**ЗМІСТ**

[ХАРАКТЕРИСТИКА БАЗИ ПРАКТИКИ 2](#_Toc11799318)

[ЗМІСТ ТА РЕЗУЛЬТАТИ ВИКОНАНИХ РОБІТ 3](#_Toc11799319)

[ОПИС ВИКОНАННЯ ЗАВДАНЬ ЗГІДНО З КАЛЕНДАРНИМ ПЛАНОМ 4](#_Toc11799320)

[ЗМІСТ ПРОВЕДЕНИХ НАВЧАЛЬНИХ ЗАНЯТЬ ТА ЕКСКУРСІЙ 5](#_Toc11799321)

[Обговорення теоретичних даних 5](#_Toc11799322)

[ОПИС ТЕХНОЛОГІЙ 6](#_Toc11799323)

[Google Cloud 6](#_Toc11799324)

[Google Cloud database, storage and data analytics services 6](#_Toc11799325)

[BigQuery 7](#_Toc11799326)

[Cloud Bigtable 7](#_Toc11799327)

[Cloud Datastore 8](#_Toc11799328)

[Cloud Spanner 8](#_Toc11799329)

[Cloud SQL 9](#_Toc11799330)

[Cloud Storage 10](#_Toc11799331)

[Firebase Hosting 10](#_Toc11799332)

[Реляційна база даних 12](#_Toc11799333)

[SQL 12](#_Toc11799334)

[MySQL 14](#_Toc11799335)

[SQLite 15](#_Toc11799336)

[Microsoft SQL Server 17](#_Toc11799337)

[ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ 29](#_Toc11799338)

ХАРАКТЕРИСТИКА БАЗИ ПРАКТИКИ

Прибувши на базу практики ТОВ "Об’єднання ЮГ", було проведено коротку екскурсію щодо діяльності компанії, її напрямів розвитку та приклади робіт. Визначили, що видавництво спеціалізується на електронному продажу, а саме в створенні пропозицій на ринку електронного обладнання та програмного забезпечення.

ТОВ «Об'єднання ЮГ» працює на ринку інформаційних технологій c 1993 року, надаючи послуги в області системної інтеграції, інформаційної безпеки, розробки прикладного програмного забезпечення.

Місія Компанії - донести свій багаторічний професійний досвід до своїх клієнтів. Стаж роботи на ринку інформаційних технологій України дозволяє нам з максимальною ефективністю вирішувати проблеми наших замовників і зберігати довгі і теплі бізнес-відносини на високому професійному рівні.

Над розробкою і впровадженням інформаційних проектів в компанії працюють сертифіковані фахівці різних напрямків - менеджери проектів, програмісти, фахівці з мережних технологій, фахівці з засобам технічного та криптографічного захисту інформації, з продажу та маркетингу. Багато з них мають сертифікати різних категорій від провідних виробників технічних інформаційних засобів і програмного забезпечення.

ЗМІСТ ТА РЕЗУЛЬТАТИ ВИКОНАНИХ РОБІТ

**Індивідуальне завдання**

Спроектувати БД для потреб виробництва, враховуючи вимоги технічного завдання. Реалізувати спроектовану БД, застосовуючи відповідні технічні засоби, такі як MySQL, MS SQL, SQLite, Google Cloud. Протестувати БД в умовах, наближених до реальної роботи. Відповідні системи керування базами даних повинні відповідати певним критеріям, таким як:

* Контроль за надлишковістю даних
* Несуперечливість даних
* Підтримка цілісності бази даних (коректність та несуперечливість)
* Цілісність описується за допомогою обмежень
* Незалежність прикладних програм від даних
* Спільне використання даних
* Підвищений рівень безпеки

За умови задовільного проходження тестів, імплементувати протестовану БД у виробничий процеc. БД заповнюються реальними даними, налаштовується доступ, здійснюються всі необхідні заходи для введеня БД в постійну експлуатацію.

ОПИС ВИКОНАННЯ ЗАВДАНЬ ЗГІДНО З КАЛЕНДАРНИМ ПЛАНОМ

**03.06** Прибуття на базу практики. Ознайомлення з специфікою роботи. Проходження інструктажу з техніки безпеки. Узгодження календарного графіку проходження практики з керівником бази практики, вирішення інших організаційних питань проходження практики.

**04.06** Остаточне узгодження індивідуального завдання. Початок роботи над індивідуальним завданням.

**05.06** Проектування схем нових БД.

**06.06** Звірення отриманих схем нових БД зі схемами, спроектованими колегами по індивідуальному завданню. Перевірка дотримання початкових вимог.

**07.06 - 10.06** Реалізація БД засобами MySQL, MS SQL, SQLite та Google Cloud.

**11.06 - 14.06** Тестування створених БД в умовах, наближених до реальної роботи (симуляція навантаження).

**15.06** Вихідний день.

**16.06** Вихідний день.

**17.06** Вихідний день.

**18.06 - 19.06** Імпелементація створених та протестованих БД у виробничий процес.

**20.06** Фінальнеоформлення звіту та інших документів про практику.

**21.06** Захист звітів з технологічної практики.

**22.06** Вихідний день.

**23.06** Вихідний день.

ЗМІСТ ПРОВЕДЕНИХ НАВЧАЛЬНИХ ЗАНЯТЬ ТА ЕКСКУРСІЙ

Обговорення теоретичних даних

Під час проходження практики було здійснено екскурсію по головному офісу ТОВ "Об’єднання ЮГ", після чого ознайомились на практиці з їх продукцією та протестували її на практиці.

ОПИС ТЕХНОЛОГІЙ

Google Cloud

Google Cloud Platform – запропонований компанією Google набір хмарних служб, які виконуються на тій же самій інфраструктурі, яку Google використовує для своїх продуктів призначених для кінцевих споживачів, таких як Google Search та YouTube. Окрім інструментів для керування, також надається ряд модульних хмарних служб, таких як обчислення, зберігання даних, аналіз даних та машинне навчання..

Google Cloud Platform надає такі оточення як інфраструктура як послуга, платформа як послуга, та безсерверні обчислення. Google Cloud Platform є частиною Google Cloud, який також включає G Suite, корпоративні версії Android та Chrome OS, а також API для машинного навчання та Google Maps.

Google Cloud database, storage and data analytics services

Повністю керовані, масштабовані служби баз даних для підтримки всіх ваших додатків сьогодні і завтра. Заряджайте свої програми. Прискорення впровадження з широкою сумісністю баз даних з відкритим кодом. Більше використовуйте свої дані за допомогою інтеграції з аналітикою та ML/AI. Зберігання даних для всіх ваших потреб. Ефективно захоплюйте, обробляйте та аналізуйте дані за допомогою продуктів Google Analytic для аналізу даних.

Найпоширеніші з них є:

* **BigQuery** – Швидке, економічне і повністю кероване сховище даних
* **Cloud Bigtable** – Високопродуктивний сервіс бази даних NoSQL для аналізу в режимі реального;
* **Cloud Datastore** – Масштабна база даних NoSQL для мобільних і веб-додатків;
* **Cloud Spanner –** Перший сервіс глобально узгоджених реляційних баз даних з горизонтальним;
* **Cloud SQL** – Повністю керований сервіс баз даних MySQL і PostgreSQL;
* **Cloud Storage** – Розпочніть роботу зі створення бакета, контейнера для ваших об'єктів у Cloud Storage;
* **Cloud Storage for Firebase** – Просте зберігання контенту;
* **Firebase Hosting** – Швидка доставка веб-контенту;
* **Firebase Realtime Database** – Зберігайте і синхронізуйте дані додатків в режимі реального часу.

