International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246/ICRTCSIT-118
Eureka Vision Publication | ICRTCSIT'25-Conference Proceeding

Original Article

Data Products as APIs: Building Self-Service Access to Trusted Financial Data with GraphQL

Saikrishna Garlapati
Sr. Software Engineer, Independent Researcher, USA.

Abstract - Today's financial industry needs to secure, flexible, reliable and immediate access to vast quantities of intricate information. Exposing data products as APIs is a powerful methodology to foster self-service access to data across inter- and intra- organizational perimeters. This paper discusses the architecture, security and governance practices, schema design, performance management and operations of financial data products as GraphQL APIs. The design choices and the implementation processes are supported by real-life examples, describing their advantages and their drawbacks. Future prospects are outlined in the fields of AI-based query understanding and optimization, privacy-preserving computations and other innovative practices that will determine the architectural blueprint of next-generation API-based financial data products.

Keywords - Financial APIs, Data products, GraphQL, Self-service data access, Data governance, API security, Financial services architecture.

1. Introduction

Financial services are at the forefront of creating ever-growing and ever-more complex data sets at phenomenal scale. Data is poured from transactional systems, trading venues, customer touchpoints, compliance filings and reports, and innumerable external third-party data sources and providers. To derive optimal use of data for risk assessment, regulatory compliance, creation of new products, and optimized opportunities for customer engagement, optimized use of data must be derived. Conversely, traditional data architectures involving ETL jobs, monolithic data warehouses, and adjunct data reporting processes have latency and complexity that reduce opportunistic behavior and active data availability access capabilities. Through APIs, data products can be abstracted, where managed data products contain bounded data sets. These data sets are served through standardization interfaces that have a high degree of reliability. Using this abstraction layer, data access can be accomplished to internal lines of business or even external partners or innovative fintech disruptors. The self-serving features, therefore, allow data access to interface directly with the data while reducing latency and increasing trust in the process. While the more standard REST APIs are functional for many applications, they can cause over-fetching, where data is retrieved in excess. They may also impose rigid integration methodologies that hinder process agility and adaptability, which can exacerbate the situation instead of simplifying it.do sba loa

A newly introduced API query language, GraphQL (developed by Facebook), proposes a valuable alternative for the future of data querying. In contrast with earlier techniques, it allows the client to ask for the exact data they need from a single query, which, in comparison with conventional REST API (application program interface) that usually fetch more data than required for getting particular data, highly increases the overall operating efficiency. The developer's productivity benefits significantly from quick and precise data gathering. Further, it reduces the overall payload size, yielding advantages for network performance. Also, due to the usage of a common single endpoint and type-safe schemas, it facilitates API introspection (the process of discovering information about a particular API), which improves API documentation (adding better, more detailed documentation considering the requirements) and considerably assists the "evolve-ability" of API in time while the requirements have changed. Such introspection enables the developer to better "explore" the API, allowing for more solid and adaptive design. This paper proposes a thorough architecture for creating reliable and elastic financial data products as GraphQL APIs that enable secure and compliant self-service access. It goes on to detail the architecture decisions, schema design, security rules and operational procedures with the help of case studies from several companies.

2. Background and Related Work

2.1. The API Economy and Financial Services

Economic partners intend to use APIs to evolve the bank-related ecosystem towards an API economy, in which modular and reusable pieces serve as the essential components for exchanging data and providing services. Open Banking-related regulations, including the Payment Service Directive 2 in Europe, require banks and financial service providers to share access to their clients' financial transaction data through publicly accessible Application Programming Interfaces. This change aims to promote

competitiveness in the financial services markets. However, banks can go beyond regulatory requirements and use the Open Banking ecosystem to integrate their services into other mechanisms and platforms not directly related to traditional banking.

2.2. Data Productization and Self-Service Movement

Data productization achieves this by adding useful metadata to datasets, applying rigorous quality checks, and delivering intuitive interfaces that ensure datasets are operational as purchasable digital goods. Using machine learning and plasticity methods, sized data allows application-oriented solution configurations. Self-service data platforms provide domain experts, business analysts, and developers with access to datasets and the ability to query, investigate, and integrate with data without routing through centralized IT services. Independence reduces decision time by relieving stakeholders of the need to contact IT programs for requests and significantly shortening the waiting period for reports from other departments. This reduces workload pressure on IT teams, allowing dedicated IT staff more time for more substantive commands.

