Видавнича система: Огляд та робота з MemSQL (SingleStore)

ЗМІСТ

---

1. Вступ

У сучасному світі обробка великих обсягів даних у реальному часі стає все більш актуальною задачею для бізнесу, науки, державних установ. Видавнича діяльність, як і багато інших галузей, потребує надійних, масштабованих та швидких систем управління базами даних (СУБД). Однією з таких сучасних платформ є MemSQL, яка з 2020 року відома під назвою SingleStore.

SingleStore — це розподілена реляційна база даних нового покоління, яка поєднує переваги класичних SQL-систем із можливостями горизонтального масштабування, in-memory обробки, підтримкою аналітичних і транзакційних навантажень. Вона дозволяє зберігати, обробляти та аналізувати великі масиви інформації з мінімальними затримками, що особливо важливо для видавничих систем, де дані про книги, авторів, редакторів, платформи, розділи та медіаматеріали постійно оновлюються та використовуються для прийняття рішень.

У цьому документі розглянуто основні принципи роботи MemSQL (SingleStore), її архітектуру, особливості, переваги, а також практичні аспекти встановлення, налаштування, створення та використання бази даних для видавничої системи. Окрему увагу приділено порівнянню з іншими популярними СУБД, питанням безпеки, масштабування, продуктивності та типових сценаріїв використання.

---

2. Історія та еволюція MemSQL (SingleStore)

2.1 Передумови створення

У другій половині 2000-х років IT-індустрія зіткнулася з новими викликами: обсяги даних зростали експоненційно, а класичні реляційні СУБД (Oracle, MySQL, PostgreSQL) не завжди справлялися з вимогами до швидкості обробки, масштабування та гнучкості. З'явилися NoSQL-рішення, але вони часто жертвували транзакційністю та стандартним SQL.

У цей час Нік Вервіл (Nik Verweil) та Ерік Франк (Eric Frenkiel), інженери Facebook, працювали над оптимізацією внутрішніх систем аналітики. Вони побачили, що сучасні процесори мають величезний обсяг оперативної пам'яті, яку можна використовувати для зберігання "гарячих" даних, забезпечуючи миттєвий доступ до них. Так виникла ідея створити реляційну базу даних, яка працює повністю у пам'яті — MemSQL.

2.2 Заснування та перші версії

У 2011 році Вервіл і Франк заснували компанію MemSQL у Сан-Франциско. Перші версії продукту були орієнтовані на in-memory зберігання: всі таблиці розміщувалися у RAM, що дозволяло виконувати SQL-запити з мілісекундними затримками. MemSQL підтримував стандартний SQL, транзакції, індекси, агрегати, що робило його знайомим для розробників, але обмежувало обсяг даних розміром оперативної пам'яті.

2.3 Еволюція функціоналу

Зі зростанням клієнтської бази та вимог до обсягу даних MemSQL почав розширювати функціонал:

- 2013–2015:
  Додано підтримку disk-based storage (columnstore), що дозволило зберігати великі обсяги даних на диску, а не лише у пам'яті.
  З'явилася можливість комбінувати rowstore (in-memory) та columnstore (disk) таблиці в одному кластері.
- 2015–2017:
  Впроваджено розподілену архітектуру: дані розбиваються на шард (shards), які розподіляються між Leaf Nodes. Додано реплікацію, fault-tolerance, автоматичне балансування навантаження.
- 2017–2019:
  З'явилася підтримка JSON, time-series, геоданих, window-функцій,

аналітичних запитів, інтеграція з Apache Kafka, Spark, BI-інструментами.

- 2019:
  MemSQL став доступним у хмарних середовищах (AWS, Azure, GCP), з'явилися офіційні Docker-образи, підтримка Kubernetes для DevOps.