BigQuery

BigQuery – це серверне, високо масштабоване та економічно ефективне сховище даних з хмарами, з вбудованим вбудованим BI-движком у пам'яті.

BigQuery, безсерверне сховище даних компанії, що не використовує сервер, розроблене для того, щоб зробити аналітиків даних більш продуктивними з неперевершеними ціновими показниками. Оскільки не існує інфраструктури для керування, ви можете зосередитися на розкритті значущої інформації, використовуючи знайомий SQL, без необхідності адміністратора бази даних.

Проаналізуйте всі ваші пакетні та потокові дані, створивши логічне сховище даних над керованим сховищем стовпців, а також дані з сховища об'єктів та електронних таблиць. Створюйте надзвичайно швидкі панелі та звіти з BI-движком у пам'яті. Побудувати та ввести в дію рішення для машинного навчання або провести геопросторовий аналіз, використовуючи простий SQL.

Потужне потокове передавання даних BigQuery дозволяє переглядати та аналізувати дані в режимі реального часу, забезпечуючи постійне поширення інформації. Крім того, ви можете аналізувати до 1 ТБ даних і зберігати 10 ГБ даних безкоштовно щомісяця.

Cloud Bigtable

Gовністю керована NoSQL служба баз даних для великих (petabyte) аналітичних та операційних навантажень.

Cloud Bigtable як механізм зберігання даних для великомасштабних додатків з низькою затримкою, а також для обміну даними та аналітики.

Забезпечують і масштабують до сотні петабайт і плавно обробляють мільйони операцій в секунду. Зміни в конфігурації розгортання є негайними, тому під час реконфігурації немає простоїв.

Реплікація додає високу доступність для додатків, які обслуговують у прямому ефірі, та ізоляцію робочого навантаження для обслуговування та аналітики.

Cloud Bigtable легко інтегрується з популярними інструментами великих даних, такими як Hadoop, Cloud Dataflow і Cloud Dataproc. Крім того, Cloud Bigtable підтримує стандартний стандартний інтерфейс HBase API, що полегшує роботу ваших команд розробників.

Оскільки ми керуємо базою даних і керуємо налаштуванням і налаштуванням, можна зосередитися на розробці додатків.

Cloud Datastore

Cloud Datastore - це масштабована база даних NoSQL для ваших веб- і мобільних додатків.

Cloud Datastore автоматично обробляє sharding і реплікацію, надаючи вам доступну і довговічну базу даних, яка автоматично масштабується для обробки навантажень ваших додатків. Cloud Datastore надає безліч можливостей, таких як транзакції ACID, SQL-подібні запити, індекси та багато іншого.

Завдяки інтерфейсу RESTful Cloud Datastore, доступ до даних може здійснюватися будь-якою ціллю розгортання. Ви можете створювати рішення, які охоплюють App Engine і Compute Engine, і покладатися на Cloud Datastore як на точку інтеграції.

Зосередьження на створенні своїх додатків, не турбуючись про надання та завантаження. Cloud Datastore масштабується плавно і автоматично з вашими даними, що дозволяє програмам підтримувати високу продуктивність, оскільки вони отримують більше трафіку.

Datastore - це база даних, яка дозволяє вам не турбуватися про внесення змін до базової структури даних, коли ваша програма розвивається. Datastore надає потужний механізм запитів, який дозволяє здійснювати пошук даних по декількох властивостях і сортувати за необхідності.

Cloud Spanner

Cloud Spanner – це перша горизонтально масштабована, сильно послідовна, реляційна служба бази даних.

Cloud Spanner є єдиним корпоративним, глобально розподіленим і сильно послідовним сервісом баз даних, побудованим для хмари, спеціально для об'єднання переваг реляційної структури бази даних з нереляційною горизонтальною шкалою. Ця комбінація забезпечує високопродуктивні транзакції та високу узгодженість між рядками, регіонами та континентами, що мають провідну угоду SLA про доступність у 99,999%, без планових простоїв та безпеки корпоративного рівня. Cloud Spanner революціонізує адміністрування та управління базами даних і робить розробку додатків більш ефективною.

Більшість баз даних сьогодні вимагають ухвалення компромісів між масштабами та послідовністю. За допомогою Cloud Spanner ви отримуєте найкращу структуру реляційних баз даних та масштаби нереляційних баз даних та продуктивність із зовнішнім сильним узгодженням між рядками, регіонами та континентами.

ІТ-адміністратори та адміністратори баз даних завалені невдячними завданнями. Завдяки Cloud Spanner зосередитися на доданні цінностей та інноваціях замість технічного обслуговування. Створення або масштабування глобально реплікованої бази даних для критично важливих програм тепер займає кілька кліків.

Високодоступні та високошвидкісні системи безпеки Google, як стандартні, а не дорогі додатки, допомагають забезпечити безпеку ваших програм у режимі онлайн. При архітектурі додатків Cloud Spanner має просту модель виставлення рахунків, яка не стягує додаткову плату за вибір конфігурації для високої доступності, реплікації або забезпечення більш безпечних, глобальних сильно узгоджених даних.

Cloud Spanner - це реляційна база даних з повною реляційною семантикою і обробляє зміни схеми як онлайн-операцію без планового простою. Повторне використання наявних навичок SQL для запиту даних у Cloud Spanner з використанням знайомого, стандартного стандарту ANSI 2011 SQL.

Забезпечення кращого досвіду для кінцевих користувачів із зовнішньою, сильною послідовністю. Програми повинні бути побудовані для глобальної бази користувачів і вам потрібна синхронна реплікація для надання правильних, точних і послідовних даних. Альтернативою є, можливо, непослідовне старенне читання, зменшений досвід роботи з кінцевим користувачем і складна логіка застосування, що призводить до складних, трудомістких помилок.

Cloud SQL

Хмара SQL для SQL Server.

Cloud SQL - це повністю керована служба баз даних, яка дозволяє легко налаштовувати, підтримувати, керувати та керувати реляційними базами даних PostgreSQL, MySQL і SQL Server у хмарі. Cloud SQL пропонує високу продуктивність, масштабованість і зручність. Розташований на Google Cloud Platform, Cloud SQL надає інфраструктуру бази даних для програм, які працюють де завгодно.

Дозволяє Google керувати вашою базою даних, щоб можна було зосередитися на своїх програмах. Cloud SQL ідеально підходить для сайтів WordPress, додатків електронної комерції, інструментів CRM, геопросторових додатків і будь-яких інших програм, сумісних з MySQL, PostgreSQL або SQL Server.

Cloud SQL забезпечує високу продуктивність і масштабованість з ємністю до 10 ТБ, 40,000 IOPS та 416 Гб оперативної пам'яті на примірник.

Легко налаштувується реплікація та резервні копії для захисту даних. Перейдіть далі, включивши автоматичне перехід на інший, щоб зробити вашу базу даних доступною (HA). Ваші дані автоматично шифруються, а Cloud SQL - SSAE 16, ISO 27001, PCI DSS v3.0 і підтримує відповідність HIPAA.

Cloud Storage

Єдиний об'єкт зберігання для розробників і підприємств.

Послідовний інтерфейс API Cloud Storage, затримка та швидкість у різних класах зберігання спрощують інтеграцію розробки та зменшують складність коду. Реалізуйте керування життєвим циклом об'єктів, щоб встановити час для живих (TTL) об'єктів, архівувати старішу версію об'єктів або знижувати класи зберігання, не порушуючи затримки або доступності. Налаштуйте власні політики для безперешкодного перенесення даних з одного класу зберігання на інший, залежно від потреб у вашій вартості та доступності.

Cloud Storage розрахований на 99,999999999% довговічності. Він зберігає дані надлишково, за допомогою автоматичних контрольних сум для забезпечення цілісності даних. За допомогою мультирегіонального сховища дані зберігаються в географічно різних місцях.

