DATA VIRTUALIZATION

Implementation and Performance Analysis

**Kristin Martin**

Boston University

Table of Contents

[Introduction 2](#_Toc34253047)

[What is Data Virtualization and Why is it Important? 2](#_Toc34253048)

[ChMS: Dataset Overview 3](#_Toc34253049)

[ChMS Entity Relationship Diagram 4](#_Toc34253050)

[Why Data Virtualization for ChMS? 5](#_Toc34253051)

[Introduction to Denodo Data Virtualization Platform 6](#_Toc34253052)

[Denodo Platform Approach to Data Virtualization 6](#_Toc34253053)

[Costs, Benefits, and Risks to Deploying with Denodo 9](#_Toc34253054)

[Implementing Data Virtualization 11](#_Toc34253055)

[Connecting Relational Database(s) (JDBC) 11](#_Toc34253056)

[Connecting API Data Sources: Interpolation Variables & Flattening 12](#_Toc34253057)

[Creating Derived Views 14](#_Toc34253058)

[Query Performance Analysis 14](#_Toc34253059)

[Denodo System & Query Performance 14](#_Toc34253060)

[SQL Server System & Query Performance 16](#_Toc34253061)

[SQL Server Query 16](#_Toc34253062)

[Analysis & Observed Outcomes 18](#_Toc34253063)

[Data-Specific Knowledge is Key 18](#_Toc34253064)

[Cache is King: The Importance of Configuring Denodo Administrative Tools 18](#_Toc34253065)

[Alternatives to Data Virtualization 19](#_Toc34253066)

[Data Lakes 19](#_Toc34253067)

[Apache Spark 19](#_Toc34253068)

# Introduction

## What is Data Virtualization and Why is it Important?

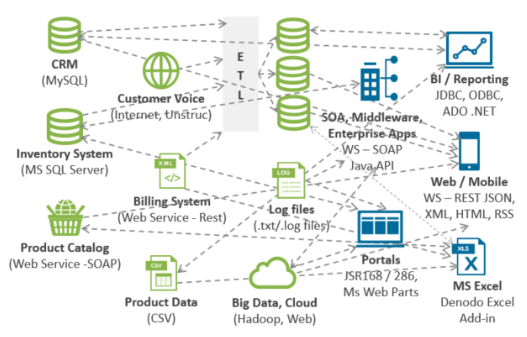
Data architectures are more complex than they have ever been, and their complexity is only increasing. As data architectures become more complex, the diversity and volume of the processes required to extract meaningful data from these disparate sources can be daunting or even insurmountable. Figure 1 (van der Lans, Modernizing Data Architectures for a Digital Age Using Data Virtualization, 2019) illustrates an example of a data architecture and the level of complexity required to meet the business’ data needs.

Figure 1 (van der Lans, Modernizing Data Architectures for a Digital Age Using Data Virtualization, 2019).

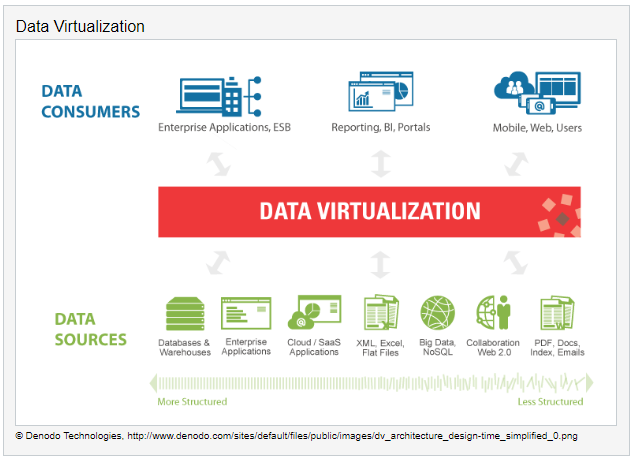
Data virtualization offers an efficient, robust, and scalable solution to this issue, abstracting the technical details of these connections and presenting a single interface to the application or user. Specifically, “…virtualization means that applications can use a resource without concern for where it resides, what the technical interface is, how it has been implemented, which platform it uses, how large it is, how much of it is available” (van der Lans, Data Virtualization:Selected Writings, 2019). Data virtualization does not replicate data, it does not require ETL, and it accesses data in real-time. It is not intended to be a replacement for any data source; it simply creates the connections required for aggregation, reporting, input, output, etc. The extent of the available data sources and the level of abstraction can vary depending upon the data virtualization provider, but the field is growing and more businesses are entering the data virtualization space (Denodo Community, n.d.; van der Lans, Modernizing Data Architectures for a Digital Age Using Data Virtualization, 2019). Some of the biggest use cases for data virtualization surround its ability to execute parallel query processing, the easy staging of big data, speeding up slow data sources, simplifying data lakes, and accessing remote big data (van der Lans, Data Virtualization:Selected Writings, 2019). It tends to require less intensive I/O, less required infrastructure, and less specialized knowledge.

