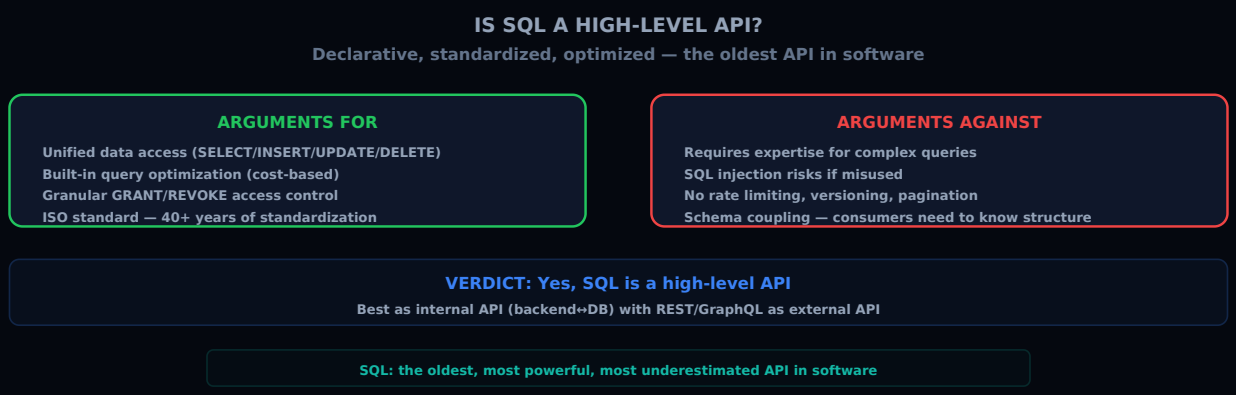
Czy SQL to API wysokiego poziomu?

Sylvain ARBAUDIE · May 18, 2025

SQL ARCHITECTURE API OPINION



Pytanie, które przeszkadza

We współczesnej architekturze oprogramowania wszystko przechodzi przez API. REST, GraphQL, gRPC — aplikacje komunikują się przez dobrze zdefiniowane, udokumentowane, wersjonowane interfejsy. Ale istnieje API, które poprzedza wszystkie inne, jest wszechobecne i którego nikt tak naprawdę nie nazywa "API": **SQL**.

SQL to standaryzowany język deklaratywny (ISO/IEC 9075) umożliwiający interakcję z danymi strukturalnymi. Deklarujesz, czego chcesz, a nie jak to uzyskać. Silnik bazy danych zajmuje się wykonaniem. To z definicji interfejs programistyczny — API.

Dlaczego więc nikt nie traktuje SQL jak API?

SQL jako API: argumenty za

Zunifikowany dostęp do danych

SQL oferuje pojedynczy punkt dostępu do złożonych danych. Niezależnie od tego, czy chodzi o odczyt jednego wiersza, złączenie dziesięciu tabel, agregację milionów wierszy czy masową modyfikację danych, interfejs jest ten sam: instrukcje SQL wysyłane przez protokół sieciowy (np. protokół MariaDB / MySQL).

To dokładnie to, co robi API REST: wystawia zunifikowany punkt dostępu do zasobów z jasną semantyką (GET = odczyt, POST = tworzenie itp.). SQL robi to samo z SELECT, INSERT, UPDATE, DELETE.

Wbudowana optymalizacja

Gdy wywołujesz API REST, to Twój kod backendowy decyduje, jak wykonać żądanie. Gdy wysyłasz zapytanie SQL, to optymalizator bazy danych wybiera najlepszy plan wykonania.

Optymalizator SQL analizuje zapytanie, ocenia statystyki tabel, rozważa dostępne indeksy i generuje optymalny plan wykonania. To warstwa abstrakcji, którą niewiele API oferuje: mówisz "co", system decyduje "jak".

Granularna kontrola dostępu

SQL integruje natywny system kontroli dostępu. GRANT i REVOKE pozwalają precyzyjnie kontrolować, kto co może robić na jakich danych. To odpowiednik systemu autoryzacji wbudowanego w API.