Cloud Storage пропонує необмежений об'єм для зберігання об'єктів, а окремі об'єкти можуть становити до 5TB. Об'єкти можуть бути перезаписані не частіше одного разу в секунду, а частота читання не обмежена. Об'єкти, розмір яких перевищує 5 Мб, слід завантажувати за допомогою багаточастинкового чи повторного завантаження. Щоб отримати докладніші відомості, перегляньте квоти & ліміти Cloud Storage.

Коли запис вдається успішно, остання копія об'єкта гарантовано буде повернута до будь-якого GET, глобально. Це відноситься до PUTs нових або перезаписаних об'єктів і DELETE.

Переміщуючи сховище даних з самостійного центру обробки даних або об'єкта колокації до GCP, викиди, безпосередньо пов'язані з сховищем даних вашої компанії, будуть нульовими.

Firebase Hosting

Зберігати дані програми на основі хмари в глобальному масштабі.

Cloud Firestore - це швидка, повністю керована база даних документів NoSQL без серверів, що спрощує зберігання, синхронізацію та запит даних для мобільних, веб- і IoT-додатків у глобальному масштабі. Її клієнтські бібліотеки забезпечують синхронізацію в реальному часі та підтримку в автономному режимі, а функції безпеки та інтеграції з Firebase та Google Cloud Platform (GCP) прискорюють створення дійсно безсерверних програм.

Cloud Firestore - це база даних, створена у хмарі, яка забезпечує автоматичне масштабування, побудоване з нуля для використання потужної інфраструктури Google Cloud Platform. Він розроблений для того, щоб забезпечити великий досвід розробників і спростити розробку додатків за допомогою синхронізації в реальному часі, підтримки в автономному режимі та транзакцій ACID на сотнях документів і колекцій. Cloud Firestore інтегрована як з Google Cloud Platform (GCP), так і з Firebase, платформою мобільного розвитку Google.

Ви можете розмовляти безпосередньо з Cloud Firestore з мобільного або веб-клієнтів для по-справжньому серверного рішення. Не потрібно встановлювати сервер-посередник для керування доступом до даних. Це відмінно підходить для створення прототипів, ітерації та швидкого запуску виробничої системи.

За допомогою Cloud Firestore ваші програми можуть оновлюватися майже в реальному часі, коли змінюються дані на сервері. Це не тільки зручно для створення спільних багатокористувацьких мобільних додатків, але також означає, що ви можете зберігати свої дані в синхронізації з окремими користувачами, які можуть захотіти використовувати вашу програму з декількох пристроїв.

Cloud Firestore має повну підтримку в автономному режимі, тому ви можете отримувати доступ до своїх даних і вносити зміни, і ці зміни синхронізуються з хмарою, коли клієнт повернеться в онлайн. Вбудована підтримка в режимі офлайн використовує місцевий кеш для обслуговування та зберігання даних, тому програма залишається реагуючою незалежно від затримки мережі або підключення до Інтернету.

Надійні клієнтські бібліотеки Cloud Firestore полегшують оновлення та отримання нових даних, не турбуючись при цьому про встановлення мережевих з'єднань або непередбачувані умови перегонів. Вона може легко масштабуватися, коли ваше додаток зростає. Cloud Firestore дозволяє запускати складні запити щодо ваших даних. Це дає вам більшу гнучкість у способі структурування даних і часто може означати, що ви повинні робити менше фільтрації на клієнті, що зберігає ефективність мережевих дзвінків і використання даних.

Cloud Firestore - це швидка і повністю керована хмарна база даних NoSQL. Вона побудована для масштабування та використання переваг потужної інфраструктури GCP, з автоматичним горизонтальним масштабуванням та виведенням у відповідь на завантаження вашої програми. Вбудовані засоби контролю доступу до даних, які дозволяють здійснювати перевірку даних за допомогою мови налаштування.

Реляційна база даних

Реляційна база даних – це набір даних із зумовленими зв'язками між ними. Ці дані організовані у вигляді набору таблиць, що складаються із стовпців і рядків. У таблицях зберігається інформація про об'єкти, представлених в базі даних. У кожному стовпчику таблиці зберігається певний тип даних, в кожній комірці – значення атрибута. Кожна стоку таблиці являє собою набір пов'язаних значень, що відносяться до одного об'єкту або сутності. Кожен рядок в таблиці може бути позначена унікальним ідентифікатором, званим первинним ключем, а рядки з декількох таблиць можуть бути пов'язані з допомогою зовнішніх ключів. До цих даних можна отримати доступ багатьма способами, і при цьому реорганізовувати таблиці БД не потрібно.

SQL

Structured Query Language – основний інтерфейс роботи з реляційними БД. SQL став стандартом Національного інституту стандартів США (ANSI) в 1986 році. Стандарт ANSI SQL підтримується всіма популярними ядрами реляційних БД. Деякі з ядер також включають розширення стандарту ANSI SQL, що підтримують специфічний для цих ядер функціонал. SQL використовується для додавання, оновлення та видалення рядків даних, вилучення наборів даних для обробки транзакцій і аналітичних програм, а також для управління всіма аспектами роботи бази даних.

SQL трохи відрізняється від своєї теоретичної основи, реляційної моделі та її обчисленням кортежів. У цій моделі таблиця є набором кортежів, а в SQL - таблицях і результатах запиту є списки рядків: один і той же рядок може з'являтися кілька разів, і порядок рядків може бути використаний у запитах (наприклад, у LIMIT) .

Критики стверджують, що SQL повинен бути замінений на мову, яка строго повертається до початкової основи. Проте, не існує жодного відомого доказу, що така унікальність не може бути додана до самого SQL, або принаймні до варіацій SQL. Іншими словами, цілком можливо, що SQL може бути "фіксованим" або принаймні поліпшеним у цьому відношенні таким чином, що промисловість, можливо, не повинна переходити на зовсім іншу мову запитів, щоб отримати унікальність. Дебати з цього приводу залишаються відкритими.

SQL призначений для конкретної мети: для запиту даних, що містяться в реляційної базі даних. SQL - це декларативний мову програмування, заснований на множинах, а не імперативний мову програмування, такий як C або BASIC. Однак розширення до стандартного SQL додають функціональні можливості мови процедурного програмування, такі як конструкції управління потоком.

На додаток до стандартних розширень SQL / PSM і пропрієтарним розширенням SQL, процедурне і об'єктно-орієнтована програмування доступне на багатьох платформах SQL за допомогою інтеграції СУБД з іншими мовами. Стандарт SQL визначає розширення SQL / JRT (SQL Routines і Types для мови програмування Java) для підтримки коду Java в базах даних SQL. SQL Server 2005 використовує SQLCLR (SQL Server Common Language Runtime) для розміщення керованих зборок .NET в базі даних, в той час як попередні версії SQL Server були обмежені некерованими розширеними збереженими процедурами, в першу чергу написаними на C. PostgreSQL дозволяє користувачам писати функції в широкому розмаїтті. мов, включаючи Perl, Python, Tcl, JavaScript (PL / V8) і C.

Цілісність даних – це повнота, точність і однаковість даних. Для підтримки цілісності даних в реляційних БД використовується ряд інструментів. В їх число входять первинні ключі, зовнішні ключі, обмеження «Not NULL», «Unique», «Default» і «Check». Ці обмеження цілісності дозволяють застосовувати практичні правила до даних в таблицях і гарантувати точність і надійність даних. Більшість ядер БД також підтримує інтеграцію призначеного для користувача коду, який виконується у відповідь на певні операції в БД.