Figure 2 (Polnar & Schudy, 2019)

## ChMS: Dataset Overview

The ChMS idea begun its first iteration of development in Boston University’s Metropolitan College Computer Science department in a database design and implementation course (CS 669). ChMS is a yet-to-come-fully-into-existence, full-service church management system. ChMS allows places of worship and other member-based non-profit service organizations to manage their transactions (e.g., donations, event attendance), their client relationships. This analysis will illustrate that, with data virtualization, ChMS clients will be able to external data sources to facilitate access to community resources, and it will explore how to glean data-driven insights into community needs and access.

A full entity relationship diagram is provided below in ChMS Entity Relationship Diagram outlining the original design of the database, with a few updates to make the data set more relevant for the data virtualization application here. This analysis focuses on the CRM aspect of ChMS but all entities were included for reference.

***Note: there are numbered SQL script files including in the submission of this document for those interested in loading the database for testing.***

### ChMS Partial Entity Relationship DiagramC:\Users\Vincent\OneDrive\Kristin School\CS 779 Advanced DB Management\Term Project\Final Deliverable\martinKristin_ChMS_ERD.pngWhy Data Virtualization for ChMS?

The focus of research into data virtualization as an effective tool for ChMS surrounded how places of worship are currently interacting with data (if at all), and what successful early adopters of technology in the industry were focusing on. When looking at contemporary Christian churches (those most likely to adopt ChMS first), there is an overwhelming lack of understanding on how to access data as an organizational asset coupled with an equally overwhelming desire to do so (Engel, 2019; Pushpay, 2018). Churches often struggle lately to connect with their communities, which is likely contributing to systemic decline in church attendance (speaking in the United States). However, as Engel (2019) asserts, “some churches are thriving as they discover new ways to meet the spiritual and communal needs of those in their congregations and communities…with data.” Of course, any data driven insights should be considered as a part of a larger strategy to broaden community engagement, not as the sole strategy. Overall, “using data in your church is about understanding your congregation’s needs and then matching those needs with relevant solutions” (Engel, 2019).

(Engel, 2019)

With data virtualization, ChMS users can connect to public sector data stores and glean insights from demographic data and access data stores of community service resources (e.g., WIC authorized retailers, drug treatment facilities) without the need to house any of that data or any of the infrastructure requirements that would , or even understand any of its technical complexities. Data federation, coupled with the ability to parallel process (as will be demonstrated in Virtualized Environment Versus SQL Server Environment) will save huge amounts of processing time, without adding complexities to user interactions, all in an integrated environment.

Although not implemented as a part of this report, the big data applications for data virtualization with ChMS are impactful and accessible. With a connection to the federal government’s API for census information, users can access current and historical demographic data free, and data virtualization providers allow access to that data to help places of worship answer important questions about their communities, like:

* What types of community support does our congregation most need?
* Was there a(n) [increase/decrease] in [X] over the last [##] years in my community since we began [Y] initiative?

In other words, “…instead of attendance, let’s think instead of decreased crime in the neighborhood, improved schools, and an increase in intact families” (Engel, 2019). Finally, it merits restating that data driven initiatives should always be specifically relevant to the organization and continually tuned and monitored to ensure relevant data and insights are being output.

