Name:	Don O'Hara
Affiliation:	Evalytica, Inc.
Email:	don.ohara@evidera.com
Presentation type	Poster

Highly scalable patient-at-a-time transformation of observational databases into OMOP CDM v5 format using cloud-based open source tools

Marzieh Golbaz, MS[1], Donald O'Hara, MS[1],

Mohammad Azimi, PhD[1], Steve Lyman, BS[1], Stephanie Reisinger, BS[1]

[1] Evalytica, Inc., San Francisco, CA

Abstract

We demonstrate the use of a distributed architecture built on Amazon Web Services to perform ETL transformations of large-scale observational databases. The transformation programs were developed in-house using Python, based on OMOP CDM transformation specifications that are publicly available [1,2]. Instead of traditional SQL set based processing, the transformation programs process each person independently, enabling the data transformation to be easily distributed and massively scaled. Using this distributed architecture, we transformed several hundred million person years of observational data into the CDM v5 format multiple times, using a different number of distributed nodes for each transformation. Not surprisingly, the results demonstrate that the more nodes that are used, the faster the data transformation executes. In addition, transformations done using this architecture perform extremely well, preliminary results range from less than 1 to 8 hours total wall time for the entire transformation, depending on the number of distributed nodes that were used.

Introduction

A major bottleneck in the analysis of very large (500+ m person year) databases is the process of transforming the original dataset into the OMOP Common Data Model (CDM v5) format [3,7,8,9]. The storage of the original input data, intermediate work files and tables, and the transformed CDM data files can require several terabytes of flat file and database files. Additional storage is required for indexes needed to optimize performance. The transformation process is typically implemented in SQL code running on a large DBMS, such as Oracle or Microsoft SQL Server. Executing set based operations on these large datasets in a time-performant manner requires enterprise-sized database servers, along with the database administration and SQL programming expertise required to code and optimize complex transformation steps. It is not unusual for an ETL transformation of a 500m person year database to take many days to complete.

We have developed a transformation process using open source tools [5,6] and Python programs that implement person-at-a-time processing. The processing is distributed in the Amazon cloud [4], and can be scaled up or down based on the number of distributed nodes (workers) that are used. The performance is very good, and while the first release does not create all CDM v5 tables (costs and observations tables are not yet created), we feel that adding that processing will not negatively impact our run times.

Methods

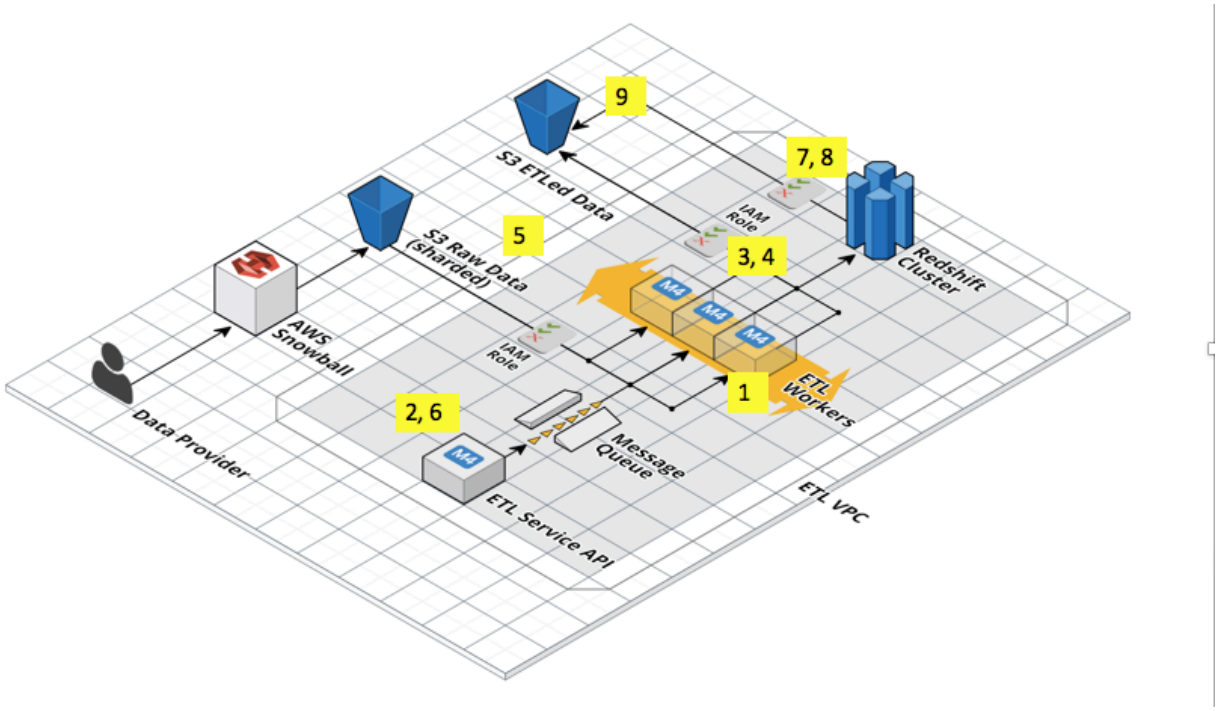
While our ETL processing pipeline includes both pre-processing steps when the data is originally received, and post processing steps to report on the results, the focus of this research is the actual transformation into CDM format. The CDM transformation includes the following steps, which are illustrated in figure 1.