Для дотримання цілісності даних всі транзакції в БД повинні відповідати вимогам ACID, тобто бути атомарними, однаковими, ізольованими і надійними. Атомарність – це умова, при якому або транзакція успішно виконується цілком, або, якщо будь-яка з її частин не виконується, вся транзакція скасовується. Одноманітність - це умова, при якому дані, що записуються в базу даних в рамках транзакції, повинні відповідати всім правилам і обмеженням, включаючи обмеження цілісності, каскади і тригери. Ізольованість необхідна для контролю над узгодженістю і гарантує базову незалежність кожної транзакції. Надійність має на увазі, що всі внесені в базу даних зміни на момент успішного завершення транзакції вважаються постійними.

Транзакція в базі даних – це один або кілька операторів SQL, виконаних у вигляді послідовності операцій, що представляють собою єдину логічну задачу. Транзакція являє собою неподільне дію, тобто вона повинна бути виконана як єдине ціле і або повинна бути записана в базу даних цілком, або не повинен бути записаний жоден з її компонентів. У термінології реляційних баз даних транзакція завершується або дією COMMIT, або ROLLBACK. Кожна транзакція розглядається як внутрішньо зв'язний, надійний і незалежний від інших транзакцій елемент.

Реалізації SQL несумісні між постачальниками і не обов'язково повністю відповідають стандартам. Зокрема, синтаксис дати і часу, конкатенація рядків, NULL і чутливість до регістру порівняння варіюються від постачальника до постачальника. Конкретними виключеннями є PostgreSQL і Mimer SQL, які прагнуть до відповідності стандартам, хоча PostgreSQL не дотримуватися стандарту в тому, як відбувається згортання імен без лапок. Згортання імен без лапок в нижній регістр в PostgreSQL несумісне зі стандартом SQL, в якому говориться, що імена без лапок повинні бути зігнуті в верхній регістр. Таким чином, Foo повинен бути еквівалентний FOO, а не foo відповідно до стандарту.

MySQL

MySQL – вільна система керування реляційними базами даних. MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблиць, для яких використовувалися ISAM — підпрограми низького рівня для індексного доступу до даних. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. Звідки походить назва «MySQL» — достеменно не відомо. Розробники дають два варіанти: або тому, що практично всі напрацювання компанії починалися з префікса My, або на честь дівчинки на ім'я My, дочки Майкла Монті Віденіуса, одного з розробників системи.

MySQL був створений шведською компанією MySQL AB, заснованої Девідом Аксмарком, Алланом Ларссоном і Майклом "Монті" Віденіус. Перша розробка MySQL Widenius і Axmark почалася в 1994 році. Перша версія MySQL з'явилася 23 травня 1995 року. Спочатку вона була створена для особистого використання з mSQL на основі низкоуровневої мови ISAM, який творці вважали занадто повільною і негнучкою. Вони створили новий інтерфейс SQL, зберігши той же API, що і mSQL. Підтримуючи API відповідно до системи mSQL, багато розробників могли використовувати MySQL замість попередньої версії mSQL (за власною ліцензією).

Логотип MySQL у вигляді дельфіна носить ім'я «Sakila». Він був обраний з великого списку запропонованих користувачами «імен дельфіна». Ім'я «Sakila» було відправлено Open Source-розробником Ambrose Twebaze.

За час розвитку під орудою Oracle дедалі більше відокремлює MySQL від спільноти і робить процес розробки все менш прозорим. Наприклад, повернута практика поставки власницьких розширених функцій в Enterprise-версії MySQL, спостерігається приховування інформації про вразливості, зі складу виключений тестовий набір, закритий доступ до більшої частини системи відстеження помилок та припинено публікація згрупованого логу змін, що дозволяє судити про прив'язку патчів до конкретних змін.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

* простота у встановленні та використанні;
* підтримується необмежена кількість користувачів, що одночасно працюють із БД;
* кількість рядків у таблицях може досягати 50 млн;
* висока швидкість виконання команд;
* наявність простої і ефективної системи безпеки.

SQLite

SQLite – полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання (англ. public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті постачання йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.

Завдяки архітектурі рушія можливо використовувати SQlite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

Особливості SQLite:

* Транзакції атомарні, послідовні, ізольовані, і міцні (ACID) навіть після збоїв системи і збоїв живлення;
* Встановлення без конфігурації — не потребує ані установки, ані адміністрування;
* Реалізує значну частину стандарту SQL92;
* База даних зберігається в одному крос-платформовому файлі на диску;
* Підтримка терабайтних розмірів баз даних і гігабайтного розміру рядків і BLOBів;
* Малий розмір коду: менше ніж 350KB повністю налаштований, і менш 200KB без додаткових функцій;
* Швидший за популярні двигуни клієнт-серверних баз даних для найпоширеніших операцій;
* Простий, легкий у використанні API;
* Написана в ANSI C, включена прив'язка до TCL; доступні також прив'язки для десятків інших мов;
* Добре прокоментований сирцевий код зі 100% тестовим покриттям гілок;
* Доступний як єдиний файл сирцевого коду на ANSI C, який можна легко вставити в інший проект;
* Автономність: немає зовнішніх залежностей;
* Крос-платформовість: з коробки підтримується Unix (Linux і Mac OS X), OS/2, Windows (Win32 і WinCE). Легко переноситься на інші системи;
* Сирцевий код перебуває в суспільному надбанні;
* Поставляється з автономним клієнтом інтерфейсу командного рядка, який може бути використаний для управління базами даних SQLite.

На відміну від систем управління базами даних клієнт-сервер, ядро ​​SQLite не має автономних процесів, з якими взаємодіє прикладна програма. Замість цього бібліотека SQLite пов'язана і, таким чином, стає невід'ємною частиною прикладної програми. Зв'язування може бути статичним або динамічним. Прикладна програма використовує функціональні можливості SQLite за допомогою простих викликів функцій, які зменшують затримку при доступі до бази даних: виклики функцій в рамках одного процесу більш ефективні, ніж міжпроцессна взаємодія.

SQLite зберігає всю базу даних (визначення, таблиці, індекси і самі дані) у вигляді одного кроссплатформенного файлу на хост-комп'ютері. Він реалізує цю просту конструкцію, блокуючи весь файл бази даних під час запису. Операції читання SQLite можуть бути багатозадачними, хоча запис може виконуватися тільки послідовно.

Через безсерверной конструкції додатки SQLite вимагають менше конфігурації, ніж бази даних клієнт-сервер. SQLite називається zero-conf, тому що він не вимагає управління службами (такими як сценарії запуску) або контролю доступу на основі GRANT і паролів. Управління доступом здійснюється за допомогою дозволів файлової системи, наданих самому файлу бази даних. Бази даних в системах клієнт-сервер використовують дозволу файлової системи, які надають доступ до файлів бази даних тільки процесу демона.

Ще одним наслідком безсерверного дизайну є те, що деякі процеси можуть не мати можливості запису в файл бази даних. У серверних базах даних кілька письменників підключаються до одного і того ж демона, який може внутрішньо обробляти свої блокування. SQLite, з іншого боку, повинен покладатися на блокування файлової системи. У нього менше знань про інших процесах, які одночасно звертаються до бази даних. Тому SQLite не є кращим вибором для розгортання з інтенсивної записом. Проте, для простих запитів з невеликим паралелізмом продуктивність SQLite виграє від уникнення накладних витрат, пов'язаних з передачею даних іншому процесу.

SQLite використовує PostgreSQL в якості еталонної платформи. «Що б PostgreSQL робив» використовується для розуміння стандарту SQL. Одним з основних відмінностей є те, що, за винятком первинних ключів, SQLite не забезпечує перевірку типів; тип значення є динамічним і не строго обмежений схемою (хоча схема запустить перетворення при збереженні, якщо таке перетворення потенційно оборотно). SQLite прагне дотримуватися правила Постеля.