# Introduction to Denodo Data Virtualization Platform

## Denodo Platform Approach to Data Virtualization

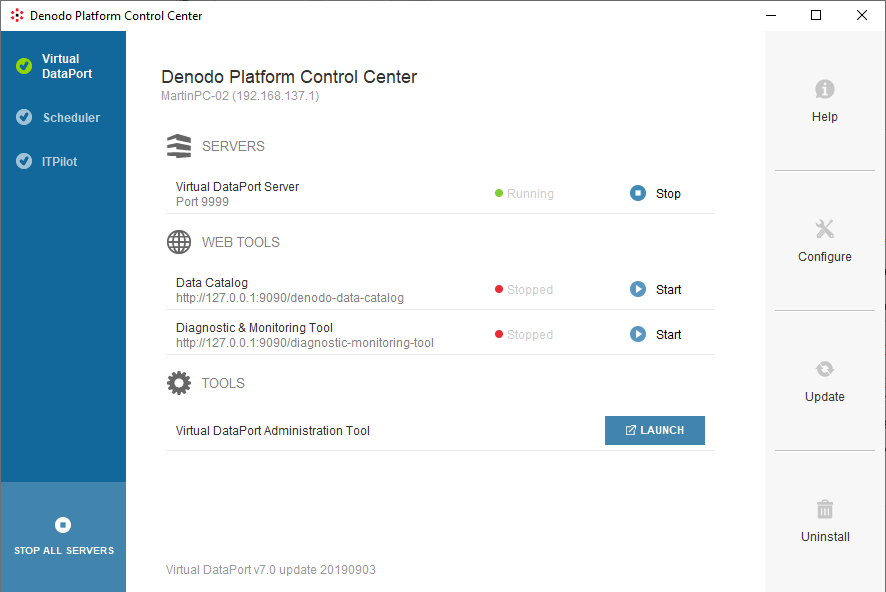
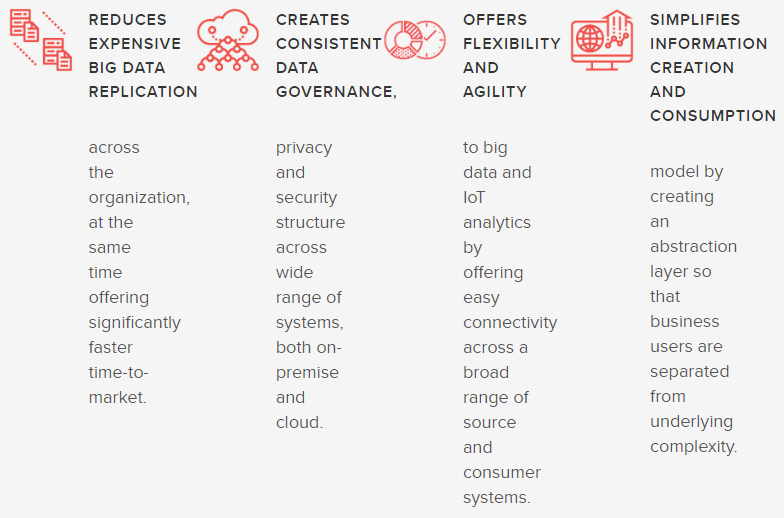
In order to grasp the most robust implementation available, this analysis implemented a data virtualization server utilizing the platform from Denodo Technologies, an industry leader in data virtualization and currently the provider of the most robust pure-play data virtualization tool on the market (Denodo Solutions, n.d.).

Figure 3 (Denodo Community, n.d.)

Denodo Technologies’ Denodo Platform 7.0 is the current release of Denodo’s data virtualization server. It runs on a Java Virtual Machine, and is installed with a data catalog, diagnostic tools, and the console from which user interacts with the virtualized environment (VDP) (see Figure 7). Lacking a business expense account, this academic analysis opted to utilize Denodo Express 7.0, a free, limited-feature release of Denodo’s platform. As a result, some exciting and important Denodo features will not be demonstrated in this report, most prominently the ITPilot, which enables unstructured data integration and is specialized for big data sources, followed by the disabling of user and role creation and version control systems, all present in the full version of Denodo Platform 7.0 (Denodo Community, n.d.).