3. GraphQL Fundamentals

GraphQL is a dynamic, versatile, and strongly typed query language and runtime that provides an interface to retrieve and upload data across various backends. Its single schema facilitates optimization and aggregation, yielding improved results. As a client-driven language, GraphQL prioritizes the client's desired data representation resulting in superior network efficiency against REST. Features like Introspection encourage dynamic app exploration and tooling.

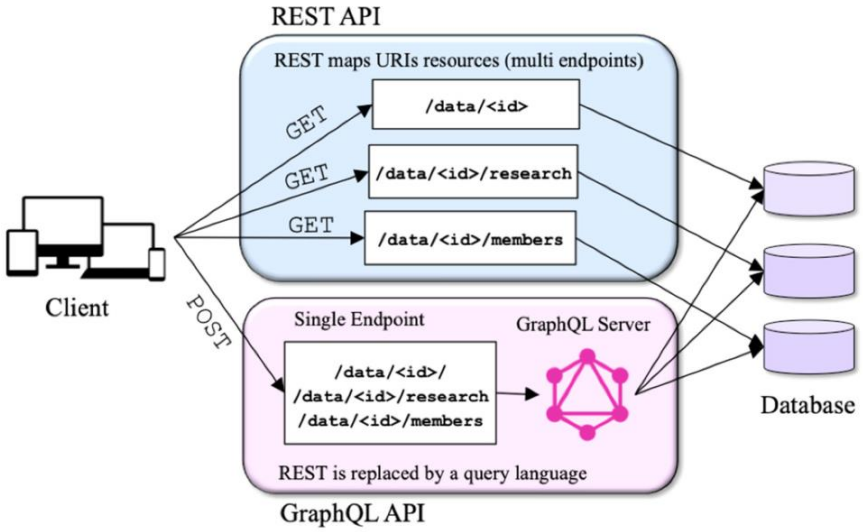

Figure 1. Difference between REST and GraphQL architectures

3.1. Challenges in Financial Data Access

First of all, financial data is subject to high standards of security, privacy legislation, and auditing standards that make sharing difficult, Second, data heterogeneity, quality, lineage, and scalability of access create further challenges to delivering trusted data at scale.

4. Architecture of Financial Data Products as APIs

4.1. Architectural Overview

For financial data products as APIs, the end-to-end solution encompasses multiple layers:

4.1.1. Data Ingestion and Processing Layer

Various core systems (transaction processing systems, payments systems, trading and settlements platforms) and external feeds (financial markets feeds) are streamed/ batch pipelined to ingest raw data. The pipelines pull data, cleanse, normalize, enrich, and aggregate the data and provide ready-to-consume data.

4.1.2. Data Storage Layer

Finally after validation, the data is stored and curated in scalable data stores such as data warehouses, data lakes or lakehouses. These systems guarantees and provide data versioning and query mechanisms that is optimized for analytics workloads.

4.1.3. GraphQL as API Layer

The GraphQL API Server implements a federated, unified schema, representing multiple data sources. It implements resolvers, that optimally convert GraphQL queries to calls to backends or databases aggregating responses.

4.1.4 .Metadata and Content Layer

Augmented content enrichment processes create an additional dimension of value for owners and users alike. These processes vest further meaning into the data paradigm, utilizing context and concepts that may vary by user, business unit, or partner. High relevance is achieved through automated concepts and classifications that go beyond initial collection in documents, collections or database structures. Policies for automated data disguising adjust this additional meaning when data is used outside the initial one.

4.1.5. Monitoring and Analytics

Enables visibility and control in compliance with real-time monitoring of API trafficking, performance metrics, schema drift and security violations.

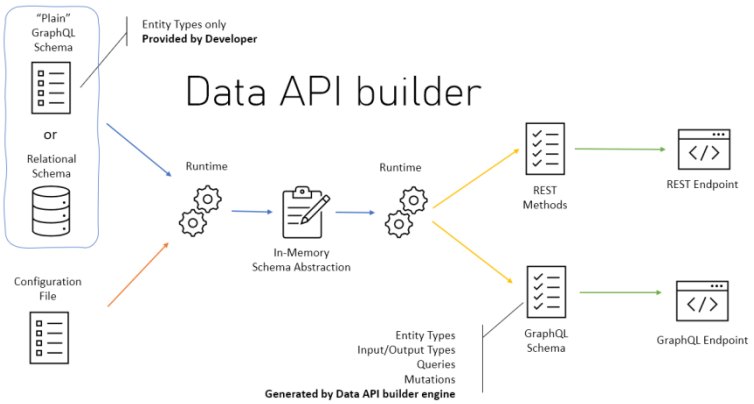

Figure 2. Architecture of Data APIs Builder

4.2. Advanced GraphQL Schema Design

GraphQL schemas model financial domain entities and their relationships using rich types and directives:

- Object Types: Define core entities like Account, Transaction, Portfolio, CreditScore.
- Interfaces and Unions: Handle polymorphic financial products (e.g., different investment instruments).
- Custom Scalars: Encode domain-specific types such as currencies, date/time formats, monetary amounts.
- Input Types and Mutations: Enable secure and validated updates to financial data.
- Directives: Annotate schema fields for nullable, deprecated, or permission-based visibility.

