ontotext

ONTOTEXT ANSWER:

To re-frame the question, “how can you scale GraphDB”? After all, no one really thinks that an enterprise-ready database adopted in numerous Fortune 500 companies would be impossible to scale.

GraphDB can scale in two ways:

- **Horizontally** on concurrent queries. The GraphDB cluster employs data replication to achieve a high availability (HA) setup. This means you have copies of your data in two or more places. The more copies you have, the more queries you can serve.
- **Vertically** on individual query and update speed. Of course, adding more workers who can serve concurrent queries does not solve the issue of slow individual operations. To this end, you can provide extra resources to GraphDB. We have followed the approach of materializing our inference and optimizing for storage, rather than the considerably more expensive memory. This means that when optimizing vertically, you want to address, roughly in that order, the speed of input and output operations, the number of CPUs and the RAM.

However, you can’t really scale vertically indefinitely. Some parts of the query would just have to run sequentially, and there’s a limit to the speed of processors. And if you have tens of billions of triples, the queries would start getting slow. Here comes data sharding. Most likely, your data has some logical dividers within it. If you are storing company data, there would be companies located under different jurisdictions.

For people, you can separate them by profession or address, whereas locations can, naturally, be divided by some top level parent location. This separation can be achieved all in the same repository, by using named graphs. If you keep the data in the same repo, you can benefit from inference across the named graphs and simplified management. However, the data structures could still be large, even with the context index enabled.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

```
PREFIX icecream: <http://example.org/creamy>
select * where {
    ?s a icecream:Vanilla .
}
```

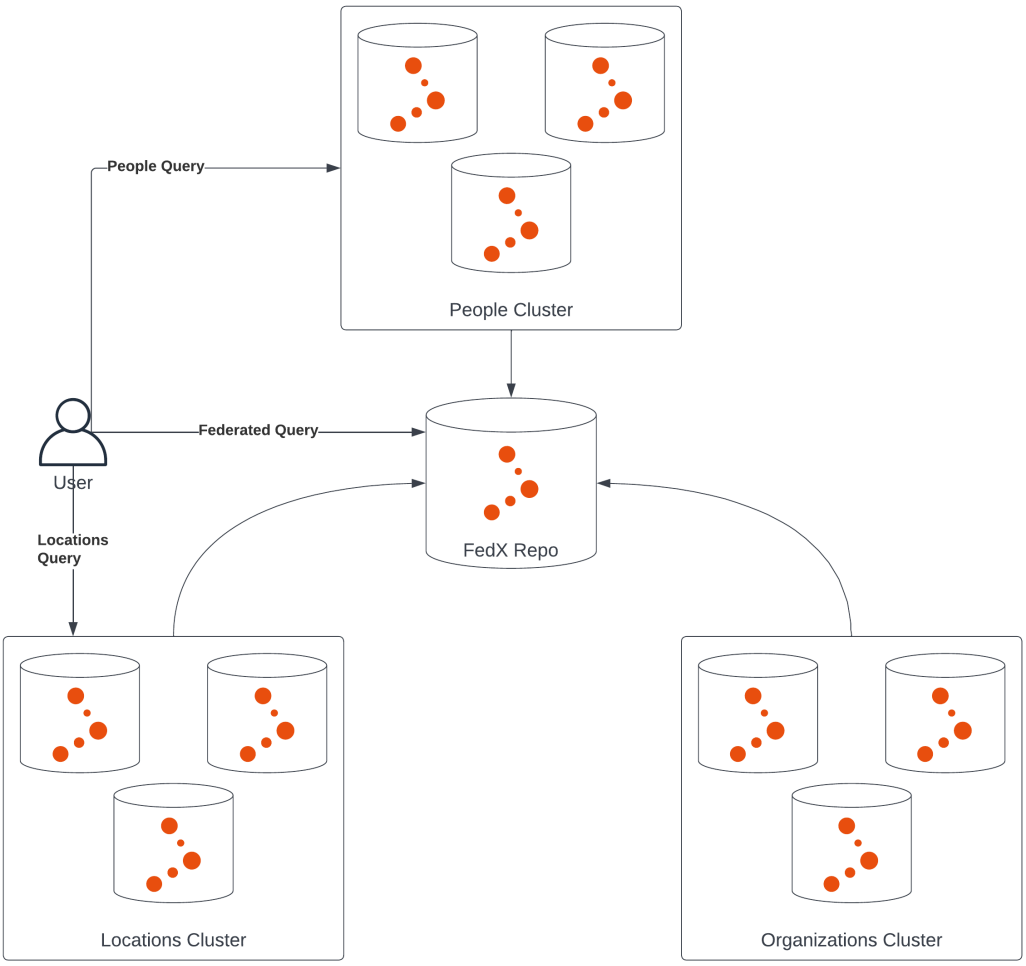
A simple query for hot days.

```
PREFIX icecream: <http://example.org/creamy>
PREFIX food: <http://example.org/food>
select * where {
    GRAPH food:desserts {
        ?s a icecream:Vanilla .
    }
}
```

Specifying a graph speeds up the search.

Our solution to this problem is employing federation. We have discussed federation earlier in this series of posts. You can use plain SPARQL federation if you are aware of the data's intricacies or you may instead switch to FedX federation to simplify query patterns. This means you can switch between the different repositories and query each one of them as needed or just ask the central federated repository for a simpler, cross-repository response.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com



Simple federated architecture

Help us get better: review GraphDB on Gartner

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com