТЕЙЛ КОМПАНІЇ

3.1 Опис предметної області

Предметна область рітейл компанії включає в себе широкий спектр діяльностей, пов'язаних з роздрібною торгівлею товарами або послугами. Основними елементами предметної області рітейл компанії є:

Товари: В базі даних рітейл компанії необхідно зберігати інформацію про товари, які продаються. Це можуть бути назви товарів, описи, атрибути, ціни, категорії, постачальники тощо. Інформація про товари дозволить відстежувати наявність товару на складі, контролювати запаси і забезпечувати належне управління асортиментом.

Клієнти: База даних повинна містити інформацію про клієнтів, які здійснюють покупки. Це можуть бути персональні дані клієнтів, історія покупок, преференції, контактна інформація. Інформація про клієнтів допоможе розробляти персоналізовані пропозиції, здійснювати маркетингові активності та вдосконалювати обслуговування клієнтів.

Продажі: База даних повинна відображати інформацію про здійснені продажі, такі як дата, час, товари, кількість, ціна, знижки тощо. Інформація про продажі дозволить аналізувати обсяги продажів, визначати популярні товари, ефективність маркетингових акцій та забезпечувати фінансовий облік.

Складське управління: База даних повинна включати інформацію про склади, запаси товарів на складі, розміщення товарів на складі.

Постачальники: База даних повинна містити інформацію про постачальників, в яких компанія здійснює закупівлі товарів. Це можуть бути контактні дані постачальників, умови поставок, історія співпраці, ціни, якість продукції. Інформація про постачальників допоможе керувати ланцюгом поставок, встановлювати оптимальні умови закупівель та контролювати якість продукції.

Управління акціями та знижками: База даних може містити інформацію про акції, знижки та програми лояльності для клієнтів. Це можуть бути дані про

акційні пропозиції, умови участі, знижки. Інформація про акції та знижки допоможе привертати та утримувати клієнтів, підвищувати лояльність та збільшувати обсяги продажів.

3.2 Встановлення сутностей та їх властивостей

На даному етапі були виділені необхідні сутності та визначені їх властивості. Код створення таблиць на основі цих сутностей наведено в додатку А. На підставі опису предметної області та встановлених вимог, були виділені такі сутності:

Замовлення у постачальника:

- Статус замовлення
- Працівник, який створив замовлення
- Постачальник товару
- Дата створення замовлення
- Очікувана дата отримання готового товару
- Товари, які замовляються у виробника
- Кількість товарів
- Вартість замовлення без урахування податку та доставки
- Сума податку
- Вартість доставки
- Загальна вартість замовлення
- Фактична кількість отриманого товару
- Забракована кількість під час перевірки
- Кількість товару прийнята до обліку

Працівник:

- ПІБ
- Дата народження
- Стать
- Сімейний стан

- Посада
- Дата прийому на роботу
- Кількість годин, доступних для відпустки
- Кількість годин, доступних для лікарняного
- Зміна, в якій даний працівник працює
- Відділ
- Зарплата

Постачальник:

- Назва компанії
- Кредитний рейтинг
- Товари, які він може постачати та їхня вартість
- Мінімальна та максимальна кількість товару, яка може бути замовлена у даного постачальника

Товар:

- Назва
- Категорія
- Рівень резервного запасу
- Наявність на складі
- Кількість товару на складі при якій необхідного замовити новий товар
- Стандартна ціна товару
- Реальна ціна товару для продажу
- Дата початку продажу товару

Замовлення на власне виробництво:

- Товар
- Кількість
- Фактична кількість, яка виготовлена та поставлена на склади зберігання
- Кількість, яка не пройшла перевірку якості
- Дата розміщення замовлення
- Термін виконання замовлення

- Орієнтовна вартість виробництва товару

Продаж:

- Дата створення замовлення клієнта
- Термін виконання замовлення
- Статус
- Вид продажу
- Клієнт, який замовив даний товар
- Адреса доставки
- Адреса виставлення рахунку
- Товари
- Кількість
- Спеціальні пропозиції які діють для певних товарів
- Вартість замовлення без урахування податку та доставки
- Сума податку
- Вартість доставки
- Загальна вартість замовлення
- Причина продажу

Клієнт:

- ПІБ
- Контактна інформація

Магазин:

- Назва
- Торгівельний представник, який є відповідальним за даний магазин

3.3 Побудова ER-діаграми

Виходячи з опису предметної області, та враховуючи певні додаткові властивості, які були необхідні для сутностей, створено ER-діаграму, яка задовольняє вимоги до бази даних. У результаті отримано реляційну схему бази даних, зображену на рисунку 3.1.

