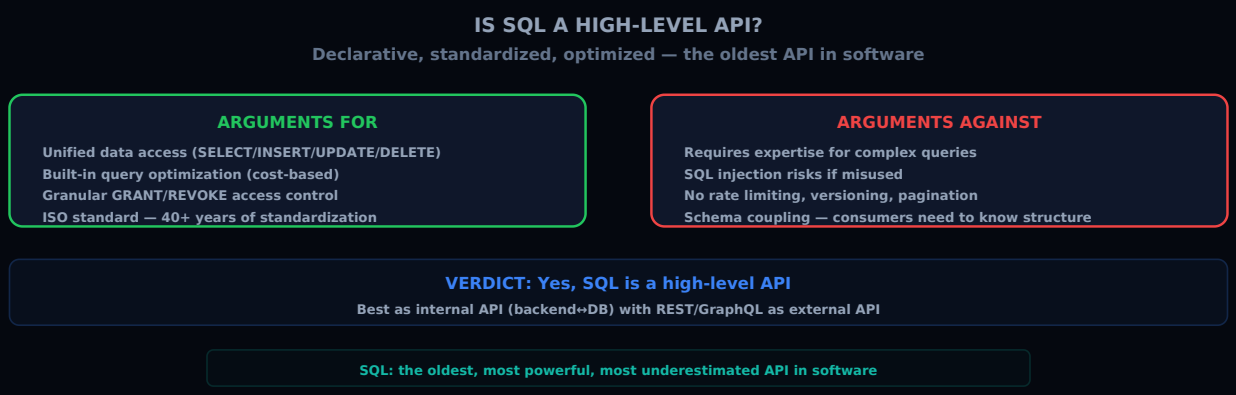
Является ли SQL высокоуровневым API?

Sylvain ARBAUDIE · May 18, 2025

SQL ARCHITECTURE API OPINION



Неудобный вопрос

В современной архитектуре программного обеспечения всё проходит через API. REST, GraphQL, gRPC — приложения общаются через чётко определённые, документированные, версионированные интерфейсы. Но существует API, который предшествует всем остальным, вездесущ, и который никто по-настоящему не называет «API»: **SQL**.

SQL — стандартизированный декларативный язык (ISO/IEC 9075), позволяющий взаимодействовать со структурированными данными. Вы объявляете, чего хотите, а не как это получить. Движок базы данных заботится о выполнении. Это, по определению, программный интерфейс — API.

Так почему же никто не воспринимает SQL как API?

SQL как API: аргументы за

Единый доступ к данным

SQL предлагает единую точку доступа к сложным данным. Будь то чтение одной строки, соединение десяти таблиц, агрегация миллионов строк или массовое изменение данных, интерфейс один и тот же: SQL-инструкции, отправляемые по сетевому протоколу (протокол MariaDB / MySQL, например).

Это именно то, что делает REST API: предоставляет единую точку доступа к ресурсам с ясной семантикой (GET = читать, POST = создать и т.д.). SQL делает то же самое с SELECT, INSERT, UPDATE, DELETE.

Встроенная оптимизация

Когда вы вызываете REST API, ваш бэкенд-код решает, как выполнить запрос. Когда вы отправляете SQL-запрос, оптимизатор базы данных выбирает лучший план выполнения.

SQL-оптимизатор анализирует запрос, оценивает статистику таблиц, рассматривает доступные индексы и генерирует оптимальный план выполнения. Это слой абстракции, который мало какие API предлагают: вы говорите «что», система решает «как».

Гранулярный контроль доступа

SQL интегрирует нативную систему контроля доступа. GRANT и REVOKE позволяют тонко контролировать, кто что может делать с какими данными. Это эквивалент системы авторизации, встроенной в API.

```
GRANT SELECT (product_name, price) ON catalog.products TO 'api_user'@'%';
```

Этот пользователь может читать название и цену продуктов, но не внутренние затраты и не маржу. Контроль доступа — на уровне столбца.

Стандартизация

SQL — стандарт ISO. Несмотря на различия между реализациями (MariaDB, MySQL, PostgreSQL, Oracle), ядро языка общее. Разработчик, знающий SQL, может взаимодействовать с любой реляционной базой данных.

Это преимущество, которым обладают немногие API. REST — не строгий стандарт (это архитектурный стиль), GraphQL — более молодой стандарт, gRPC привязан к Protocol Buffers. SQL имеет более 40 лет стандартизации.

SQL как API: аргументы против

Требуемая экспертиза

SQL мощный, но сложный. Написать простой SELECT доступно любому разработчику. Написать производительный запрос со сложными соединениями, рекурсивными CTE и оконными функциями требует значительной экспертизы.

Хороший REST/GraphQL API абстрагирует эту сложность: потребителю не нужно знать, как данные физически организованы. С SQL потребитель должен понимать схему, связи и ограничения производительности.

SQL-инъекции

Риск SQL-инъекций — ахиллесова пята SQL-как-API. Если потребитель строит запросы конкатенацией строк, данные в опасности.

```
# ОПАСНО — возможна SQL-инъекция
query = f"SELECT * FROM users WHERE name = '{user_input}'"

# БЕЗОПАСНО — параметризованный запрос
query = "SELECT * FROM users WHERE name = %s"
cursor.execute(query, (user_input,))
```

REST/GraphQL API не имеют этой фундаментальной проблемы: интерфейс отделён от данных по конструкции.

Нет управления соединениями

SQL не управляет понятием HTTP-сессии, rate limiting, стандартизированной пагинации и версионирования API. Протокол MariaDB / MySQL управляет соединениями, но нет эквивалента HTTP-заголовков, кодов статуса 4xx/5xx или механизмов HTTP-кэширования.

Эти функции должны быть реализованы поверх SQL, через прокси (MaxScale, ProxySQL) или прикладные слои.

Привязка к схеме

Прямое предоставление SQL привязывает потребителя к физической схеме базы данных. Если вы переименуете столбец, добавите таблицу или измените связь, все SQL-запросы потребителя должны быть обновлены. REST или GraphQL API изолирует потребителя от этих изменений через слой абстракции.

Вердикт: да, но...

SQL функционально является высокоуровневым API. Он отвечает основным критериям: стандартизированный интерфейс, декларативный доступ к данным, встроенная оптимизация, контроль доступа.

Но это API, требующий экспертизы для корректного использования, несущий риски безопасности при неправильном применении и не предоставляющий современных механизмов управления (версионирование, rate limiting, пагинация).

Лучший подход, вероятно, гибридный:

- **SQL как внутренний API** между бэкенд-сервисами и базой данных, с представлениями и хранимыми процедурами как слоем абстракции
- **REST/GraphQL как внешний API** для потребителей, которым не нужно знать физическую схему

SQL не мёртв. SQL не инструмент прошлого. Это API — самый древний, самый мощный и самый недооценённый в программной экосистеме.

Эта статья была первоначально опубликована на Medium.