Код SQLite розміщується в Fossil, розподіленої системі контролю версій, яка сама побудована на базі даних SQLite.

У дистрибутиві SQLite передбачена окрема програма командного рядка. Його можна використовувати для створення бази даних, визначення таблиць, вставки і зміни рядків, виконання запитів і управління файлом бази даних SQLite. Він також служить прикладом для створення програмного забезпечення, що використовують бібліотеку SQLite.

SQLite використовує автоматичне регресійне тестування перед кожним випуском. Більше 2 мільйонів тестів виконуються як частина перевірки релізу. Починаючи з випуску SQLite 3.6.17 від 10 серпня 2009 року, випуски SQLite мають 100% покриття тестів розгалуження, один з компонентів покриття коду. Тести і тестові набори є частково суспільним надбанням і частково пропрієтарними.

Microsoft SQL Server

Microsoft SQL Server — комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

*Архітектура*

Рівень протоколу реалізує зовнішній інтерфейс для SQL Server. Всі операції, які можна викликати на

SQL Server, повідомляються йому через визначений у Microsoft формат, який називається табличним потоком даних (TDS). TDS - це протокол прикладного рівня, який використовується для передачі даних між сервером баз даних і клієнтом. Первинно розроблений і розроблений компанією Sybase Inc. для свого реляційного двигуна Sybase SQL Server у 1984 році, а пізніше - Microsoft у Microsoft SQL Server, пакети TDS можуть бути укладені в інші фізичні транспортні протоколи, включаючи TCP / IP, назви труб пам'яті. Отже, доступ до SQL Server доступний через ці протоколи. Крім того, API SQL Server також відкритий через веб-сервіси.

*Data storage*

Data storage - це база даних, яка являє собою набір таблиць з типізованими стовпцями. SQL Server підтримує різні типи даних, включаючи примітивні типи, такі як Integer, Float, Decimal, Char (включаючи рядки символів), Varchar (рядки символів змінної довжини), двійкові (для неструктурованих крапель даних), Text (для текстових даних) серед інших . Округлення поплавців до цілих чисел використовує або Симетричне арифметичне округлення, або Симетричне округлення вниз (fix) залежно від аргументів: SELECT Round (2.5, 0) дає 3.

Microsoft SQL Server також дозволяє визначити та використовувати визначені користувачем складові типи (UDT). Вона також робить статистику сервера доступною у вигляді віртуальних таблиць і переглядів (званих Динамічними переглядами керування або DMV). Крім таблиць, база даних також може містити інші об'єкти, включаючи види, збережені процедури, індекси та обмеження, а також журнал транзакцій. База даних SQL Server може містити не більше 231 об'єктів і може охоплювати декілька файлів на рівні ОС з максимальним розміром файлу 260 байт (1 екзабайт). Дані в базі даних зберігаються в первинних файлах даних з розширенням .mdf. Файли вторинних даних, ідентифіковані з розширенням .ndf, використовуються для того, щоб дозволити дані однієї бази даних поширюватися більш ніж в одному файлі, а при необхідності - на більш ніж одній файловій системі. Файли журналів ідентифікуються розширенням .ldf.

Простір для зберігання, виділений для бази даних, розділяється на сторінки з послідовно пронумерованими сторінками, кожен розміром 8 КБ. Сторінка є основною одиницею вводу-виводу для операцій SQL Server. Сторінка позначена 96-байтовим заголовком, який зберігає метадані про сторінку, включаючи номер сторінки, тип сторінки, вільний простір на сторінці та ідентифікатор об'єкта, що його має. Тип сторінки визначає дані, що містяться на сторінці: дані, що зберігаються в базі даних, індекс, карта розподілу, яка містить інформацію про те, як сторінки виділяються для таблиць і індексів, змінюють карту, яка містить інформацію про зміни, зроблені на інших сторінках з моменту останнього резервного копіювання або реєстрації або містять великі типи даних, такі як зображення або текст. Хоча сторінка є базовою одиницею операції введення-виведення, простір фактично управляється в термінах, які складаються з 8 сторінок. Об'єкт бази даних може або охоплювати всі 8 сторінок в масштабі ("рівномірний розмір") або спільно використовувати об'єм до 7 додаткових об'єктів ("змішана міра"). Рядок у таблиці баз даних не може охоплювати більше однієї сторінки, тому він обмежений розміром 8 Кб. Однак, якщо дані перевищують 8 КБ, а рядок містить дані varchar або varbinary, дані в цих стовпцях переміщуються на нову сторінку (або, можливо, послідовність сторінок, яка називається блоком розподілу) і замінюється покажчиком на дані.

Для фізичного зберігання таблиці її рядки поділяються на ряд розділів (з нумерацією від 1 до n). Розмір розділу визначається користувачем; за замовчуванням всі рядки знаходяться в одному розділі. Таблиця розділена на декілька розділів, щоб поширити базу даних на комп'ютерний кластер. Рядки в кожному розділі зберігаються в структурі B-tree або heap. Якщо таблиця має асоційований кластерний індекс, що дозволяє швидко отримувати рядки, рядки зберігаються в порядку відповідно до їх значень індексу, при цьому в B-дереві передбачено індекс. Дані знаходяться в листовому вузлі листя, а інші вузли зберігають значення індексу для даних листів, доступних з відповідних вузлів. Якщо індекс не кластеризований, рядки не сортуються відповідно до ключів індексу. Індексований вигляд має ту ж саму структуру зберігання, що й індексована таблиця. Таблиця без кластерного індексу зберігається в невпорядкованій купі. Однак таблиця може мати некластерізовані індекси, що дозволяють швидко отримувати рядки. У деяких ситуаціях структура купи має переваги в продуктивності над кластерною структурою. Обидва кучі та B-дерева можуть охоплювати кілька одиниць виділення.

*Управління буфером*

SQL Server буферизує сторінки в оперативній пам'яті, щоб звести до мінімуму диск вводу-виводу. Будь-яка сторінка розміром 8 Кб може бути буферизована в пам'яті, а набір всіх поточних буферизованих сторінок називається буферним кешем. Обсяг пам'яті, доступний SQL Server, вирішує, скільки сторінок буде кешуватися в пам'яті. Кеш буфера керується менеджером буферів. Читання чи записування на будь-яку сторінку копіює її в буферний кеш. Наступні читання або запису перенаправляються на копію в пам'яті, а не на версію диска. Сторінка оновлюється на диску менеджером буферів лише в тому випадку, якщо на кеші в пам'яті не посилається деякий час. Під час запису сторінок на диск використовується асинхронний ввід / вивід, за допомогою якого операція вводу-виводу виконується у фоновому потоці, так що інші операції не повинні чекати завершення операції введення-виведення. Кожна сторінка написана разом з її контрольною сумою, коли вона написана. При читанні сторінки назад її контрольна сума обчислюється знову і узгоджується з збереженою версією, щоб переконатися, що сторінка не була пошкоджена або підроблена тим часом.

*Паралельність і блокування*

SQL Server дозволяє декільком клієнтам використовувати одну і ту ж базу даних одночасно. Таким чином, він повинен контролювати одночасний доступ до загальних даних, щоб гарантувати цілісність даних - коли кілька клієнтів оновлюють ті ж самі дані, або клієнти намагаються прочитати дані, які перебувають у процесі зміни іншого клієнта. SQL Server забезпечує два режими управління паралелізмом: песимістичний паралелізм і оптимістичний паралелізм. Коли використовується песимістичний контроль паралельності, SQL Server контролює одночасний доступ за допомогою блокування. Замки можуть бути спільними або ексклюзивними. Ексклюзивний замок надає користувачеві ексклюзивний доступ до даних - жоден інший користувач не може отримати доступ до даних, поки блокування тримається. Спільні замки використовуються, коли деякі дані зчитуються - численні користувачі можуть читати з даних, заблокованих спільним блокуванням, але не отримують ексклюзивного блокування. Останній буде чекати, коли всі спільні замки будуть випущені.

