Name:	Mohammad Azimi
Affiliation:	Evalytica, Inc.
Email:	mohammad.azimi@evidera.com
Presentation type (s):	**Poster**

A JSON-Based Implementation of the OMOP Common Data Model Can Enable Scalable Low-Cost Analytics on Larger Datasets

Mohammad Azimi, PhD[1], Steve Lyman, BS[1], Stephanie Reisinger, BS[1]
[1]Evalytica, Inc., San Francisco, CA

Abstract

The exponential growth of healthcare data brings with it new challenges that require the adoption of standards and technologies designed to ingest and process large volumes of data in a performant and cost-effective manner. Standardized data models based purely on a relational design, relying on SQL for analysis present challenges in data ingestion, low-cost and on-demand scalability, time required for application development/maintenance/testing, and present a barrier to rapidly implementing updates to the common data model format. We propose a technology-agnostic and human-readable JSON-based representation of the common data model (CDM) that can be maintained in parallel to the relational model, while allowing users to better deal with the challenges of large-scale datasets.

Introduction

Analysis of healthcare data has historically been done within relational databases. While the size of these datasets grow exponentially due to the adoption of EHRs as well as the richness of data captured, advances in relational database technology have been slow to keep up. Recent advances to database technologies involve adding support for parallel query execution (i.e. Postgres 9.6 added parallel query support to a subset of functions) which helps improve performance but requires vertical scaling of hardware to take advantage of the functionality. The alternative approach is horizontal scaling of the database which typically involves partitioning the data across multiple machines. Third party tools and modern data warehouse type services such as Amazon's Redshift (OHDSI supported) try to abstract the data partitioning task away from the user but have many drawbacks, including high cost to operate since they are always on and require scaling compute and storage in parallel as the amount of data grows and suffer from low concurrency limits due to memory constraints when operating on entire datasets. Traditional relational databases are well suited for applications where ACID compliance and referential integrity are requirements (i.e. financial and healthcare data capture). However, the typical access pattern of data for OHDSI users is read-intensive. Furthermore, the use of set-based logic via a declarative language such as SQL is well suited for processing aggregate summary statistics (i.e. OHDSI ACHILLES functionality) and ad-hoc queries but presents significant overhead and challenges in terms of development time, testing and maintainability compared to iterative or procedural languages when identifying complex temporal relationships in single patient histories.

Document-Oriented Databases and Infrastructure Elasticity

We have developed a document-oriented representation of patient data that captures key attributes of the OMOP CDM without loss of information along with a simple process of serializing CDM data from a relational format to the document format and vice versa. This document representation utilizes JavaScript Object Notation (JSON) and

stores an entire patient's data as an object with demographic criteria and collection of events as attributes of the patient object. The JSON representation of the patient's healthcare record lends itself well to easy deserialization and instantiation of a patient object within an object-oriented programming (OOP) language. The ease of use within OOP and development of cohort and analysis pipelines that leverage person-at-a-time processing facilitate development, testing and maintenance of software tools. The use of document-oriented storage of data simplifies the task of partitioning data and allows the community to leverage a wide range of high-concurrency data stores that serve the single purpose of data storage rather than the traditional relational database that serves as both the storage and compute layer. This decoupling of storage and compute enables infrastructure elasticity when leveraging cloud resources. The availability of fast and high-throughput networks in the cloud further reduce the requirement for data locality. Table 1 demonstrates a comparison of the costs between a high-performance, relational data warehouse service (Redshift) and the utilization of decoupled storage and compute services (Elastic Compute Cloud and Simple Storage Service , EC2 and S3 respectively.

Table 1. Comparison of costs of a relational data warehouse service[1] and decoupled storage/compute services[2,3].

Technology	Compute (ECU)	Storage (TB)	Reserved Cost ($/hr)	Spot Cost ($/hr)
Redshift (dc1.large)	7	0.16	0.20	NA
EC2 (r4.large) S3	7	0.16	0.133 (compute) 0.005 (storage)	0.015 (compute) 0.005 (storage)
Redshift (ds2.xlarge)	14	2	0.68	NA
EC2 (r4.xlarge) S3	14	2	0.266 (compute) 0.064 (storage)	0.034 (compute) 0.064 (storage)

The first observation here is that decoupling of storage and compute enables data to persist long-term while allowing for compute resources to be completely scaled down during periods of inactivity to enable significant cost-savings in "bursty" workloads, eliminating the major contributor to cost. Another observation is that the tight coupling between storage and compute can make it prohibitively expensive to store a large number of datasets since the addition of more storage requires the addition of compute power that may be unnecessary. Finally, by decoupling compute, we can leverage excess unused capacity in the cloud (i.e. spot instances) at a fraction of the cost. This is not possible with traditional relational databases where dedicated instances are required since the loss of compute capacity corresponds to loss of data and time-consuming re-ingestion of data.

Conclusion

We have implemented and used a JSON-based analog of the OMOP CDM that has been tested and used in production workloads. This model provides significant flexibility in performance and cost tuning of cohort building and analysis applications. Next, we plan to publish and iterate on this model for use by the OHDSI community.

References

1. Amazon Redshift Pricing. https://aws.amazon.com/redshift/pricing/. Accessed 29 Aug. 2017.
2. Amazon EC2 Pricing. https://aws.amazon.com/ec2/pricing/. Accessed 29 Aug. 2017.
3. Amazon S3 Pricing. https://aws.amazon.com/s3/pricing/. Accessed 29 Aug. 2017.