Рисунок 3.1 - ERR-діаграма бази даних рітейл компанії

3.4 Програмування переглядів (Views)

Для бази даних створено декілька переглядів (Views). Вони необхідні для оптимізації бази даних та забезпечення безпеки даних. Оскільки це віртуальні таблиці, вони не зберігають фактичних даних, а отримують їх з інших таблиць за допомогою запитів SQL. Також, перегляди можуть містити тільки обмежену кількість полів з таблиці, щоб уникнути витоку конфіденційної інформації. Код створення переглядів наведено у додатку Б.

Перегляд «product_list»:

- Name. Назва товару.
- Color. Колір товару.
- Line. Назва лінійки товарів, до якої належить даний товар.
- Class. Класс, до якого належить даний товар.
- Style. Стиль даного товару.
- Category. Категорія до якої належить даний товар.
- Price. Ціна товару.
- Vendors. Список виробників даного товару.

Перегляд «content_of_storages»:

- Name. Назва складу.
- Shelf. Стелаж на якому зберігаються товари.
- Products. Список товарів, які зберігаються на певному стелажі.

Перегляд «sales_by_reason»:

- SalesReason. Назва причини.
- TotalSales. Загальна вартість усіх продажів товарів.

Перегляд «sales_by_category»:

- Name. Назва категорії.
- TotalSales. Загальна вартість усіх продажів товарів даної категорії.

Перегляд «sales_to_stores»:

- Buyer. Назва магазину, який здійснив покупку.
- Sold by. Ім’я та прізвище продавця.
- Ordered items. Список товарів, які замовив магазин.
- Total. Вартість замовлених товарів з урахуванням податку та доставки.
- Date of order. Дата здійснення продажу.

Перегляд «staff_list»:

- Name. Прізвище та ім’я працівника.
- Age. Вік.
- Gender. Стать.
- BirthDate. Дата народження.

- Marital Status. Сімейний статус.
- Phone. Телефонний номер.
- Email. Електронна адреса працівника.
- Job. Посада, яку займає даний працівник.
- Salary. Зарплата працівника.
- Department. Назва відділу та групи до якої належить працівник.
- Shift. Зміна в якій він працює.
- Shift Starts At. Час початку зміни.
- Shift Ends At. Час кінця зміни.
- VacationHours. Кількість годин доступних для відпустки.
- SickLeaveHours. Кількість годин доступних для лікарняної відпустки.
- Currently working. Вказує на те, чи працює даний працівник, чи перебуває у відпустці.
- Hired in. Дата найму на роботу.
- Experience. Стаж роботи працівника в нашій компанії.

3.5 Програмування тригерів та процедур бази даних.

Тригери можуть бути корисні для забезпечення цілісності даних, контролю доступу до даних та автоматизації рутинних завдань. Для бази даних було створено декілька тригерів, які виконуються автоматично відповідно до певних подій або змін в таблицях бази даних.

Для таблиці «sales_orderdetails»:

```
CREATE DEFINER=`root`@`localhost` TRIGGER `LineTotalCalculationBeforeInsert`
BEFORE INSERT ON `sales_orderdetails` FOR EACH ROW BEGIN
    DECLARE offerID INT DEFAULT 0;
    DECLARE discount INT DEFAULT 0;
    DECLARE subTotal INT DEFAULT 0;

    SELECT SpecialOfferID, MAX(DiscountPct) INTO offerID , discount
```

```
        FROM
            special_offer AS so RIGHT JOIN specialoffer_product sp
            ON so.SpecialOfferID = sp.SpecialOffer
        WHERE
            so.EndDate > CURRENT_TIMESTAMP
                AND sp.ProductID = NEW.ProductID
                AND NEW.OrderQuantity < so.MaxQuantity
                AND NEW.OrderQuantity > so.MinQuantity
        GROUP BY SpecialOfferID;

        IF offerID IS NULL THEN
            SET NEW.SpecialOfferID = 1;
            SET NEW.UnitPriceDiscount = 0.0000;
        ELSE
            SET NEW.SpecialOfferID = offerID;
            SET NEW.UnitPriceDiscount = discount;
            SET NEW.LineTotal = NEW.UnitPrice * (1 - NEW.UnitPriceDiscount) *
            NEW.OrderQuantity;
        END IF;
END

CREATE DEFINER=`root`@`localhost` TRIGGER `InventoryAndSalesOrderUpdate`
AFTER INSERT ON `sales_orderdetails` FOR EACH ROW BEGIN
        DECLARE subTotal INT DEFAULT 0;
        DECLARE taxAmt INT DEFAULT 0;
        DECLARE freight INT DEFAULT 0;

        SELECT SUM(LineTotal) INTO subTotal FROM sales_orderdetails  WHERE
        SalesOrderID = NEW.SalesOrderID GROUP BY SalesOrderID;

        SET taxAmt = subTotal * 0.2;
```

```
        SET freight = subTotal * 0.1;

        UPDATE sales_order
        SET SubTotal = subTotal, TaxAmt = taxAmt, Freight = freight, TotalDue =
        subTotal + taxAmt + freight WHERE  SalesOrderID = NEW.SalesOrderID;

        UPDATE inventory SET Quantity = Quantity - NEW.OrderQuantity WHERE
        ProductID = NEW.ProductID AND Quantity IN (SELECT MAX(Quantity)
        FROM inventory WHERE ProductID = NEW.ProductID);
END
```