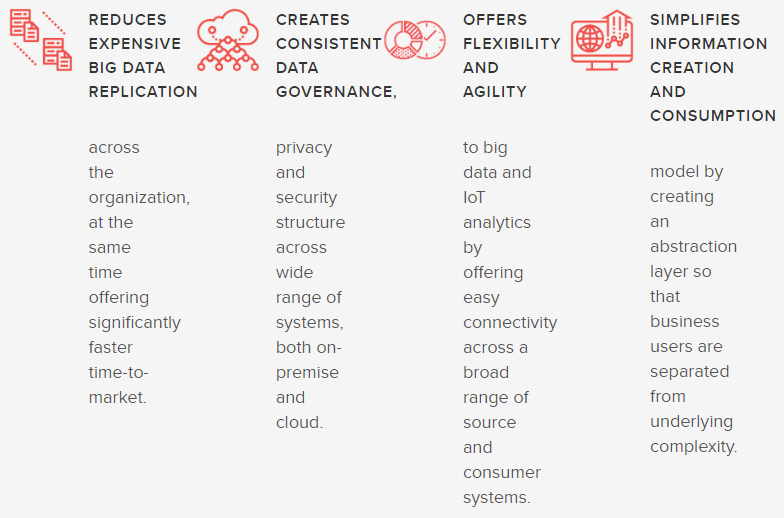
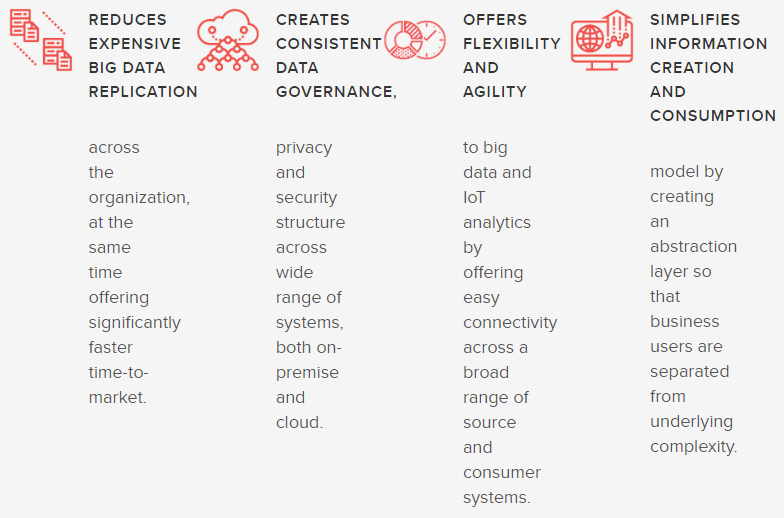
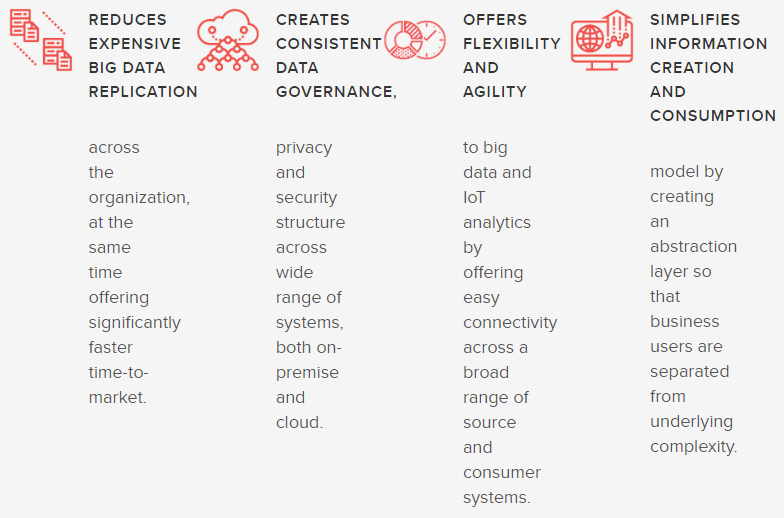
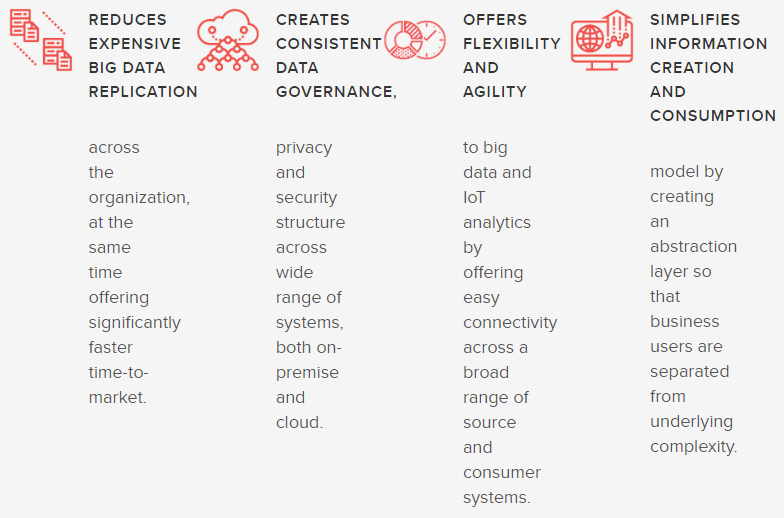
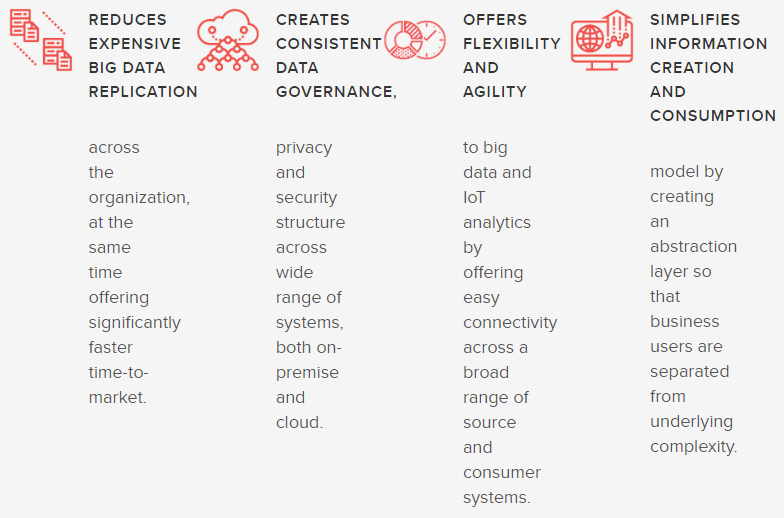
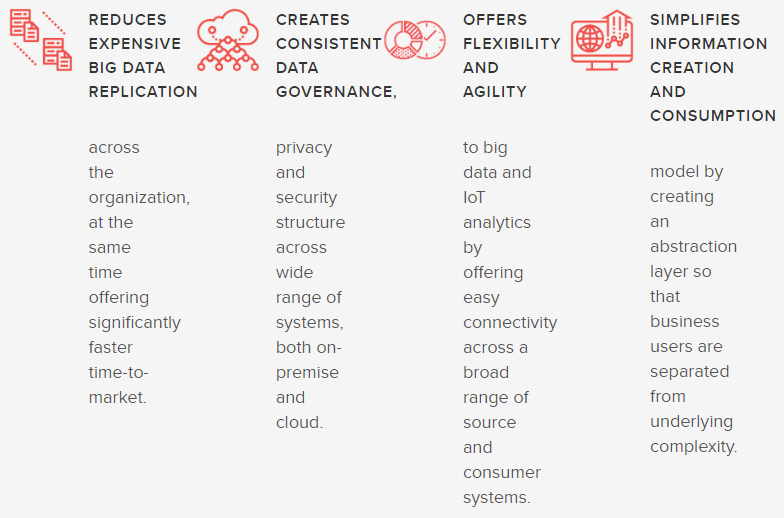