2.4 Ребрендинг: перехід до SingleStore

У 2020 році компанія MemSQL провела масштабний ребрендинг.
Нова назва — SingleStore — підкреслює універсальність платформи: вона поєднує in-memory та disk storage, OLTP та OLAP, транзакції та аналітику, роботу з різними типами даних (структуровані, напівструктуровані, неструктуровані).
Причини ребрендингу:

- MemSQL асоціювався лише з in-memory, хоча платформа вже підтримувала disk storage.
- SingleStore позиціонується як "єдине сховище" для всіх типів даних і навантажень.
- Підкреслення можливості працювати з petabyte-обсягами, real-time аналітикою, гнучким масштабуванням.

2.5 Сучасний стан та застосування

Станом на 2024–2025 рік SingleStore — це одна з найшвидших розподілених реляційних СУБД, яка використовується у:

- Фінансових установах:
  Для обробки транзакцій, fraud detection, аналітики у реальному часі.
- Телекомунікаціях:
  Для моніторингу мережі, аналізу трафіку, управління обладнанням.
- E-commerce:
  Для аналітики покупок, рекомендаційних систем, обробки замовлень.
- IoT-платформах:
  Для збору, зберігання та аналізу сенсорних даних з мільйонів пристроїв.
- Геймінгу:
  Для зберігання ігрової статистики, лідербордів, аналітики у реальному часі.

- Аналітичних платформах:
  Для побудови дашбордів, звітів, BI, інтеграції з Tableau, PowerBI, Looker.

2.6 Віхи розвитку та нагороди

- 2014: MemSQL потрапив у топ-10 найінноваційніших стартапів за версією Forbes.
- 2016: Визнаний Gartner як “Cool Vendor” у категорії Data Management.
- 2018: MemSQL отримав нагороду “Best Big Data Solution” на Data Innovation Awards.
- 2020: Офіційний перехід до SingleStore, запуск SingleStore Studio, інтеграція з хмарними платформами.

2.7 Вплив на ринок СУБД

SingleStore став одним із лідерів у сегменті HTAP (Hybrid Transactional/Analytical Processing), тобто систем, які поєднують транзакційну та аналітичну обробку даних.

Платформа активно конкурує з такими рішеннями, як SAP HANA, Oracle Exadata, Google BigQuery, Amazon Redshift, але вирізняється простотою розгортання, гнучкістю, підтримкою стандартного SQL та високою продуктивністю.

2.8 Відкритість та розвиток спільноти

SingleStore підтримує відкриту документацію, має активний GitHub-репозиторій, співпрацює з університетами, проводить хакатони, конференції, вебінари.

Компанія постійно оновлює продукт, додає нові функції, оптимізує продуктивність, розширює інтеграції з сучасними інструментами аналітики та обробки даних.

---

3. Архітектура MemSQL (SingleStore)

Архітектура SingleStore побудована за принципом розподіленої системи, що дозволяє масштабувати базу даних горизонтально — тобто додавати нові сервери (вузли) для збільшення продуктивності та обсягу збережених даних.

Основні компоненти архітектури:

- Aggregator Nodes (Агрегатори):
  Вузли, які приймають SQL-запити від клієнтів, розподіляють їх між вузлами зберігання (Leaf Nodes), агрегують результати та повертають відповідь користувачу. Aggregator може бути Master (головний) або Child (дочірній).
- Leaf Nodes (Листи):
  Вузли, які безпосередньо зберігають дані та виконують обчислення над ними. Дані розподіляються між Leaf-вузлами за допомогою шардінгу (розбиття на частини).
- Distributed Storage:
  Дані зберігаються у вигляді шард (shards), кожен з яких може мати одну або кілька реплік для забезпечення відмовостійкості.
- Query Processing Engine:
  Оптимізатор запитів розподіляє виконання між вузлами, забезпечуючи паралельну обробку та мінімізацію часу відповіді.
- Management Tools:
  Веб-інтерфейс (SingleStore Studio), командний рядок, REST API для адміністрування, моніторингу, налаштування кластера.

Схематично архітектура SingleStore виглядає так:

[Клієнти]
    |
[Aggregator Nodes]
    |
[Leaf Nodes] <--- [Диски/Пам’ять]

Така архітектура дозволяє досягати високої продуктивності, масштабованості та відмовостійкості.