Для таблиці «purchasing_order»:

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`PurchasingOrderDetailsLineTotalCalculationBeforeInsert` BEFORE INSERT ON
`purchasing_orderdetails` FOR EACH ROW BEGIN
        SET NEW.LineTotal = NEW.UnitPrice * NEW.OrderQuantity;
END


CREATE DEFINER=`root`@`localhost` TRIGGER
`PurchasingOrderTotalCalculationAfterInsert` AFTER INSERT ON
`purchasing_orderdetails` FOR EACH ROW BEGIN
        DECLARE subTotal INT DEFAULT 0;

        SELECT SUM(LineTotal) INTO subTotal FROM  purchasing_orderdetails
        WHERE PurchaseOrderID = NEW.PurchaseOrderID GROUP BY
        PurchaseOrderID;

        UPDATE purchasing_order
        SET SubTotal = subTotal, TotalDue = subTotal + TaxAmt + Freight    WHERE
        PurchaseOrderID = NEW.PurchaseOrderID;
END
```

Для таблиці «inventory»:

```
CREATE DEFINER=`root`@`localhost` TRIGGER `CheckReorderPoint` AFTER
UPDATE ON `inventory` FOR EACH ROW BEGIN
    DECLARE isProduced TINYINT(1) DEFAULT 0;
    DECLARE reorderPoint INT DEFAULT 0;
    DECLARE productQuantity INT DEFAULT 0;
    DECLARE safetyStockLevel INT DEFAULT 0;

    SELECT MakeFlag, ReorderPoint, SafetyStockLevel INTO isProduced,
    reorderPoint, safetyStockLevel FROM product WHERE product.ProductID =
    NEW.ProductID;

    SELECT SUM(Quantity) INTO productQuantity FROM inventory WHERE
    ProductID = NEW.ProductID GROUP BY ProductID;

    IF productQuantity <= reorderPoint THEN
        IF MakeFlag = 1 THEN
            CALL CreateProductionOrder(NEW.ProductID, safetyStockLevel -
            productQuantity);
        ELSE
            CALL CreatePurchasingOrder(NEW.ProductID, safetyStockLevel -
            productQuantity);
        END IF;
    END IF;
END
```

Для даної бази даних також створено декілька процедур, які полегшать роботу з нею.

Процедура «CreateProductionOrder» необхідна для створення нового замовлення на виробництво:

```
DELIMITER $$
USE `retaildb`$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`CreateProductionOrder`(productID INT, quantity INT)
BEGIN
    DECLARE daysToManufacture INT DEFAULT 0;

    SELECT DaysToManufacture INTO daysToManufacture FROM product
    WHERE product.ProductID = productID;

    INSERT INTO production_order (ProductID, OrderQuantity, DueDate)
    VALUES (productID, quantity, ADDDATE(CURRENT_TIMESTAMP,
    INTERVAL daysToManufacture DAY));

END$$
DELIMITER ;
```

Процедура «CreatePurchasingOrder» необхідна для розміщення нового замовлення у постачальника.