Figure 4 (Denodo Solutions, n.d.)illustrates Denodo’s approach to data virtualization. Its goal is to connect heterogeneous data sources, combine them in meaningful ways, and consume them either in the Denodo interface or as visualizations, tables, and application data exported from Denodo (Denodo Community, n.d.). As Figure 4 (Denodo Solutions, n.d.)details, Denodo integrates with nearly any available data source and abstracts the technical details of different data sources inside of wrappers.



As Figure 6 (Denodo Community, n.d.)shows, Denodo’s Virtual DataPort (VDP) is the hub for the virtualized environment from which all calls to data sources are initiated. Figure 6 also details the data types, protocols, and services that Denodo interacts with. In this analysis, mostly JDBC (SQL Server) wrappers and JSON (REST, Flat Files) wrappers were used. Finally, import to note is the cache as a part of the VDP. A properly configured cache allows Denodo to speed up the query process by storing parts of previously queried data.

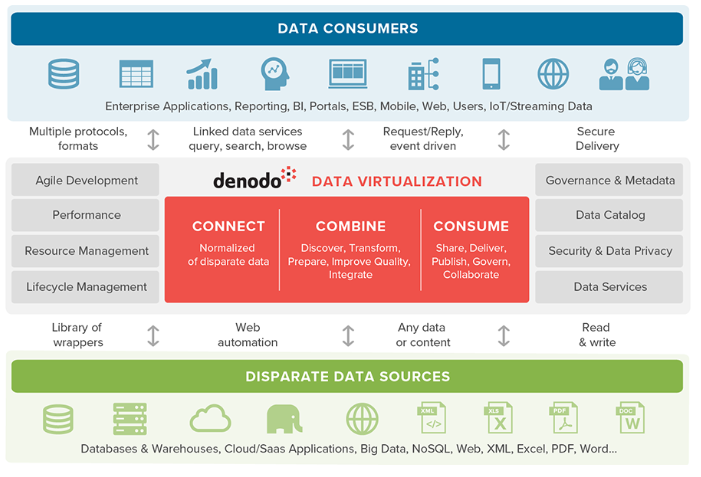


Figure 4 (Denodo Solutions, n.d.)