Замки можуть бути застосовані на різних рівнях деталізації - на цілих таблицях, сторінках або навіть на окремих таблицях. Для індексів він може бути або на всьому індексі, або на індексі. Рівень деталізації, який буде використовуватися, визначається адміністратором бази даних на основі бази даних. Хоча дрібнозерниста система блокування дозволяє більше користувачів використовувати таблицю або індекс одночасно, вона вимагає більших ресурсів, тому вона не дає автоматично більшої продуктивності. SQL Server також включає в себе два більш легкі рішення для взаємного виключення - засувки і закрутки, які є менш надійними, ніж замки, але менш ресурсомісткі. SQL Server використовує їх для DMV та інших ресурсів, які зазвичай не зайняті. SQL Server також відстежує всі робочі потоки, які купують замки, щоб переконатися, що вони не потрапляють у глухий кут - у випадку, якщо вони це роблять, SQL Server вживає заходів щодо виправлення, які в багатьох випадках вбивають один з потоків, заплутаних у глухий кут, і повертають назад транзакцію, яку вона розпочала. Для реалізації блокування SQL Server містить менеджер блокування. Менеджер блокування підтримує таблицю пам'яті, яка керує об'єктами бази даних і блокує, якщо такі є, на них разом з іншими метаданими про блокування. Доступ до будь-якого спільного об'єкта опосередковується менеджером блокування, який або надає доступ до ресурсу, або блокує його.

SQL Server також надає оптимістичний механізм управління паралелізмом, який схожий на багатоваріантне управління паралелізмом, що використовується в інших базах даних. Механізм дозволяє створювати нову версію рядка кожного разу, коли рядок оновлюється, на відміну від перезапису рядка, тобто рядок додатково ідентифікується ідентифікатором транзакції, яка створила версію рядка. Як старі, так і нові версії рядка зберігаються і зберігаються, хоча старі версії переміщуються з бази даних в системну базу даних, позначену як Tempdb. Коли рядок перебуває в процесі оновлення, будь-які інші запити не блокуються (на відміну від блокування), але виконуються на старій версії рядка. Якщо інший запит є оператором оновлення, це призведе до двох різних версій рядків - обидва з них будуть зберігатися в базі даних, ідентифікованої їх відповідними ідентифікаторами транзакцій.

*Вилучення даних і програмованість*

Основним режимом отримання даних з бази даних SQL Server є запит до неї. Запит виражається за допомогою варіанту SQL під назвою T-SQL, діалект Microsoft SQL Server спільно використовує Sybase SQL Server у зв'язку з його спадщиною. Запит декларативно вказує, що потрібно отримати. Він обробляється процесором запитів, який визначає послідовність кроків, необхідних для отримання запитаних даних. Послідовність дій, необхідних для виконання запиту, називається планом запитів. Можливо, існує кілька способів обробки одного і того ж запиту. Наприклад, для запиту, що містить оператор приєднання і оператор select, виконуючи об'єднання на обох таблицях, а потім виконуючи вибір за результатами, буде отримано той же результат, що і вибір з кожної таблиці, а потім виконання приєднання, але приводить до іншого виконання планів. У такому випадку SQL Server вибирає план, який, як очікується, дасть результати в найкоротші терміни. Це називається оптимізацією запитів і виконується самим процесором запитів.

SQL Server включає в себе оптимізатор запитів на основі витрат, який намагається оптимізувати вартість, з точки зору ресурсів, необхідних для виконання запиту. Враховуючи запит, оптимізатор запитів розглядає схему бази даних, статистику бази даних і завантаження системи в той час. Потім він вирішує, яка послідовність для доступу до таблиць, згаданих у запиті, яка послідовність для виконання операцій і який метод доступу повинен використовуватися для доступу до таблиць. Наприклад, якщо таблиця має асоційований індекс, чи слід використовувати індекс: якщо індекс знаходиться на стовпці, який не є унікальним для більшості стовпців (низька "вибірковість"), не варто використовувати індекс для доступу до даних. Нарешті, він вирішує, чи слід виконувати запит одночасно чи ні. Хоча одночасне виконання є більш дорогим з точки зору загального часу процесора, оскільки виконання насправді розбивається на різні процесори, може означати, що він буде виконуватися швидше. Як тільки план запиту буде сформовано для запиту, він тимчасово кешується. Для подальших викликів одного і того ж запиту використовується кешований план. Невикористані плани відкидаються через деякий час.

SQL Server також дозволяє визначати збережені процедури. Збережені процедури є параметризованими T-SQL запитами, які зберігаються на самому сервері (і не видаються клієнтським додатком, як у випадку з загальними запитами). Збережені процедури можуть приймати значення, надіслані клієнтом як вхідні параметри, і повертати результати як вихідні параметри. Вони можуть викликати певні функції та інші збережені процедури, включаючи ту ж збережену процедуру (до певної кількості разів). Їм можна вибірково надавати доступ. На відміну від інших запитів, збережені процедури мають асоційоване ім'я, яке використовується під час виконання для вирішення фактичних запитів. Крім того, оскільки код не повинен бути відправлений від клієнта кожного разу (як це можна отримати за ім'ям), це зменшує мережевий трафік і дещо підвищує продуктивність. Плани виконання для збережених процедур також кешуються в міру необхідності.

*T-SQL*

T-SQL (Transact-SQL) є пропрієтарним розширенням мови Microsoft для SQL Server. Він надає інструкції REPL (Read-Eval-Print-Loop), які розширюють стандартний набір інструкцій SQL для команд маніпуляції даними (DML) і даних (DDL), включаючи параметри, специфічні для SQL Server, управління статистикою безпеки та бази даних.

Вона надає ключові слова для операцій, які можуть бути виконані на SQL Server, включаючи створення та зміну схем бази даних, введення та редагування даних у базі даних, а також моніторинг і керування самим сервером. Клієнтські програми, які споживають дані або керують сервером, використовують функціональні можливості SQL Server, посилаючи T-SQL запити і оператори, які потім обробляються сервером, і результати (або помилки) повертаються клієнтському додатку. Для цього він відкриває таблиці лише для читання, з яких можна прочитати статистику сервера. Функціональні можливості управління піддаються через системно визначені збережені процедури, які можуть бути викликані з T-SQL запитів для виконання операції керування. Також можна створювати пов'язані сервери за допомогою T-SQL. Пов'язані сервери дозволяють одному запиту обробляти операції, що виконуються на декількох серверах.

*Рідний клієнт SQL Server (також відомий як. SNAC)*

Рідний клієнт SQL Server є рідною бібліотекою доступу до даних на стороні клієнта для Microsoft SQL Server, версія 2005 року. Вона впроваджує підтримку функцій SQL Server, включаючи реалізацію табличного потоку даних, підтримку дзеркальних баз даних SQL Server, повну підтримку всіх типів даних, що підтримуються SQL Server, асинхронні операції, повідомлення про запити, підтримку шифрування, а також отримання безлічі наборів результатів в одному сеансі бази даних. Кінцевий клієнт SQL Server використовується під капотом за допомогою плагінів SQL Server для інших технологій доступу до даних, включаючи ADO або OLE DB. Рідний клієнт SQL Server також може бути безпосередньо використаний, обходячи загальні шари доступу до даних.