```
DELIMITER $$
USE `retaildb`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`CreatePurchasingOrder`(productID INT, quantity INT, userID INT)
BEGIN
    DECLARE vendor INT DEFAULT 0;
    DECLARE price INT DEFAULT 0;
    DECLARE leadTime INT DEFAULT 0;
    DECLARE orderID INT DEFAULT 0;

    SELECT MIN(pv.StandardPrice), pv.VendorID, AverageLeadTime INTO price,
    vendor, leadTime FROM product_vendor AS pv INNER JOIN vendor AS v
    ON pv.VendorID = v.EntityID  WHERE PreferredVendorStatus = 1 AND
    ActiveFlag = 1 GROUP BY pv.ProductID HAVING pv.ProductID = productID;
```

```
SELECT PurchaseOrderID + 1 INTO orderID FROM purchasing_order
ORDER BY PurchaseOrderID DESC LIMIT 1;

INSERT INTO purchasing_order (PurchaseOrderID, EmployeeID, VendorID,
ShipDate, TaxAmt, Freight) VALUES (orderID, userID, vendor,
ADDDATE(CURRENT_TIMESTAMP, INTERVAL leadTime DAY), price * quantity *
0.2, price * quantity * 0.1);

INSERT INTO purchasing_orderdetails (PurchaseOrderID, ProductID, DueDate,
OrderQuantity, UnitPrice) VALUES (orderID, productID,
ADDDATE(CURRENT_TIMESTAMP, INTERVAL leadTime DAY), quantity, price);

END$$
DELIMITER ;
```

3.6 Програмування запитів до бази даних

Основна мета створення бази даних полягає в забезпеченні швидкого доступу до необхідної інформації. Для того, щоб отримати необхідну інформацію у потрібному форматі, необхідно скористатися набором стандартних запитів.

Для бази даних було створено такі запити:

1. Кількість продажів, які здійснив кожний окремий продавець.

```
SELECT
  CONCAT_WS('. ',
      SUBSTRING(p.FirstName, 1, 1),
      p.LastName) AS `Name`,
  COUNT(so.SalesOrderID) AS `Number of sales`
FROM
  sales_order AS so
    RIGHT JOIN
  sales_person AS sp ON so.SalesPersonID = sp.EntityID
```

```
    INNER JOIN
  person AS p ON sp.EntityID = p.EntityID
GROUP BY so.SalesPersonID
ORDER BY `Number of sales` DESC;
```

2. Кількість продажів, здійснених в кожній окремій країні.

```
SELECT
  st.Country AS `Country`,
  COUNT(so.SalesOrderID) AS `Number of sales`
FROM
  sales_order AS so
    RIGHT JOIN
  sales_territory AS st ON so.TerritoryID = st.TerritoryID
GROUP BY st.Country
ORDER BY `Number of sales` DESC;
```

3. Список найгірших постачальників, в яких була найбільша кількість бракованих товарів (відсоток браку ≥ 5%).

```
SELECT
  v.Name AS `Vendor`,
  ROUND(SUM(pod.RejectedQuantity) / SUM(pod.OrderQuantity) * 100,
      2) AS `Rejected rate`
FROM
  purchasing_orderdetails AS pod
    INNER JOIN
  purchasing_order AS po ON pod.PurchaseOrderID = po.PurchaseOrderID
    INNER JOIN
  vendor AS v ON po.VendorID = v.EntityID
GROUP BY po.VendorID
HAVING `Rejected rate` >= 5
ORDER BY `Rejected rate` DESC;
```

4. Список товарів, які найбільш часто продаються.

```
SELECT
   p.Name AS `Product`,
   COUNT(sod.SalesOrderDetailID) AS `Number of sales`,
   SUM(sod.OrderQuantity) AS `Sold quantity`
FROM
   sales_orderdetails AS sod
      INNER JOIN
   product AS p ON sod.ProductID = p.ProductID
GROUP BY sod.ProductID
ORDER BY `Number of sales` DESC;
```

5. Наявність товарів та їхня кількість на складах.

```
SELECT
   p.Name AS `Product`,
   IFNULL(SUM(i.Quantity), 0) AS `Quantity`,
   GROUP_CONCAT(l.Name
      SEPARATOR ', ') AS `Stored at`
FROM
   product AS p
      LEFT JOIN
   inventory AS i ON p.ProductID = i.ProductID
      LEFT JOIN
   location AS l ON i.LocationID = l.LocationID
WHERE
   p.SellEndDate IS NULL
GROUP BY p.ProductID;
```

6. Кількість працівників, які працюють в кожній зміні.

```
SELECT
   s.Name AS `Shift`,
   COUNT(edh.EntityID) AS `Number of workers`
FROM
```

```
    employee_department_history AS edh
        INNER JOIN
    shift AS s ON edh.ShiftID = s.ShiftID
WHERE
    edh.EndDate IS NULL
GROUP BY s.ShiftID
ORDER BY `Number of workers` DESC;
```

7. Кількість працівників, які працюють в кожному відділі.