Figure 5 (Denodo Sample JSON Wrapper)

### Costs, Benefits, and Risks to Deploying with Denodo

Figure 6 (Denodo Community, n.d.)

The robust capabilities of Denodo’s virtualization platform described above constitute a number of great reasons to implement data virtualization using Denodo. Most notably for ChMS are fast analysis, parallel processing data, and easy staging of big data (van der Lans, Data Virtualization:Selected Writings, 2019; Denodo Solutions, n.d.; Pushpay, 2018).

One potential major drawback for users with limited budgets, which nonprofit organizations often are, is that, according to Blomberg, some older databases do not allow a database license to transfer to a virtual system. Specifically, “…older Oracle databases, it is not possible to transfer the previous database licenses 1:1 to a virtualized system since the charges refer to the ‘potential’ performance of the system and now what it actually uses” (Blomberg, 2015).

Additionally, data virtualization does require a system with relatively high processing power and physical memory requirements (see hardware requirements in Figure 7)(Denodo Community, n.d.).

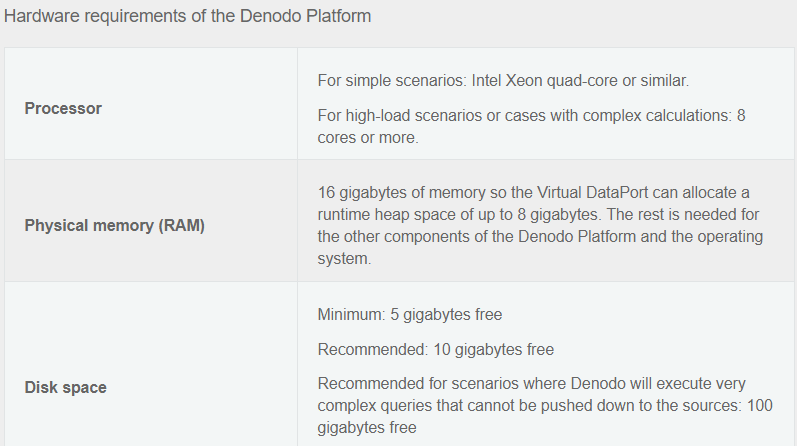


Figure 7 (Denodo Community, n.d.)

However, this is flexible depending on user computing needs, and contrary to the cautions of some data virtualization critics, data virtualization is not a “hog” when it comes to system resources, when managed properly and when allocated the proper memory required for the organization’s data needs.

# Implementing Data Virtualization

## Connecting Relational Database(s) (JDBC)

As someone without integration experience and a need for extra support configuring a JDBC database, the Denodo support topics in this area were surprisingly lacking and required scouring research engines to assist in the set up. However, the focus of this analysis is not on JDBC connections, it is on the virtue of Denodo as a platform for ChMS. So while poor documentation in this area is notable relative to ensuring the proper drivers were installed in the proper places, the proper syntax was used in the connection, making sure the JAR files correlated the local Java environment, etc, it doesn’t further the analysis to belabor the point. In the end, the Microsoft JDBC Driver was used (Microsoft Community, 2019) and connected the SQL Server database running on the local system to Denodo (see Figure 8 JDBC Connection).

Figure 8 JDBC Connection

After a JDBC data source is imported, in order to extrapolate any data, one must create a base view. Figure 9 is a screenshot of the imported JDBC data source, where Denodo permits new data sources to integrate into Denodo with a query or with table selection (Denodo Community, n.d.). After a base view is created, Denodo will automatically map associations like foreign keys and users can add or alter the associations (see Figure 10). Finally, users can create derived views including data from its own base view(s), others, or both.