28 листопада 2011 року було випущено попередній випуск драйвера ODBC для SQL Server для Linux.

*SQL CLR*

Microsoft SQL Server 2005 включає компонент з назвою SQL CLR ("Common Language Runtime"), за допомогою якого він інтегрується з .NET Framework. На відміну від більшості інших додатків, що використовують .NET Framework, сам SQL Server розміщує середовище виконання .NET Framework, тобто, вимоги до пам'яті, потоків і управління ресурсами .NET Framework задовольняються самим SQLOS, а не базовою операційною системою Windows. SQLOS також надає послуги з виявлення та усунення перешкод для коду .NET. З SQL CLR, збережені процедури і тригери можуть бути записані в будь-якій керованій мові .NET, включаючи C # і VB.NET. Керований код також можна використовувати для визначення UDT (типів, визначених користувачем), які можуть зберігатися в базі даних. Керований код компілюється до збірок CLI і після перевірки на безпеку типу, зареєстрований у базі даних. Після цього їх можна викликати, як і будь-яку іншу процедуру. [30] Проте доступна лише підмножина бібліотеки базових класів, коли запускається код під SQL CLR. Більшість API, що стосуються функціональності інтерфейсу користувача, недоступні. [30]

При написанні коду для SQL CLR, дані, що зберігаються в базах даних SQL Server, можуть бути доступні за допомогою API ADO.NET, як і будь-яка інша керована програма, яка звертається до даних SQL Server. Однак, роблячи це, створюється нова сесія бази даних, відмінна від тієї, в якій виконується код. Щоб уникнути цього, SQL Server надає деякі удосконалення для постачальника ADO.NET, який дозволяє перенаправлення з'єднання на той самий сеанс, де вже розміщено запущений код. Такі підключення називаються контекстними зв'язками і встановлюються за допомогою параметра контекстного з'єднання значення true у рядку з'єднання. SQL Server також надає кілька інших удосконалень API ADO.NET, включаючи класи для роботи з табличними даними або єдиним рядком даних, а також класи для роботи з внутрішніми метаданими про дані, що зберігаються в базі даних. Він також надає доступ до функцій XML в SQL Server, включаючи підтримку XQuery. Ці вдосконалення також доступні в процедурах T-SQL внаслідок введення нового типу даних XML (функції запитів, значень, вузлів).

*Послуги*

SQL Server також включає асортимент додаткових послуг. Хоча вони не є необхідними для роботи системи баз даних, вони надають додаткові послуги на додаток до основної системи управління базами даних. Ці служби або виконуються як частина деяких компонентів SQL Server або поза процесом як служба Windows і представляють свій власний API для управління та взаємодії з ними.

*Послуги з машинного навчання*

Послуги SQL Server Machine Learning функціонують в екземплярі сервера SQL, дозволяючи людям здійснювати машинне навчання та аналітику даних без необхідності відправляти дані по мережі або обмежуватися пам'яттю власних комп'ютерів. Служби постачаються з дистрибутивами Microsoft R і Python, які містять найчастіше використовувані пакети для науки про дані, а також деякі патенти (наприклад, revoscalepy, RevoScaleR, microsoftml), які можна використовувати для створення моделей машин в масштабі.

Аналітики можуть або налаштувати клієнтську машину для підключення до віддаленого SQL-сервера, так і виконати виконання сценаріїв, або вони можуть запускати сценарії R або Python як зовнішній скрипт у запиті T-SQL. Підготовлену модель машинного навчання можна зберігати в базі даних і використовувати для оцінки.

*Сервісний брокер*

Використовується всередині екземпляра, середовище програмування. Для додатків, що перетинаються між собою, Service Broker здійснює зв'язок через TCP / IP і дозволяє синхронізувати різні компоненти через обмін повідомленнями. Сервісний брокер, який працює як частина механізму бази даних, забезпечує надійну платформу для обміну повідомленнями та повідомленнями для додатків SQL Server.

*Послуги реплікації*

Служби реплікації SQL Server використовуються SQL Server для реплікації і синхронізації об'єктів бази даних, або в повному обсязі, або в підмножині існуючих об'єктів, через агенти реплікації, які можуть бути іншими серверами баз даних у мережі, або кеші баз даних на стороні клієнта. Lulla дотримується моделі видавця / абонента, тобто зміни надсилаються одним сервером бази даних ("видавцем") і отримуються іншими ("абонентами").

SQL Server підтримує три різних типи реплікації:

1. *Реплікація транзакцій*

Кожна транзакція, зроблена до бази даних видавців (основна база даних), синхронізується з абонентами, які оновлюють свої бази даних транзакцією. Транзакційна реплікація синхронізує бази даних в реальному часі.

1. *Реплікація злиття*

Відслідковуються зміни, зроблені як у базі даних видавця, так і в абонентських базах даних, і періодично зміни синхронізуються двосторонньо між видавцем і абонентами. Якщо однакові дані були змінені по-різному в базах даних видавця і абонента, синхронізація призведе до конфлікту, який необхідно вирішити, або вручну, або за допомогою попередньо визначених політик. rowguid потрібно налаштувати на стовпці, якщо налаштовано тиражування злиття.

1. *Реплікація знімка*

Реплікація знімків публікує копію всієї бази даних (тодішній знімок даних) і реплікації для абонентів. Подальші зміни до знімка не відстежуються.

*Служби аналізу*

Служби аналізу SQL Server додають можливості OLAP і інтелектуального аналізу даних для баз даних SQL Server. Механізм OLAP підтримує режими зберігання MOLAP, ROLAP та HOLAP для даних. Служби Analysis Services підтримують стандарт XML for Analysis як базовий протокол зв'язку. Доступ до даних куба можна отримати за допомогою запитів MDX і LINQ. Спеціальні функціональні можливості інтелектуального аналізу даних розкриваються через мову запитів DMX. Аналіз послуг включає в себе різні алгоритми - дерева рішень, алгоритм кластеризації, алгоритм наївного байєсу, аналіз часових рядів, алгоритм кластеризації послідовностей, лінійний і логістичний регресійний аналіз і нейронні мережі - для використання в інтелектуальному аналізі даних.

*Служби звітності*

Служби звітування SQL Server - це середовище створення звітів для даних, зібраних з баз даних SQL Server. Вона керується через веб-інтерфейс. Служби звітності мають інтерфейс веб-служб для підтримки розробки спеціальних додатків звітів. Звіти створюються як файли RDL.

Звіти можуть бути розроблені з використанням останніх версій Microsoft Visual Studio (Visual Studio.NET 2003, 2005, 2008, 2011, 2013, 2015, 2017, 2019) за допомогою Business Intelligence Development Studio, встановленого або з включеним Report Builder. Після створення, файли RDL можуть відображатися в різних форматах, включаючи Excel, PDF, CSV, XML, BMP, EMF, GIF, JPEG, PNG і TIFF, і HTML Web Archive.

*Служби сповіщень*

Служби Notification Services були вперше і єдино вбудовані як платформа Microsoft SQL Server з SQL Server 2005. Служби сповіщень SQL Server - це механізм генерування сповіщень, керованих даними, які надсилаються абонентам служб Notification Services. Абонент реєструє конкретну подію або транзакцію (яка зареєстрована на сервері бази даних як тригер); коли відбувається подія, Служба Notification Services може використовувати один з трьох способів, щоб надіслати повідомлення абоненту, який повідомляє про виникнення події. Ці методи включають SMTP, SOAP або запису до файлу у файловій системі. Служба сповіщень була припинена корпорацією Майкрософт з випуском SQL Server 2008 у серпні 2008 року і більше не є офіційно підтримуваним компонентом платформи баз даних SQL Server.

