SuperSQL Database

Architecture

www.clusterpoint.com

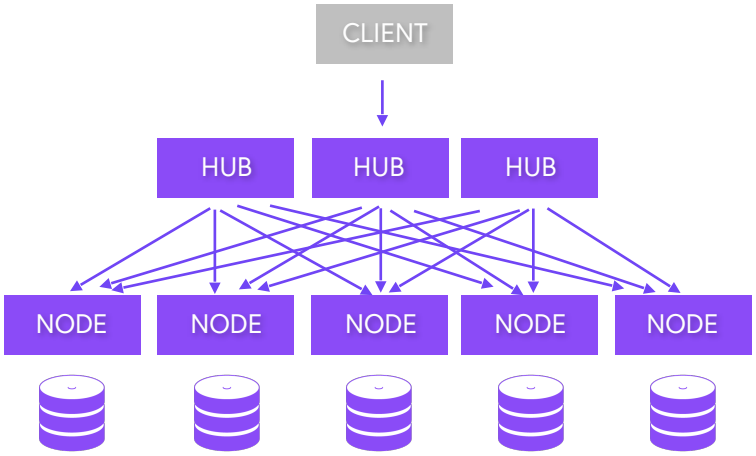
Introduction

In this paper we explain the architectural choices that give Clusterpoint 4 its capability to handle multiple data models, relational and NoSQL, in single database platform. Since the architecture allows addressing all the use cases of SQL, we refer to it as a SuperSQL database, as in a superset of SQL.

Architecture

We started off with a typical distributed architecture, which includes nodes hosting data and hubs scheduling requests between those nodes.

Figure 1. Architecture Diagram.



For the data model, we started with a simple NoSQL data model where data is managed in self-contained documents identified by a primary key. A document is a potentially hierarchical data structure represented in JSON or XML. Documents are grouped into collections and collections into a database (See Figure 1). The database is scheduled to reside on a certain set of computational nodes.

Depending on their configuration, there are two distribution mechanisms for collections:

1) A collection can be sharded and replicated, in which case a collection is split among many computational nodes to divide storage requirements and parallelize processing. The collection is also replicated to backup nodes for high availability. This distribution mechanism is suited for large collections which need to operate at scale.
2) A collection may be hyper-replicated, in which case a collection is present on every computational node where its database is scheduled. This distribution mechanism is suited for small collections, which would be effectively joined with sharded collections. Joins will be discussed in detail in subsequent sections.

Data Modeling Recommendations

Suppose that a logical data model of the data managed by a database is represented with a UML Class Diagram.

Classes

First, we recommend that classes be modelled as documents in the Clusterpoint database, with attributes in the class corresponding to fields on the document. A class with "flat" structure of single valued fields of primitive types would correspond to rows in a table using a simple SQL model. A more complex class can be represented using hierarchical JSON or XML structures.

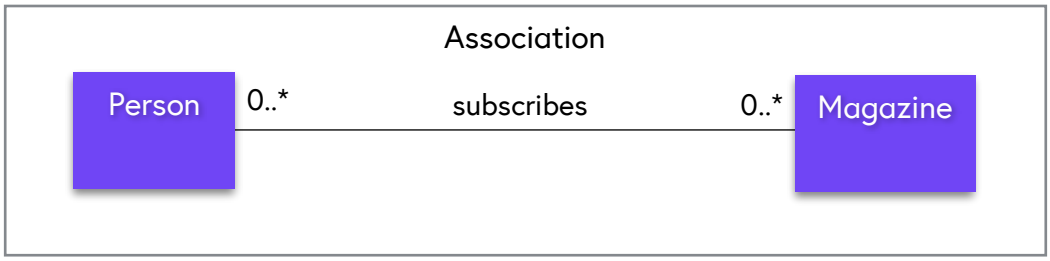
Thus, SuperSQL is more expressive than plain SQL, as a document may contain repeated fields, arrays, and nested objects along with simple SQL types. Nested objects may also be represented as an object within the class using a composition relationship.

Relationships

SuperSQL allows for a wide variety of expressive and powerful models for relationships between documents. Here, we provide the some recommendations and explain the performance implication of different access patterns.

Association

An *association* defines a simple relationship between classes. A particular instance of one class may be associated with a specific multiplicity of instances of other class. For example, in the above diagram a person may subscribe to zero or more magazines, and a magazine may have zero or more subscribers.