Figure 9 JDBC Base View

## Connecting API Data Sources: Interpolation Variables & Flattening

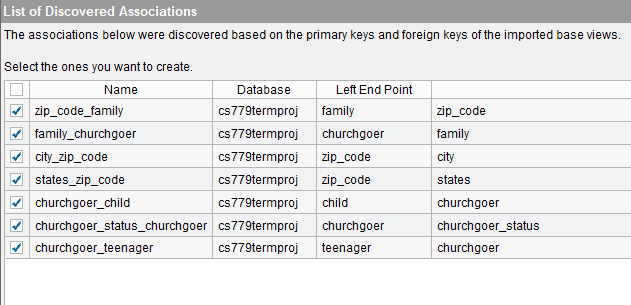
The API data sources explored functioned differently, one required parameters be passed into the API call, and the other simply returning a result set from the call. In the former scenario, Denodo allows interpolation variables, which allows parameters to be determined when an actual call is being made, as opposed to hard coding different parameters for each API call (Denodo Community, n.d.). In the latter, the data source is imported as a JSON array and is then “flattened” using a Denodo wrapper and executed in the GUI or in the VQ Shell with Denodo. This permits data access at the table level, rather than working with arrays.

Figure 10 Associations

Figure 11 (left) is an example snippet code that was used to create the JSON data source in Denodo, Figure 12 (below) is the creation of the HTTP connection to the source and how the API call was made. Note that the JSON wrapper for this data source is in Figure 5.

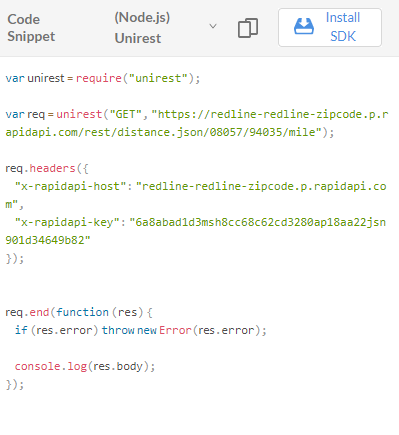


Figure 11 (Rapid API Developer Dashboard, 2020) code snippet from API source

Certain data sources need to be flattened before data can be extrapolated. The WIC\_Vendor source in the submitted files contains location information for all current WIC vendors in the state of Connecticut, that is, retailers that accept vouchers from the government for supplemental nutritional needs for Women, Infants, and Children (WIC) (WIC Authorized Vendors, n.d.). The data set is imported as a JSON array, which is flattened in Figure 13 below.

Figure 12 (JSON HTTP Connection Initiation)

Figure 13 (Flatten Preview)

## Creating Derived Views

Derived views are where the abstraction layer becomes incredibly powerful. Figure 14 below illustrates the joining of an array of WIC Retailers, a distance API calculator, and relational data from multiple tables. This particular query ended up being inefficient (see discussion in Data-Specific Knowledge is Key) but is an excellent illustration of the automated, real-time data transformation and integration that Denodo is capable of. Querying data can happen in a base view, derived view, or in the VQL Shell (which allows specialized SQL language comparable to major RDBMS) (Denodo Community, n.d.).

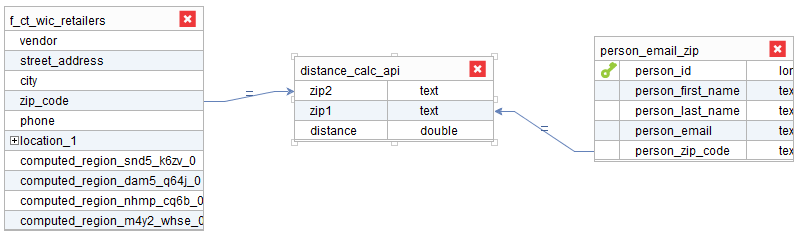


Figure 14 Derived View

# Query Performance Analysis

## Denodo System & Query Performance

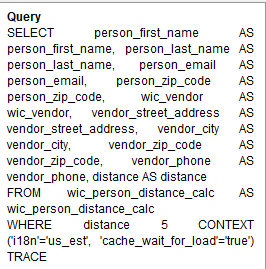
Much of time dedicating to the implementation of this initiative was spent figuring out the configurations, learning how to tune queries in Denodo, understanding differences in ways to access data, and the query chosen was selected because of its high compute needs and large row returns. Without using the Denodo cache, the query ran in about 8.5 seconds on average, returning about 9,500 rows. In SQL Server, a similarly built query using the same data sets (with WIC files imported using SQL Server Import Wizard) ran in about 1.5 minutes. Using the Denodo cache, the query consistently finished executing in about 200 ms. Figure 16 below shows the execution time and query plan of the original query written before indexing, while Figure 17 shows an approximate decrease in 3 seconds of query execution time. Perhaps the most significant advantage to Denodo’s implementation, aside from its faster execution, is that once the data sources are configured in Denodo, they do not need tend to need to be manipulated after that. There is no replication, no ETL, the data is accessed and transformed in real time.

