CluedIn

white paper

GraphQL vs SQL

2021 | cluedin.com

GraphQL vs SQL

There is not a shortage of query languages available to be able to easily filter and sort through mountains of data. Naturally SQL is one of the more popular languages of choice, but as more and more languages evolve, we realise that these languages are purpose built to solve a particular part of the overall challenge of querying data.

Derivatives of SQL have evolved to give the same familiarity of SQL, but catering for operating on non traditional relational stores. kSQL, uSQL are just some examples of this. However it is worth mentioning that the relational store is only one of many different types of data stores. What is also apparent is that although SQL is familiar and adaptable, it really is still rooted in relational style querying. This is why languages like Cypher, Lucene and others have evolved - in that SQL really was not a good fit. But like SQL is not a good fit for querying Search and Graph stores, Cypher and Lucene are not suitable for querying relational stores.

Enter GraphQL. A query language that from the outset confuses itself due to the nature that on the surface it sounds like a language to query Graph Databases. Where in reality, it is something more interesting. At CluedIn we really see CluedIn as a query language that was designed with a Polyglot database architecture in mind i.e. there is not one datastore to query, there are many. Although not as familiar as SQL, it is in our mind the most powerful and abstract query interface to be able to effectively achieve a virtualisation of data, no matter where the data is coming from.

> In effect, GraphQL is a language that relies on different parts of the query not only running against different sources, but the result also potentially being composed from many sources.

For example, imagine a situation where you wanted to ask a Search database to do the initial "predicate" or filter, but after that, based off what results there were, you would then jump over to another data source to do its work.

This is effectively a "join" across databases, but with smart optimisation so as to have the subsequent parts of the query essentially achieve simple index or key lookups, with the first query being the exhaustive search to filter the initial result set.

> The benefit in world of CluedIn is that we can provide a simple and abstract language for easily getting to the data that you want and not necessarily to mutate or aggregate the data before returning it. What this results in is a GraphQL implementation that does not require any joins to effectively bring back any data that you want. This removes the common bottleneck which is stitching together your result at query time and needing to understand what types of joins you will need to get to your result.

Although GraphQL can be implemented to fetch data from REST endpoints, CluedIn's implementation will talk to the 4 different databases that it uses to persist data i.e. Graph, Search, Blob, Key-Value. This allows you to provide immense flexibility when it comes to building queries but where the complexity of figuring out which database should run which part of the query, being handled by the CluedIn query optimiser. As a simple example, imagine you want to run a query such as "I want all Danish customers, all their history and all Employees connected to these customers" - you are effectively asking 3 databases to fulfil this query.

What we think is more exciting about this, is that this is a generic and abstracted language that easily allows us to wire in any future databases that CluedIn sees as valuable in adding to our technology stack, such as a Time-Series database. In addition to all these advantages, GraphQL provides inbuilt schema detection, so that we can have the query engine help us build our queries by suggesting what is possible with the GraphQL query implementation.

Naturally, at CluedIn we support all different ways to talk to your data including Lucene, SQL, GraphQL and more. Part of this is to provide familiarity, but in our mind GraphQL is the best fit language for our customers and partners to be able to get quick access to their data without having to deal with the common complexities of SQL.