Schema design balances expressiveness with simplicity and leverages schema federation to compose large API surfaces without monolithic schemas.

4.3. API Federation and Composability

Big financial organizations operate several heterogenous systems that are being implemented over several years. With GraphQL Federation, it is possible to compose several GraphQL services that are managed by different teams, where each team can keep the independence and yet expose a single API surface. Such approach allows:

- Independent evolution and deployment of services.
- Seamless integration of internal and external APIs.
- Dynamic resolution of queries spanning multiple domains (e.g., customer data joined with market data).

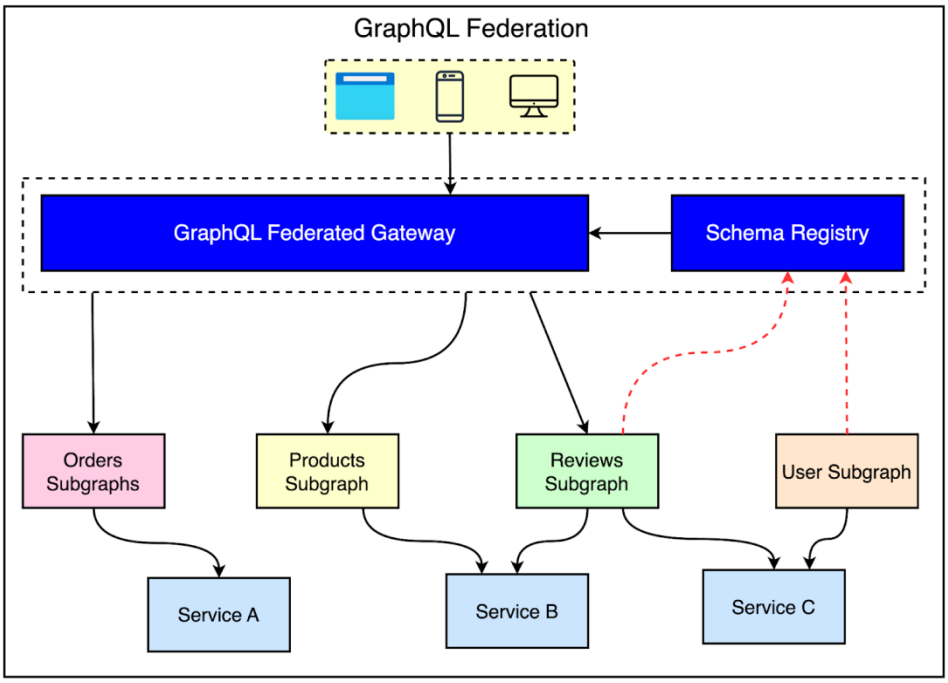

Figure 3. Federated GraphQL Architecture for Multi-Domain Financial Systems

5. Implementation Considerations

5.1. Trust and Data Quality

These measures would be complemented by metadata catalogs for data lineage, schema versions, and quality metrics; automated validation pipelines to preemptively discover anomalies and enforce correctness before data is exposed; change management processes for the safe evolution of a schema with backward compatibility; and analytics on data usage.

5.2. Security and Compliance

OAuth 2.0 and OpenID Connect for federated identity management with fine-grained RBAC and Attribute-Based Access Control (ABAC) to provide field-level and row-level access controls. Data masking, encryption in transit and encryption at rest for sensitive financial and personal data. Immutable auditing with logging of query used for regulatory review and compliance. Continuous monitoring for anomalous patterns or misuse with integrated SIEM (security information and event management).

5.3. Performance Optimization and Scalability

Batch and cache database access by using data loader patterns. Analyze query cost and depth to defend against overuse of queries and backend systems. Use result caching and persisted queries to optimize hot access patterns. Create APIs and infrastructure for horizontal scaling and failover to support peak use and fault tolerance.