```
SELECT
    d.Name AS `Deparment`,
    COUNT(edh.EntityID) AS `Number of workers`
FROM
    employee_department_history AS edh
        INNER JOIN
    department AS d ON edh.DepartmentID = d.DepartmentID
WHERE
    edh.EndDate IS NULL
GROUP BY d.DepartmentID;
```

8. Кількість окремих товарів, які були замовлені онлайн.

```
SELECT
    p.Name AS `Product`,
    COUNT(sod.SalesOrderDetailID) AS `Number of sales`,
    SUM(sod.OrderQuantity) AS `Sold quantity`
FROM
    sales_orderdetails AS sod
        INNER JOIN
    sales_order AS so ON so.SalesOrderID = sod.SalesOrderID
        INNER JOIN
    product AS p ON sod.ProductID = p.ProductID
WHERE
```

```
    so.OnlineOrderFlag = 1
GROUP BY sod.ProductID
ORDER BY `Number of sales` DESC;
```

9. Загальна кількість кожного виготовленого товару та кількість його браку.

```
SELECT
    p.Name AS `Product`,
    SUM(po.StockedQuantity) AS `Produced quantity`,
    SUM(po.ScrappedQuantity) AS `Scrapped quantity`,
    ROUND(SUM(po.ScrappedQuantity) / SUM(po.StockedQuantity) * 100,
            2) AS `Scrapped rate`
FROM
    production_order AS po
        INNER JOIN
    product AS p ON po.ProductID = p.ProductID
WHERE
    p.DiscontinuedDate IS NULL
GROUP BY po.ProductID;
```

10. Загальна вартість усіх продажів, витрат на закупівлю у постачальника, вартість власного виробництва та вартість усіх виплат персоналу.

```
SELECT
        (SELECT
        SUM(TotalDue)
    FROM
        sales_order
    WHERE
        Status = 5) AS `Total Sales`,
    (SELECT
        SUM(TotalDue)
    FROM
        purchasing_order
```

```
  WHERE
    Status = 4) AS `Total purchasing`,
  (SELECT
    SUM(ActualCost)
  FROM
    production_orderdetails) AS `Total production`,
  (SELECT
    SUM(Salary)
  FROM
    salaries AS s
  INNER JOIN employee AS e ON s.EntityID = e.EntityID
  WHERE
    e.ActiveFlag = 1
      AND s.SalaryChangeDate IN (SELECT
        MAX(SalaryChangeDate)
      FROM
        salaries
      WHERE
        EntityID = e.EntityID)) AS `Total salaries`;
```

11. Загальна сума продажів по рокам.

```
SELECT
  SUM(TotalDue) AS `Total sales`, YEAR(OrderDate) AS `Year`
FROM
  sales_order
WHERE
  Status = 5
GROUP BY `Year`;
```

12. Загальна кількість клієнтів в кожній окремій країні.

```
SELECT
  st.Country AS `Country`,
```

```
   COUNT(c.CustomerID) AS `Number of customers`
FROM
   customer AS c
      INNER JOIN
   sales_territory AS st ON c.TerritoryID = st.TerritoryID
GROUP BY st.Country
ORDER BY `Number of customers` DESC;
```

13. Контактна інформація про магазини, яким ми продаємо наші товари.

```
SELECT
   s.Name AS `Store`,
   CONCAT_WS('. ',
         SUBSTRING(p.FirstName, 1, 1),
         p.LastName) AS `Contact Person `,
   p.PhoneNumber AS `Phone`,
   p.EmailAddress AS `Email`
FROM
   store AS s
      INNER JOIN
   person AS p ON s.EntityID = p.EntityID;
```

14. Кількість працівників пенсійного віку.

```
SELECT
   COUNT(EntityID) `Number of employee`
FROM
   employee
WHERE
   TIMESTAMPDIFF(YEAR, BirthDate, NOW()) > 60;
```

15. Кількість товарів по категоріям.