---

4. Особливості та переваги MemSQL (SingleStore)

4.1 In-memory та disk storage

Однією з головних інновацій SingleStore є поєднання in-memory та disk-based зберігання даних.

In-memory storage дозволяє зберігати таблиці у оперативній пам’яті, що забезпечує надшвидкий доступ та обробку даних. Це особливо корисно для аналітичних запитів, які потребують миттєвої відповіді, наприклад, у фінансових системах, онлайн-аналітиці, геймінгу.

Disk storage використовується для довготривалого зберігання великих обсягів інформації, архівів, історичних даних. SingleStore автоматично оптимізує розміщення даних, дозволяючи користувачу обирати тип зберігання для кожної таблиці (ROWSTORE — in-memory, COLUMNSTORE — disk).

Переваги:

- Можливість балансувати між швидкістю та обсягом даних.
- Зниження витрат на апаратне забезпечення.
- Гнучкість у виборі типу таблиці залежно від задачі.

4.2 Горизонтальне масштабування

SingleStore розроблена як розподілена система, що дозволяє додавати нові вузли у кластер без зупинки роботи.

Горизонтальне масштабування означає, що продуктивність та обсяг даних можна збільшувати простим додаванням серверів.

Технічні аспекти:

- Дані розбиваються на шард (shards), які розподіляються між Leaf Nodes.
- Запити виконуються паралельно на всіх вузлах, що забезпечує лінійне зростання продуктивності.
- Можливість автоматичного балансування навантаження.

Порівняння:

У класичних СУБД (PostgreSQL, MySQL) масштабування зазвичай вертикальне (додавання ресурсів одному серверу), а горизонтальне — складне і потребує додаткових рішень (шардінг, реплікація).

4.3 Висока продуктивність

SingleStore демонструє одну з найвищих швидкостей обробки SQL-запитів серед сучасних СУБД.

Причини:

- Використання in-memory обробки для “гарячих” даних.

- Паралельне виконання запитів на всіх вузлах.
- Оптимізований рушій SQL, який автоматично будує ефективний план виконання запиту.
- Підтримка columnstore-таблиць для аналітики (зберігання даних у колонках, а не рядках).

Реальні кейси:

У фінансових системах SingleStore дозволяє обробляти тисячі транзакцій за секунду, у IoT — аналізувати мільйони сенсорних подій у реальному часі.

4.4 Підтримка стандартного SQL

SingleStore повністю сумісна з SQL:2016, що дозволяє використовувати знайомі інструменти та синтаксис:

- DDL (CREATE, ALTER, DROP)
- DML (INSERT, UPDATE, DELETE)
- SELECT, JOIN, GROUP BY, HAVING, ORDER BY
- Вигляди (views), процедури, функції, індекси
- Підзапити, агрегати, CASE, WINDOW-функції

Переваги:

- Легкість міграції з інших SQL-систем.
- Можливість використання стандартних клієнтів (DBeaver, MySQL Workbench).
- Висока сумісність з BI та ETL-інструментами.

4.5 Транзакції та ACID

SingleStore гарантує повну підтримку ACID-принципів:

- Атомарність: всі операції у транзакції виконуються повністю або не виконуються взагалі.
- Узгодженість: база даних завжди переходить з одного коректного стану в інший.
- Ізольованість: паралельні транзакції не впливають одна на одну.
- Надійність: дані зберігаються навіть у разі збою системи.

Технічні аспекти:

- Підтримка BEGIN, COMMIT, ROLLBACK.
- Можливість виконувати багатотабличні транзакції.
- Журнали транзакцій для відновлення даних.

4.6 Підтримка JSON, геоданих, time-series

SingleStore дозволяє зберігати та обробляти складні структури даних:

- JSON: зберігання, пошук, фільтрація, індексація JSON-об’єктів.