5.4. Developer Enablement and Self-Service

Develop interactive API explorers (e.g. GraphiQL, Apollo Studio) for exploratory querying and documentation to push API schema discovery and understanding. Create a development portal that works as an API registry, providing means for generating access keys, usage statistics, onboarding documentation, and more. Promote developer-centric governance and versioned APIs to minimizes tedious frictional points.

6. Case Studies

6.1. Wealthsimple: Unified GraphQL Financial API

In contrast, Wealthsimple adopted the federated GraphQL gateway approach, which combines the separate microservices related to different domains (investments, payments, users profiles) under one API.

- Implementation: Used TypeScript and Apollo server with schema stitching and data sources to integrate RESTful and GraphQL microservices.
- Governance: Integrated GraphQL Hive for schema validation and usage monitoring enabling data-driven schema evolution.

- Outcomes: Improved developer confidence, enhanced schema reliability, faster feature rollout, and unified developer experience.

6.2. Branch Insurance: API Management for Embedded Insurance
Branch deployed Tyk's GraphQL API management and developer portal to expose APIs to third-party insurance partners.

- Use Cases: Developed client-facing and internal APIs for quote, policy purchase, and staff management functionalities.
- Advantages: Speedy on boarding for new partners, Security - token-based authentication, and standardized API governance across different environments (prod, staging, dev).
- Outcome: Increased partner integration efficiency and reduced time-to-market for embedded insurance products.

6.3. Fraud Detection at Fortune 500 Financial Services Firm

Using real-time graph analytics powered by Neo4j and integrated with AWS managed GraphQL services, this firm improved fraud pattern detection and reduced manual review times.

- Approach: GraphQL API layered over graph databases to expose complex relationships and transaction anomalies.
- Impact: Significant reduction in fraudulent transaction value, improved detection accuracy, and operational cost savings.

7. Challenges and Solutions

7.1. Legacy System Integration

Legacy mainframes and ERP systems often lack native API support. Middleware adapters or real-time event streaming architectures serve as bridging components, abstracting legacy interfaces into modern API surfaces.

7.2. Privacy and Regulatory Dynamics

Data residency, consent capture, and "right to erasure" under regulations like GDPR require dynamic data policies at the API layer. Leveraging API gateways with policy enforcement and masking mitigates these risks.

7.3. API Lifecycle Governance

Proactive API lifecycle management including versioning, deprecation policies, and detection of unused APIs prevents sprawl and enhances security posture.

7.4. Query Complexity and Denial of Service Risks

GraphQL's expressive queries necessitate robust anti-abuse controls via query complexity scoring, maximum depth, rate limiting, and persisted queries to maintain system stability.

8. Future Directions

Automated Schema Evolution: AI-based solutions for impact assessment and automated schema evolution that support backward compatibility. AI-Enhanced Query Optimization: Utilizing ML algorithms to anticipate frequent queries, allowing dynamic precomputation and result caching. Privacy-Aware Computation: Integration of homomorphic encryption, zero-knowledge proofs, and federated learning in API layers for secure multi-party computations. DeFi and API Interoperability Across Institutions: The future of API ecosystems for decentralized finance and cross-regulatory compliance will require open extensible GraphQL standards.

9. Conclusion

The paradigm of exposing financial data products as GraphQL-based self-service APIs ushers in a transformative era in digital finance. By enabling precise, flexible data retrieval clients empower key stakeholders—from analysts to regulators—to access curated, up-to-date financial data swiftly and securely. Attention to architecture makes data pipelines available and integrated; API management is federated, and security is enforced through multiple layers. Capacity schemas and defined terminology make data more usable and scalable and provide a compromise between flexibility and governance. Tools available to developers and portals accelerate datification through adoption and innovation, leading to a culture shift in the organization surrounding data.The approach is validated as a competitive advantage leaver due to the real-world gains shown by the likes of Wealth simplee, Branch, and Fortune 500s – lower latency, improved risk management and compliance, and better collaboration. With increasing complexity of financial ecosystems, the combination of AI-assisted query processing and privacy-enhancing technologies will also amplify the functionality and integrity of data product APIs. As the decentralized finance and open regulatory frameworks emerge, secure and composable financial API ecosystems will have broadening horizons. By having and investing in this API-first data-product culture, organizations are equipping themselves not only to be a service but as a leader that pushes innovation and driven the future of finance. Future success will be defined by how well change is embraced with a company and how security and

governance is check to enable a marketplace of value for data to be efficiently and effectively used on the go. This vision marks a decisive step toward resilient, agile, transparent, and customer-centric financial services empowered by trusted, self-service data access.