```
SELECT
   c.Name AS `Category`,
   COUNT(p.ProductID) AS `Number of products`
```

```
FROM
    product AS p
        RIGHT JOIN
    category AS c ON p.CategoryID = c.CategoryID
WHERE
    p.SellEndDate IS NULL
GROUP BY c.CategoryID
ORDER BY `Number of products` DESC;
```

Схожість

Джерела з Бібліотеки 275

1	Студентська робота	ID файлу: 1008368650	Навчальний заклад: Lviv Polytechnic National University	63 Джерело	3.06%
2	Студентська робота	ID файлу: 8457039	Навчальний заклад: National University Ostroh Academy	100 Джерело	2.38%
3	Студентська робота	ID файлу: 1012692049	Навчальний заклад: National Aviation University	4 Джерело	2.29%
4	Студентська робота	ID файлу: 1008187861	Навчальний заклад: National Aviation University	3 Джерело	2.28%
5	Студентська робота	ID файлу: 1005782008	Навчальний заклад: National Aviation University		2.13%
6	Студентська робота	ID файлу: 1003528287	Навчальний заклад: Cherkasy State Technological Univers	5 Джерело	2%
7	Студентська робота	ID файлу: 1008404145	Навчальний заклад: National University of Water Manag	11 Джерело	1.93%
8	Студентська робота	ID файлу: 3375077	Навчальний заклад: Lviv Polytechnic National University		1.76%
9	Студентська робота	ID файлу: 1008296232	Навчальний заклад: Lviv Polytechnic National University		1.42%
10	Студентська робота	ID файлу: 1008272038	Навчальний заклад: National University of Life and Environmenta...		1.32%
11	Студентська робота	ID файлу: 2048611	Навчальний заклад: National University of Life and Environmental Sc...		1.2%
12	Студентська робота	ID файлу: 952250	Навчальний заклад: Lviv Polytechnic National University	2 Джерело	1.18%
13	Студентська робота	ID файлу: 1008376259	Навчальний заклад: State University Kyiv National Economic Univ...		0.97%
14	Студентська робота	ID файлу: 6038551	Навчальний заклад: Lviv Polytechnic National University	12 Джерело	0.75%
15	Студентська робота	ID файлу: 1011348886	Навчальний заклад: National Aviation University	2 Джерело	0.73%
16	Студентська робота	ID файлу: 1008311664	Навчальний заклад: National Aviation University	2 Джерело	0.65%
17	Студентська робота	ID файлу: 5961574	Навчальний заклад: National Technical University of Ukraine “Kyiv Po...		0.46%
18	Студентська робота	ID файлу: 1008305845	Навчальний заклад: Lutsk National Technical University	2 Джерело	0.29%
19	Студентська робота	ID файлу: 1000089689	Навчальний заклад: National Technical University of Ukraine “Kyi...		0.23%
20	Студентська робота	ID файлу: 1000090474	Навчальний заклад: National Technical University of Ukr	24 Джерело	0.23%

21	Студентська робота	ID файлу: 1015083094	Навчальний заклад: National Technical University of Ukraine “Kyi...		0.23%
22	Студентська робота	ID файлу: 1015210109	Навчальний заклад: National Technical University of Ukraine “Kyi...		0.16%
23	Студентська робота	ID файлу: 104779	Навчальний заклад: Lviv Polytechnic National University	2 Джерело	0.16%
24	Студентська робота	ID файлу: 1015279046	Навчальний заклад: National Technical University of Ukra	2 Джерело	0.15%
25	Студентська робота	ID файлу: 1013089717	Навчальний заклад: Yuriy Fedkovych Chernivtsi National	2 Джерело	0.15%
26	Студентська робота	ID файлу: 1015081164	Навчальний заклад: National University of Life and Envir	2 Джерело	0.15%
27	Студентська робота	ID файлу: 1014862359	Навчальний заклад: State University Kyiv National Economic Univ...		0.13%
28	Студентська робота	ID файлу: 1015276574	Навчальний заклад: National Aviation University		0.13%
29	Студентська робота	ID файлу: 1013076375	Навчальний заклад: Lviv Polytechnic National University	17 Джерело	0.12%
30	Студентська робота	ID файлу: 1004101991	Навчальний заклад: Lviv Polytechnic National University		0.12%
31	Студентська робота	ID файлу: 1014520146	Навчальний заклад: Taras Shevchenko National Universit	2 Джерело	0.12%
32	Студентська робота	ID файлу: 1015283129	Навчальний заклад: Taras Shevchenko National University of Kyiv		0.12%
33	Студентська робота	ID файлу: 1008366983	Навчальний заклад: National Technical University of Ukraine “Kyi...		0.12%
34	Студентська робота	ID файлу: 1011568343	Навчальний заклад: Vasyl Stus Donetsk National University		0.12%
35	Студентська робота	ID файлу: 1013251953	Навчальний заклад: National Technical University of Ukraine “Kyi...		0.12%
36	Студентська робота	ID файлу: 1015278907	Навчальний заклад: National Aviation University		0.12%