- Геодані: робота з координатами, геометричними об’єктами, пошук у радіусі, побудова маршрутів.
- Time-series: оптимізоване зберігання та аналіз часових рядів, агрегування по часу, побудова графіків.

Приклад:

sql

```
CREATE TABLE SensorData (
    id INT AUTO_INCREMENT PRIMARY KEY,
    data JSON,
    timestamp DATETIME
);
SELECT * FROM SensorData WHERE JSON_EXTRACT(data,
'$.temperature') > 30;
```

4.7 Вбудовані засоби аналітики

SingleStore містить оптимізовані функції для аналітичних запитів:

- Columnstore-таблиці: ефективне зберігання та обробка великих обсягів даних для аналітики.
- Агрегати: SUM, AVG, COUNT, MIN, MAX, GROUP BY, HAVING.
- Вигляди (views): збереження складних запитів для швидкого доступу.
- Map-Reduce: розподілене виконання аналітичних операцій.

Переваги:

- Можливість використовувати SingleStore як OLAP-систему.
- Висока швидкість побудови звітів, дашбордів, аналітики.

4.8 Відмовостійкість та реплікація

SingleStore автоматично реплікує дані між вузлами:

- Primary Replica: основна копія даних.
- Secondary Replica: резервна копія для відмовостійкості.
- Автоматичне переключення: у разі збою Primary Replica система переходить на Secondary Replica без втрати даних.

Переваги:

- Висока доступність даних.
- Можливість оновлення та масштабування без простоїв.

4.9 Гнучкість розгортання

SingleStore підтримує різні варіанти розгортання:

- Локальні дата-центри: для корпоративних рішень.
- Хмарні сервіси: AWS, Azure, Google Cloud.

- Контейнери: Docker, Kubernetes для DevOps та CI/CD.
- Гібридні рішення: поєднання локальних та хмарних вузлів.

Переваги:

- Легкість масштабування та міграції.
- Можливість швидко розгорнути тестове або продакшн-середовище.

4.10 Інтеграція з BI та ETL-інструментами

SingleStore легко інтегрується з популярними інструментами для бізнес-аналітики та ETL:

- BI: Tableau, PowerBI, Qlik, Looker.
- ETL: Apache Kafka, Apache Spark, Talend, Informatica.
- Потокова обробка: підтримка real-time ingestion, CDC (Change Data Capture).

Переваги:

- Можливість будувати дашборди, звіти, аналітику у реальному часі.
- Легкість інтеграції у існуючу інфраструктуру.

---

5. Встановлення та налаштування MemSQL (SingleStore)
Встановлення SingleStore можливе кількома способами: через офіційний інсталятор, Docker-контейнер, хмарні сервіси або вручну на Linux/Windows.
5.1 Встановлення через Docker
Найшвидший спосіб розгорнути SingleStore для тестування — використати Docker-контейнер:
bash
```
docker run --platform linux/amd64 -d \
  --name singlestore \
  -e ROOT_PASSWORD=yourStrongPassword \
  -p 3307:3306 \
  -p 8080:8080 \
  singlestore/cluster-in-a-box:latest
```
Після запуску контейнера:

- Порт 3307 — для підключення через MySQL-клієнти (DBeaver, HeidiSQL, командний рядок).
- Порт 8080 — для веб-інтерфейсу SingleStore Studio.

5.2 Встановлення на Linux/Windows
Для розгортання у продакшн-середовищі рекомендується використовувати офіційний інсталятор або скрипти, які доступні на сайті SingleStore. Встановлення включає налаштування вузлів Aggregator та Leaf, конфігурацію мережі, безпеки, реплікації.
5.3 Налаштування кластеру
Після встановлення можна додавати нові вузли у кластер, налаштовувати шардінг, реплікацію, резервне копіювання, моніторинг. Всі ці операції виконуються через веб-інтерфейс або командний рядок.
5.4 Підключення до бази даних
Підключення здійснюється через стандартний MySQL-протокол, що дозволяє використовувати будь-які клієнти, які підтримують MySQL (DBeaver, MySQL Workbench, командний рядок).