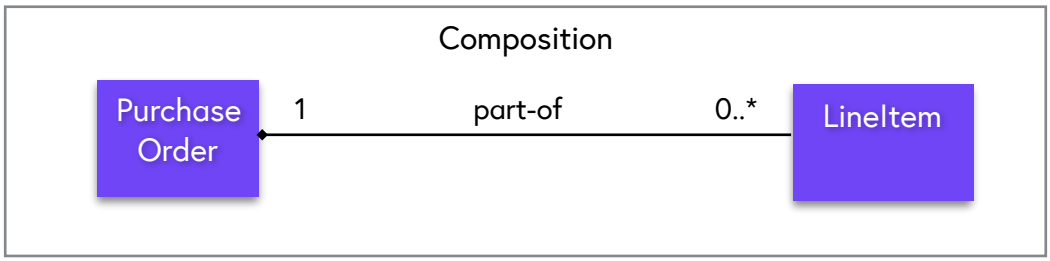
An association is modeled as a document containing a field referring to the primary key of another document. This is conceptually identical to a foreign key in SQL.

At query time documents are joined using this foreign key. Clusterpoint provides two types of joins: local and remote. A local join involves joining with a hyper-replicated collection, where all documents have been stored on the same computational node that handles the query. A remote join involves fetching the documents to be joined from a remote computational node. Under heavy load remote joins can become expensive, which is why we recommend hyper-replicating small, frequently-joined collections. On the other hand, hyper-replicated collections result in storage space and update cost overheads proportional to the number of nodes that handle the database.

SuperSQL modeling of association relationship between entities is comparable to ANSI-SQL; however it provides tunable performance tradeoffs within the distributed environment, giving you more control over the balance of query performance and storage use.

Composition

Composition is a "has a" association relationship, which implies a lifecycle dependency between instances of a container class and instances of the contained class.



There are several ways to potentially model a composition:

1) Since Clusterpoint allows hierarchical documents, composition can be modelled by just putting repeated groups inside the "composite" document. This is the simplest way to model composition, but we recommend it only if the lifetime of the parts is closely coupled with that of the composite or if multiplicity is low. Otherwise adding parts will result in updating the entire composite document every time and degrade performance as the composite document grows.
2) The most robust way to model composition is using the concept of a nested collection. In nested collections each document has a parent. Child documents can automatically access their parent and parents can similarly access their array of children from within their processing context.
3) Since composition is a special case of association it could also be modelled the same way using a foreign key. This would be a standard approach in pure SQL. However we recommend against this approach as it can negatively impact performance if the "part" documents are distributed across many nodes. Clusterpoint provides more explicit ways to model composition that take the distributed nature of the database into account.

Thus SuperSQL provides several effective ways to model composition in a distributed environment with nested collections or hierarchical documents. Performance-wise, nested collections are analogous to using columns in a NoSQL wide-column store, however Clusterpoint's combined JavaScript/SQL API takes it a step further with flexible and powerful in-database processing capabilities. Read more about computational capabilities in the "Computational Model" section.

Generalization

Clusterpoint's document model allows storing documents with different set of fields in the same collection. This makes it easy to model generalization by inserting documents representing subclasses into a collection named after a superclass. This is currently the recommended way to model generalization, however future versions of Clusterpoint will have a more explicit mechanisms for defining and manipulating sub-collections.

Consistency

For many applications it is vital to ensure the consistency of queries in the presence of multi-document updates.

Like most SQL databases, Clusterpoint provides ACID-compliant multi-document transactions. However Clusterpoint's transaction mechanism works in a distributed environment and scales horizontally when adding more computational nodes.

Computational Model

SQL databases operate on top of a relational model using a declarative method for specifying queries. Many relational database vendors provide proprietary procedural extensions to SQL that allow in-database processing.

Clusterpoint extends SQL with our JS/SQL query language that expands the common SQL query structure by allowing the execution of procedural JavaScript code. Clusterpoint presents a document view on the data where each data item is representable as JSON or XML document in serializable form. The JavaScript computational capabilities allow you to manipulate documents at query time as if they were native JavaScript objects in memory. Since JavaScript objects can represent hierarchical data and repeated structures like arrays of primitives or arrays of objects, the JS/SQL combination allows for expressive and powerful queries and computations within the database itself.

Conclusion

SuperSQL represents a convenient and powerful option for data modelling that standard SQL doesn't offer, like modelling class hierarchies and composition relationships through JSON or XML document models. At the same time these hierarchical objects are accessible through a JS/SQL API, which allows hierarchical objects to be manipulated through procedural JavaScript extensions.