*Інтеграційні послуги*

Служби інтеграції SQL Server (SSIS) надає можливості ETL для SQL Server для потреб імпорту даних, інтеграції даних і зберігання даних. Інтеграційні послуги включають засоби графічного інтерфейсу для побудови робочих процесів, таких як вилучення даних з різних джерел, запит даних, перетворення даних, включаючи агрегування, видалення дублікатів, де-нормалізацію і злиття даних, а потім експортують перетворені дані в цільові бази або файли.

*Cлужба пошуку повного тексту*

Архітектура служби повного текстового пошуку SQL Server

Служба повного текстового пошуку SQL Server - це спеціалізована служба індексування та запитів для неструктурованого тексту, що зберігається в базах даних SQL Server. Повнотекстовий індекс пошуку можна створити на будь-якому стовпчику з символьними текстовими даними. Це дозволяє шукати слова в текстових колонках. Хоча це може бути виконано з оператором SQL LIKE, використання служби SQL Server Full Text Search може бути більш ефективним. Повний дозволяє для неточного узгодження вихідного рядка, позначеного значенням рангу, який може варіюватися від 0 до 1000 - більш високий ранг означає більш точну відповідність. Він також дозволяє лінгвістичне зіставлення ("флективний пошук"), тобто мовні варіанти слова (наприклад, дієслово в іншому часі) також будуть відповідати даному слову (але з нижчим рейтингом, ніж точний збіг). Також підтримуються пошуки близькості, тобто, якщо шукані слова не відбуваються в послідовності, яку вони вказані в запиті, але знаходяться поруч один з одним, вони також вважаються відповіддю. T-SQL надає спеціальні оператори, які можуть використовуватися для доступу до можливостей FTS.

Повнотекстова пошукова система розділена на два процеси: процес фільтрування демонів (msftefd.exe) і процес пошуку (msftesql.exe). Ці процеси взаємодіють з SQL Server. Процес пошуку включає індексатор (який створює повні текстові індекси) і повний текстовий процесор запиту. Індексатор сканує текстові стовпці бази даних. Він також може індексувати через двійкові стовпці і використовувати iFilters для вилучення значущого тексту з двійкового блоба (наприклад, коли документ Microsoft Word зберігається як неструктурований двійковий файл у базі даних). IFilters розміщуються в процесі фільтра Демонів. Після вилучення тексту процес демону фільтрів розбиває його на послідовність слів і передає його індексатору. Індексатор відфільтровує шумові слова, тобто такі слова, як А, І т.д., які часто зустрічаються і не є корисними для пошуку. Що стосується інших слів, створюється інвертований індекс, що пов'язує кожне слово з знайденими стовпцями. Сам SQL Server містить компонент Gatherer, який контролює зміни в таблицях і викликає індексатор у разі оновлення.

Коли повний текстовий запит отримано процесором запитів SQL Server, він передається процесору запитів FTS в процесі пошуку. Процесор запитів FTS розбиває запит на складові слова, фільтрує шумові слова і використовує вбудований тезаурус, щоб дізнатися мовні варіанти для кожного слова. Потім слова запитуються щодо інвертованого індексу і обчислюється ранг їхньої точності. Результати повертаються клієнту через процес SQL Server.

*SQLCMD*

SQLCMD - це програма командного рядка, що поставляється з Microsoft SQL Server, і надає функції управління SQL Server. Це дозволяє запитам SQL виконуватися з командного рядка. Він також може виступати в якості мови сценаріїв для створення і виконання набору SQL-висловлювань як сценарію. Такі скрипти зберігаються як .sql-файл і використовуються або для управління базами даних, або для створення схеми бази даних під час розгортання бази даних.

SQLCMD був введений з SQL Server 2005, і це продовжується з SQL Server 2012, 2014 і 2016, 2018. Його попередник для більш ранніх версій був OSQL і ISQL, який функціонально еквівалентний, оскільки стосується виконання TSQL, і багато з параметрів командного рядка ідентичні, хоча SQLCMD додає додаткову універсальність.

*Visual Studio*

Microsoft Visual Studio містить вбудовану підтримку для програмування даних за допомогою Microsoft SQL Server. Він може бути використаний для запису та налагодження коду, який буде виконуватися SQL CLR. Він також включає конструктора даних, який можна використовувати для графічного створення, перегляду або редагування схем бази даних. Запити можна створювати візуально або за допомогою коду. SSMS 2008 року, також забезпечує intellisense для запитів SQL.

*Студія управління SQL Server*

SQL Server Management Studio - це інструмент графічного інтерфейсу, який входить до складу SQL Server 2005 і пізніших версій для налаштування, керування та адміністрування всіх компонентів у Microsoft SQL Server. Інструмент включає в себе як редактори сценаріїв, так і графічні інструменти, які працюють з об'єктами і можливостями сервера. SQL Server Management Studio замінює Enterprise Manager як основний інтерфейс керування для Microsoft SQL Server з SQL Server 2005. Версія SQL Server Management Studio також доступна для SQL Server Express Edition, для якої він відомий як SQL Server Management Studio Express (SSMSE).

Центральною особливістю SQL Server Management Studio є Object Explorer, який дозволяє користувачеві переглядати, вибирати і діяти на будь-який з об'єктів на сервері. Він може бути використаний для візуального спостереження та аналізу планів запитів і оптимізації продуктивності бази даних, серед інших. SQL Server Management Studio також може використовуватися для створення нової бази даних, зміни будь-якої існуючої схеми бази даних шляхом додавання або зміни таблиць і індексів або аналізу продуктивності. Вона включає в себе вікна запитів, які забезпечують інтерфейс на основі графічного інтерфейсу для запису і виконання запитів.

*Студія операцій SQL Server*

Студія операцій SQL Server (Попередній перегляд) - це редактор запитів для перехресних платформ, доступний як додаткове завантаження. Інструмент дозволяє користувачам писати запити; результати експорту запитів; виконувати SQL-скрипти до сховищ GIT і виконувати базову діагностику сервера. Операційна студія SQL Server підтримує системи Windows, Mac і Linux.

*Студія розвитку Business Intelligence*

Business Intelligence Development Studio (BIDS) - це IDE від Microsoft, що використовується для розробки аналізу даних і рішень Business Intelligence з використанням служб аналізу Microsoft SQL Server, служб звітності та служб інтеграції. Вона заснована на середовищі розробки Microsoft Visual Studio, але налаштована з розширеннями та типами проектів, специфічними для служб SQL Server, включаючи інструменти, елементи керування та проекти для звітів (з використанням служб Reporting Services), кубів та структур інтелектуального аналізу даних (за допомогою служб Analysis Services). Для SQL Server 2012 і пізніших версій цей IDE був перейменований в SQL Server Data Tools (SSDT).

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

Під час проходження практики на видавництві "Об’єднання ЮГ", було розглянуто процес створення електронних баз даних для вимог компанії. Також, протестували бази даних для галузі підприємства та розглянули типи завдань, які запропоновані, та їх якість взаємодії з користувачами.

Під час проведення екскурсії по підприємству, протестували також електронне обладнання, яке допомагають людям у різних сферах діяльності. Вони спрямовані на те, щоб у користувачів покращувались не тільки знання з різних сфер життєдіяльності, а й на те, щоб вони вчились зберігати свої дані від інших користувачів, що дуже необхідно у комп’ютерному світі.

Також виконали індивідуальні завдання, а саме: спроектували БД для потреб виробництва, враховуючи вимоги технічного завдання; реалізували спроектовану БД, застосовуючи відповідні технічні засоби, такі як MySQL, MS SQL, SQLite, Google Cloud; протестували БД в умовах, наближених до реальної роботи; імплементувати протестовану БД у виробничий процеc, за умови задовільного проходження тестів.