---

6. Створення бази даних та таблиць

Після встановлення та підключення до SingleStore можна створити базу даних та необхідні таблиці для видавничої системи.

6.1 Створення бази даних

sql

```
CREATE DATABASE publishing_system;
USE publishing_system;
```

6.2 Створення таблиць

sql

```
CREATE TABLE Editor (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    role VARCHAR(100),
    manager_id INT
);

CREATE TABLE Author (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100)
);

CREATE TABLE Book (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    publication_date DATE,
    end_of_print_run DATE,
    editor_id INT,
    author_id INT
);

CREATE TABLE Platform (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);

CREATE TABLE Book_Platform (
```

```
    book_id INT,
    platform_id INT,
    budget DECIMAL(12,2),
    PRIMARY KEY (book_id, platform_id)
);

CREATE TABLE Chapter (
    id INT AUTO_INCREMENT,
    book_id INT,
    title VARCHAR(255) NOT NULL,
    genre VARCHAR(100),
    target_audience VARCHAR(100),
    language VARCHAR(50),
    PRIMARY KEY (id, book_id)
);

CREATE TABLE MediaMaterial (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    creation_date DATE,
    path VARCHAR(255)
);

CREATE TABLE AudioBook (
    id INT PRIMARY KEY,
    duration VARCHAR(50)
);

CREATE TABLE Illustration (
    id INT PRIMARY KEY,
    resolution VARCHAR(50)
);

CREATE TABLE Annotation (
    id INT AUTO_INCREMENT PRIMARY KEY,
    text TEXT
);
```

```
CREATE TABLE Chapter_MediaMaterial_
Annotation (
    chapter_id INT,
    media_material_id INT,
    annotation_id INT,
    PRIMARY KEY (chapter_id, media_material_id, annotation_id)
);
```

---

7. Заповнення даними
Після створення структури бази даних необхідно додати тестові або реальні дані для подальшої роботи.
sql
```
INSERT INTO Editor (name, role) VALUES
('Alice Smith', 'Chief Editor'),
('Bob Johnson', 'Editor'),
('Carol White', 'Senior Editor');

INSERT INTO Author (name, email) VALUES
('John Doe', 'john@example.com'),
('Jane Roe', 'jane@example.com');

INSERT INTO Platform (name) VALUES
('Amazon Kindle'),
('Google Books'),
('Apple Books');

INSERT INTO Book (title, publication_date, end_of_print_run, editor_id, author_id) VALUES
('The Great Adventure', '2023-01-15', NULL, 1, 1),
('Mystery of the Night', '2024-03-10', NULL, 2, 2);

INSERT INTO Book_Platform (book_id, platform_id, budget) VALUES
(1, 1, 5000.00),
(1, 2, 3000.00),
(2, 2, 4000.00),
(2, 3, 2500.00);

INSERT INTO Chapter (book_id, title, genre, target_audience, language) VALUES
(1, 'Prologue', 'Adventure', 'Adults', 'English'),
(1, 'Into the Wild', 'Adventure', 'Adults', 'English'),
(2, 'The Shadow', 'Mystery', 'Teens', 'English');

INSERT INTO MediaMaterial (name, creation_date, path) VALUES
('Cover Art', '2023-01-01', '/media/cover1.jpg'),
```

('Audio Sample', '2023-01-05', '/media/audio1.mp3');

INSERT INTO AudioBook (id, duration) VALUES
(2, '01:20:00');

INSERT INTO Illustration (id, resolution) VALUES
(1, '1920x1080');

INSERT INTO Annotation (text) VALUES
('Introductory note'),
('Chapter 1 summary');

INSERT INTO Chapter_MediaMaterial_Annotation (chapter_id,
media_material_id, annotation_id) VALUES
(1, 1, 1),
(2, 2, 2);