References

[1] M. Zachariadis and P. Ozcan, "The API economy and digital transformation in financial services: The case of open banking," SSRN Electronic Journal, 2017.

[2] "Modernizing legacy financial systems with API-driven architectures," World Journal of Advanced Engineering Technology and Sciences, vol. 14, no. 3, pp. 450–461, Mar. 2025.

[3] "A comprehensive analysis of GraphQL," SSRN Electronic Journal, Jan. 2024. [Online]. Available: https://papers.ssrn.com/sol3/Delivery.cfm/4915678.pdf

[4] Akamai Technologies, "API security in financial services: Mitigating risks and ensuring trust," White Paper, Mar. 2024.

[5] Financial Data Exchange (FDX), "Introduction to APIs," Tech. Rep., Dec. 2024.

[6] AppSentinels, "API Data Governance: Master Guide to Protect Sensitive Data," Tech. Rep., Sep. 2025.

[7] The Guild, "Building a unified financial API: Wealthsimple's Hive implementation," Oct. 2025. [Online]. Available: https://the-guild.dev/graphql/hive/case-studies/wealthsimple

[8] Tyk, "Branch case study: API management for GraphQL in insurance," Jun. 2023. [Online]. Available: https://tyk.io/case-studies/branch/

[9] Neo4j, "Real-time graph analysis for fraud detection," Feb. 2025. [Online]. Available: https://neo4j.com/customer-stories/fortune-500-financial-services-company/

[10] DataIntelo, "GraphQL for Financial Services APIs Market Research Report 2033," Market Research Rep., Sep. 2025.

[11] "How modern banking is using GraphQL," Nordic APIs, Oct. 2024. [Online]. Available: https://nordicapis.com/how-modern-banking-is-using-graphql/

[12] F5 Networks, "The role and impact of GraphQL," Tech. Rep., Apr. 2023.

[13] J. Scott, "API management trends in 2025," API7.ai, Feb. 2025. [Online]. Available: https://api7.ai/blog/2025-top-8-api-management-trends

[14] Velotix, "How self-service data access revolutionizes productivity and efficiency," Mar. 2025. [Online]. Available: https://www.velotix.ai/resources/blog/how-self-service-data-access-revolutionizes-productivity-and-efficiency/

[15] Stripe, "What are financial APIs? Here's what to know," Jul. 2024. [Online]. Available: https://stripe.com/resources/more/financial-apis-explained

[16] Plaid, "What is a financial API integration and how does it work?" Apr. 2025. [Online]. Available: https://plaid.com/resources/open-finance/financial-api-integration/

[17] Databricks, "Empowering business users with self-service data intelligence," Apr. 2025. [Online]. Available: https://www.databricks.com/blog/empowering-business-users-self-service-data-intelligence

[18] J. P. Morgan, "How financial institutions can unlock value from APIs," Jan. 2025. [Online]. Available: https://www.jpmorgan.com/insights/technology/application-program-interface/how-financial-institutions-can-unlock-value-from-apis

[19] S. Ray, "A comprehensive guide to GraphQL: Benefits and comparison with REST API," LinkedIn Pulse, Jan. 2025.

[20] D. Sutherland et al., "API architecture: A comprehensive guide to REST, SOAP, GraphQL, gRPC, and OData," DEV Community, Sep. 2024.

[21] Hypermode, "GraphQL's top benefits explained," Jun. 2024. [Online]. Available: https://hypermode.com/blog/advantages-of-graphql

[22] S. Srinivasan, "Financial-grade API security enables banking and fintech innovation," Forbes Technology Council, Nov. 2024.

[23] R. Morgan and K. Liu, "Benefits of using React and GraphQL to simplify FinTech data fetching," Yalantis Tech Blog, Aug. 2025.

[24] AWS, "Decision guide to GraphQL implementation," Sep. 2025. [Online]. Available: https://aws.amazon.com/graphql/guide/

[25] Apollo GraphQL, "Apollo GraphQL unveils first comprehensive API orchestration research study," Press Release, Jun. 2025.

[26] Collibra, "Data governance for financial institutions," Oct. 2025. [Online]. Available: https://www.collibra.com/use-cases/industry/financial-services

[27] Oracle, "How to implement self-service data analytics for finance teams," Oct. 2021. [Online]. Available: https://blogs.oracle.com/database/post/self-service-data-analytics-for-finance

[28] Acceldata, "Data products for fintech growth: Strategies you can't ignore," Sep. 2024. [Online]. Available: https://www.acceldata.io/blog/data-products-in-financial-services-innovative-approaches-to-gain-a-competitive-edge