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

Ten użytkownik może czytać nazwę i cenę produktów, ale nie koszty wewnętrzne ani marże. Kontrola dostępu jest na poziomie kolumny.

Standaryzacja

SQL to standard ISO. Pomimo różnic między implementacjami (MariaDB, MySQL, PostgreSQL, Oracle), rdzeń języka jest wspólny. Deweloper znający SQL może interagować z dowolną relacyjną bazą danych.

To zaleta, którą niewiele API posiada. REST nie jest ścisłym standardem (to styl architektoniczny), GraphQL to nowszy standard, gRPC jest powiązany z Protocol Buffers. SQL ma ponad 40 lat standaryzacji.

SQL jako API: argumenty przeciw

Wymagana ekspertyza

SQL jest potężny, ale złożony. Napisanie prostego SELECT jest dostępne dla każdego dewelopera. Napisanie wydajnego zapytania ze złożonymi złączeniami, rekurencyjnymi CTE i funkcjami okienkowymi wymaga znaczącej ekspertyzy.

Dobre API REST/GraphQL abstrahuje tę złożoność: konsument nie musi wiedzieć, jak dane są fizycznie zorganizowane. Z SQL konsument musi rozumieć schemat, relacje i ograniczenia wydajnościowe.

Iniekcja SQL

Ryzyko iniekcji SQL to pięta achillesowa SQL-jako-API. Jeśli konsument buduje zapytania przez konkatenację ciągów, dane są zagrożone.

```
# NIEBEZPIECZNE – możliwa iniekcja SQL
query = f"SELECT * FROM users WHERE name = '{user_input}'"

# BEZPIECZNE – zapytanie parametryzowane
query = "SELECT * FROM users WHERE name = %s"
cursor.execute(query, (user_input,))
```

API REST/GraphQL nie mają tego fundamentalnego problemu: interfejs jest oddzielony od danych z założenia.

Brak zarządzania połączeniami

SQL nie zarządza pojęciem sesji HTTP, rate limitingu, standardowej paginacji ani wersjonowania API. Protokół MariaDB / MySQL zarządza połączeniami, ale nie ma odpowiednika nagłówków HTTP, kodów statusu 4xx/5xx ani mechanizmów cache HTTP.

Te funkcjonalności muszą być implementowane ponad SQL, przez proxy (MaxScale, ProxySQL) lub warstwy aplikacyjne.

Sprzężenie ze schematem

Bezpośrednie wystawienie SQL sprzęga konsumenta z fizycznym schematem bazy. Jeśli zmienisz nazwę kolumny, dodasz tabelę lub zmodyfikujesz relację, wszystkie zapytania SQL konsumenta muszą być zaktualizowane. API REST lub GraphQL izoluje konsumenta od tych zmian przez warstwę abstrakcji.

Werdykt: tak, ale...

SQL jest funkcjonalnie API wysokiego poziomu. Spełnia kluczowe kryteria: standaryzowany interfejs, deklaratywny dostęp do danych, wbudowana optymalizacja, kontrola dostępu.

Ale to API wymagające ekspertyzy do prawidłowego użycia, wystawiające ryzyka bezpieczeństwa przy złym użyciu i nieudostępniające nowoczesnych mechanizmów zarządzania (wersjonowanie, rate limiting, paginacja).

Najlepsze podejście jest prawdopodobnie hybrydowe:

- **SQL jako wewnętrzne API** między usługami backendowymi a bazą danych, z widokami i procedurami składowanymi jako warstwa abstrakcji
- **REST/GraphQL jako zewnętrzne API** dla konsumentów, którzy nie muszą znać fizycznego schematu

SQL nie umarł. SQL nie jest narzędziem przeszłości. To API — najstarsze, najpotężniejsze i najbardziej niedoceniane w ekosystemie oprogramowania.

Ten artykuł został pierwotnie opublikowany na Medium.