---

8. Запити та аналітика
SingleStore підтримує стандартні SQL-запити для вибірки, агрегації, обробки даних. Ось приклади основних запитів для видавничої системи:
8.1 Вибірка книг з автором та редактором
sql
```
SELECT b.title, a.name AS author, e.name AS editor, b.publication_date
FROM Book b
LEFT JOIN Author a ON b.author_id = a.id
LEFT JOIN Editor e ON b.editor_id = e.id;
```
8.2 Книги на платформах з бюджетом > 3000
sql
```
SELECT b.title, p.name AS platform, bp.budget
FROM Book b
JOIN Book_Platform bp ON b.id = bp.book_id
JOIN Platform p ON bp.platform_id = p.id
WHERE bp.budget > 3000;
```
8.3 Всі розділи певної книги
sql
```
SELECT c.id AS chapter_id, c.title AS chapter_title, c.genre, c.target_audience, c.language, b.title AS book_title
FROM Chapter c
JOIN Book b ON c.book_id = b.id
WHERE b.id = 1;
```
8.4 Всі медіаматеріали типу аудіо
sql
```
SELECT m.id, m.name, m.creation_date, m.path, a.duration
FROM MediaMaterial m
JOIN AudioBook a ON m.id = a.id;
```
8.5 Збережені запити (views)
sql
```
CREATE VIEW books_with_authors_editors AS
SELECT b.title, a.name AS author, e.name AS editor, b.publication_date
FROM Book b
LEFT JOIN Author a ON b.author_id = a.id
LEFT JOIN Editor e ON b.editor_id = e.id;
```

9. Масштабування та реплікація

Однією з ключових переваг SingleStore є можливість горизонтального масштабування та забезпечення високої доступності даних завдяки реплікації.

9.1 Горизонтальне масштабування

SingleStore дозволяє додавати нові вузли (сервери) у кластер без зупинки роботи системи. Дані автоматично розподіляються між Leaf Nodes за допомогою шардінгу. Кожен шард містить частину даних, і запити до бази виконуються паралельно на всіх вузлах, що значно підвищує продуктивність.

Масштабування здійснюється через:

- Додавання Leaf Nodes: для збільшення обсягу збережених даних та пропускної здатності.
- Додавання Aggregator Nodes: для балансування навантаження запитів.

9.2 Реплікація та відмовостійкість

Для забезпечення надійності та збереження даних SingleStore підтримує реплікацію шард:

- Primary Replica: основна копія даних, на яку записуються всі зміни.
- Secondary Replica: резервна копія, яка використовується у разі збою Primary Replica.

У разі виходу з ладу одного з вузлів, система автоматично переключається на резервну копію, забезпечуючи безперервну роботу додатків.

9.3 Бекапи та відновлення

SingleStore підтримує створення резервних копій (backup) бази даних, які можна зберігати локально або у хмарі. Відновлення даних здійснюється через спеціальні команди або інтерфейс адміністрування.

---

10. Безпека

Безпека даних — один із найважливіших аспектів сучасних СУБД. SingleStore реалізує комплексний підхід до захисту інформації.

10.1 Аутентифікація та авторизація

- Користувачі та ролі:
  Система дозволяє створювати користувачів з різними правами доступу, призначати ролі, обмежувати доступ до окремих баз, таблиць, операцій.
- Паролі та політики безпеки:
  Підтримується політика складних паролів, регулярна зміна паролів, блокування користувачів.

10.2 Шифрування даних

- Шифрування на рівні диска:
  Дані можуть зберігатися у зашифрованому вигляді, що захищає їх від несанкціонованого доступу у разі фізичного доступу до серверів.
- Шифрування трафіку:
  Підтримується SSL/TLS для захисту даних при передачі між клієнтом і сервером.

10.3 Бекапи та аудит

- Резервне копіювання:
  Регулярне створення бекапів дозволяє відновити дані у разі втрати або пошкодження.
- Аудит дій користувачів:
  Всі операції над даними можуть логуватися для подальшого аналізу та розслідування інцидентів.

