

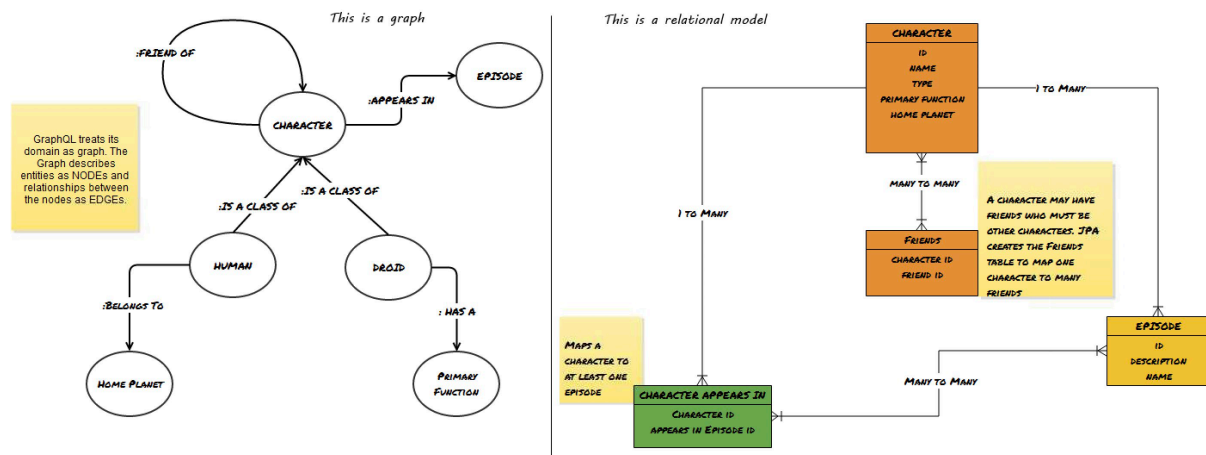
GraphQL vs Swagger (REST) Comparison




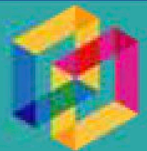




GraphQL: It is an open-source package for API which allows you to query your domain objects dynamically.

Overview: GraphQL is a query language for APIs, and a runtime for fulfilling those queries with your existing data. GraphQL provides a type system that you use to define your data, and use that the type system to allow clients to construct GraphQL queries.

GraphQL is not tied down to any specific database or storage engine, and especially GraphQL does not need to have data stored in graph database.

Example:



 SWAGGER (REST)	 VS	 GRAPHQL
<p>Multiple endpoints required. One endpoint per result set, leading to increased complexity</p>	 ABSTRACTION	<p>Only require single endpoint for all queries. Reducing complexity through abstraction</p>
<p>Due to its wide adoption, more authentication methods are available to ensure optimal security</p>	 SECURITY	<p>In its nascent stage, GraphQL does not yet support the latest security methods</p>
<p>Developers do not have an option to query what they need. Cache control is easier to implement</p>	 PERFORMANCE	<p>Due to its flexibility, developers will not suffer from under/over-fetching of data. However, cache control is more difficult to implement</p>
<p>Lower predictability as developers need to query for a response before performing any filtering</p>	 USABILITY	<p>High predictability as developers can obtain the exact result they are querying for</p>
<p>Industry-level adoption and REST is an extremely popular mechanism to serve API responses</p>	 POPULARITY	<p>GraphQL is touted as the future of API standards due to its simplicity and predictability. Less of a learning curve</p>

N.B: Apollo Client or urql on how to cache GraphQL queries.

Why should I use GraphQL?

GraphQL has a few key features that stand out. For example, GraphQL enables you to:

- Aggregate data from multiple UI components.
- Create a representation of your data that feels familiar and natural (a graph).
- Ensure that all of your data is statically typed and these types inform what queries the schema supports.
- Reduce the need for breaking changes, but utilize a built-in mechanism for deprecations when you need to.
- Access to a powerful tooling ecosystem with GUIs, editor integrations, code generation, linting, analytics, and more.