Transformation into CDM format:

1. N worker nodes allocated.

2. Coordinator program uses RabbitMQ to send messages to each worker nodes.
3. Each worker node executes multiple subprocesses.
4. Each subprocess transforms a set of persons, one person at a time.
5. All work is done via flat file manipulation, reading from and writing to the Amazon S3 file storage system.
    - Each process generates CDM files in CSV format, as well as a flat-file, single-person representation of the CDM in JSON format for further PaaT (Person-at-a-Time) processing.
    - The PaaT approach continues to simplify and provide performance enhancement of downstream analyses.
6. Coordinator program handles error processing and reruns any shards that may have failed.
7. CDM flat files are bulk imported into Redshift database.
8. In house developed summarization run against CDM data
9. Summary results extracted from Redshift.

Figure 1: ETL Transformation architecture



Preliminary Results

We have executed our transformation process against the full dataset twice, varying the number of concurrent workers used. We have identified bottlenecks in accessing the network file system (S3), and are currently testing fixes which we believe will reduce run time further.

The input dataset, containing 280m person years of observational data, was sharded into 800 groups of approximately 100,000 persons. In the first transformation 25 concurrent workers were allocated to process each shard; in the second we used 50 concurrent workers. The results are shown in Table 1 below.

Table 1: Transformation times based on number of workers allocated

25 concurrent workers	total wall time 8 hours

50 concurrent workers	total wall time 4 hours
*200 concurrent workers ** *planned; not yet executed*	*total wall time <1 hour (estimated)*

Conclusions

We demonstrate a significantly reduced wall time for ETL transformation, using open source tools executing in the AWS cloud infrastructure. Complex set-based manipulation of large volumes of data using SQL code can be replaced by much simpler person-at-a-time logic implemented in procedural languages (Python in our case). This approach utilizes on-demand infrastructure and can be scaled as needed to meet ETL processing time requirements.

We are currently extending the person-at-a-time distributed design to analysis applications that use the CDM data.

One disadvantage of this approach is the increased complexity of managing distributed processes, to ensure that any failures are restarted correctly and that all data is processed correctly. We have utilized open source tools and libraries (such as Python, PostgreSQL[5], and RabbitMQ[6]) to minimize the amount of custom development required. Another drawback is that while the processes are executing, the CDM data being created is not immediately available in a database for querying or for quality control. However, the greatly reduced transformation time frees up other time in the overall ETL pipeline that can be utilized for quality control analysis and verification.

Conflicts of Interest

The authors work at a software company producing a cloud-based, software as a service (SaaS) platform of healthcare data analysis applications.

References

1. OMOP Common Data Model (CDM V5.0) Clinical Practice Research Datalink (CPRD) Mapping Specification ; https://github.com/OHDSI/ETL-CDMBuilder/blob/master/man/CPRD/CPRD_ETL_CDM_V5.doc

2. Johnson & Johnson. Common Data Model (CDM v5.0) ETL Mapping Specification for TRUVEN (CCAE and MDCR); https://github.com/OHDSI/ETL-CDMBuilder/blob/master/man/TRUVEN_CCAE_MDCR/Truven_CCAE_and_MDCR_ETL_CDM_V5.doc

3. OMOP Common Data Model (CDM) v5: http://www.ohdsi.org/web/wiki/doku.php?id=documentation:cdm:single-page

4. Amazon Web Services: https://aws.amazon.com/

5. PostgreSQL; https://www.postgresql.org/

6. RabbitMQ messaging broker; https://www.rabbitmq.com/

7. Voss EA, Makadia R, Matcho A, et al. Feasibility and utility of applications of the common data model to multiple, disparate observational health databases.Journal of the American Medical Informatics Association : JAMIA. 2015;22(3):553-564. doi:10.1093/jamia/ocu023.

8. Overhage JM, Ryan PB, Reich CG et al. Validation of a common data model for active safety surveillance research. J Am Med Inform Assoc. 2012;19(1):54-60.

9. Reisinger, Stephanie J et al. “Development and Evaluation of a Common Data Model Enabling Active Drug Safety Surveillance Using Disparate Healthcare Databases.” Journal of the American Medical Informatics Association : JAMIA17.6 (2010): 652–662. PMC. Referenced Web. 22 June 2016.