---

11. Порівняння з іншими СУБД

SingleStore поєднує переваги класичних реляційних баз даних та сучасних розподілених платформ. Ось коротке порівняння з іншими популярними СУБД:

Вимір	SingleStore	PostgreSQL	MySQL	MongoDB
SQL	Повна підтримка	Повна підтримка	Повна підтримка	Обмежена
Масштабування	Горизонтальне	Вертикальне/шардінг	Вертикальне	Горизонтальне
ACID	Так	Так	Так	Частково
JSON	Так	Так (JSONB)	Так	Так
In-memory	Так	Ні	Ні	Ні
Реплікація	Так	Так	Так	Так
OLAP/OLTP	OLAP+OLTP	OLTP+OLAP (через розширення)	OLTP	OLTP/OLAP (через aggregation)

SingleStore виділяється завдяки поєднанню in-memory обробки, горизонтального масштабування, високої продуктивності та підтримки стандартного SQL.

---

12. Практичне застосування

SingleStore використовується у багатьох сферах, де потрібна швидка обробка великих обсягів даних:

- Фінансові системи:
  Обробка транзакцій у реальному часі, аналітика, fraud detection.
- IoT-платформи:
  Збір, зберігання та аналіз сенсорних даних з мільйонів пристроїв.
- E-commerce:
  Аналітика покупок, рекомендаційні системи, обробка замовлень.
- Телекомунікації:
  Моніторинг мережі, аналіз трафіку, управління обладнанням.
- Геймінг:
  Зберігання та аналіз ігрової статистики, лідерборди, реальний час.

У видавничих системах SingleStore дозволяє:

- Зберігати інформацію про книги, авторів, редакторів, платформи, розділи, медіаматеріали.
- Швидко виконувати пошук, фільтрацію, аналітику по книгах, платформах, жанрах.
- Масштабувати систему при зростанні кількості користувачів та обсягу даних.

---

13. Тестування продуктивності

Продуктивність SingleStore оцінюється за допомогою стандартних бенчмарків (TPC-H, TPC-C), а також власних тестів на реальних даних.

13.1 Бенчмарки

- TPC-H:
  Тестує аналітичні запити, агрегати, JOIN, GROUP BY.
- TPC-C:
  Оцінює транзакційну продуктивність, швидкість запису та оновлення.

13.2 Оптимізація запитів

- Використання індексів для прискорення пошуку.
- Збережені запити (views) для часто використовуваних вибірок.
- Паралельне виконання запитів на всіх вузлах кластера.

13.3 Результати

SingleStore демонструє високу швидкість обробки як аналітичних, так і транзакційних запитів, особливо при великій кількості даних та користувачів.

---

14. Проблеми та їх вирішення

14.1 Типові проблеми

- Помилки при створенні таблиць:
  У Community Edition не підтримуються FOREIGN KEY, тому зв’язки реалізуються через поля з id.
- Виконання масових INSERT:
  Рекомендується використовувати режим "Execute SQL Script" у клієнтах типу DBeaver.
- Відновлення після збою:
  Регулярне створення бекапів та налаштування реплікації.

14.2 Troubleshooting

- Оптимізація запитів:
  Аналіз плану виконання, додавання індексів, використання views.
- Моніторинг системи:
  Використання SingleStore Studio для перегляду стану вузлів, навантаження, журналів.
- Відновлення даних:
  Використання резервних копій, автоматичне переключення на резервні репліки.

---

15. Висновки

SingleStore — це сучасна розподілена реляційна база даних, яка поєднує переваги in-memory обробки, горизонтального масштабування, високої продуктивності та підтримки стандартного SQL. Вона ідеально підходить для систем, де потрібна швидка обробка великих обсягів даних у реальному часі, збереження надійності, безпеки та гнучкості.

У видавничих системах SingleStore дозволяє ефективно управляти інформацією про книги, авторів, редакторів, платформи, розділи, медіаматеріали, забезпечує швидкий пошук, аналітику, масштабування та інтеграцію з сучасними інструментами.