Figure 15 Denodo Query

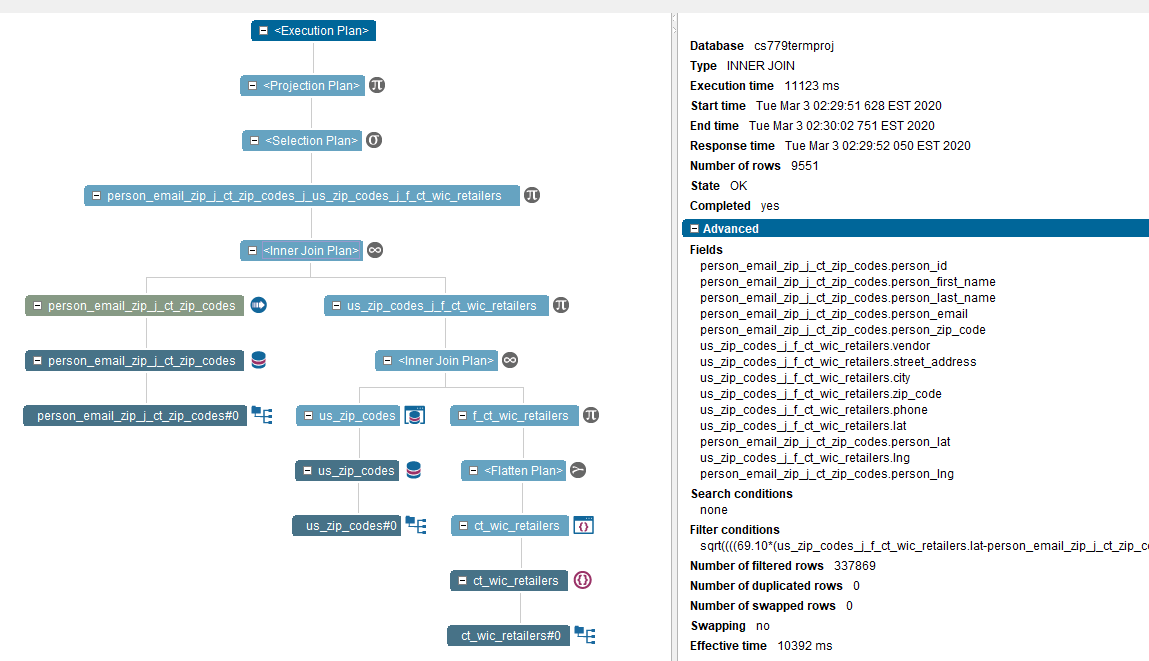


Figure 16 Before Tuning

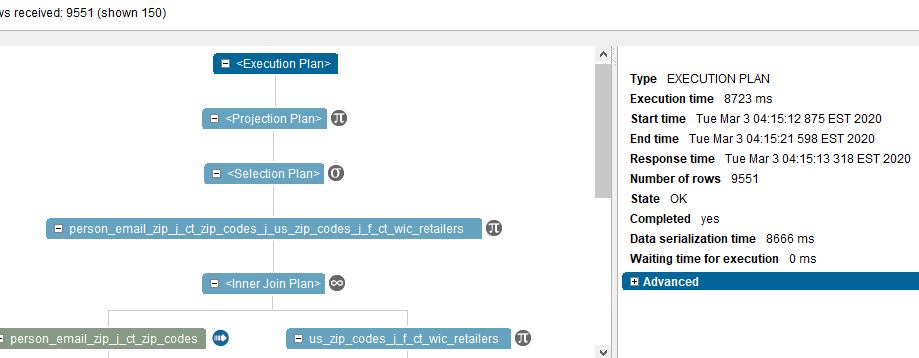


Figure 17 After Tuning

## SQL Server System & Query Performance

SQL Server, even when all sources were imported directly into the database, struggled to return the query in a reasonable amount of time, despite tuning efforts and extra data filtering. Figure 18 below shows SQL Server estimated execution plan and its’ query execution time of just over one and a half minutes. If ChMS were to utilize solely SQL Server to run this query, schema changes including denormalizing the zip code values into the Churchgoer table or querying a smaller subset of data. The script from the SQL Server query is located below.

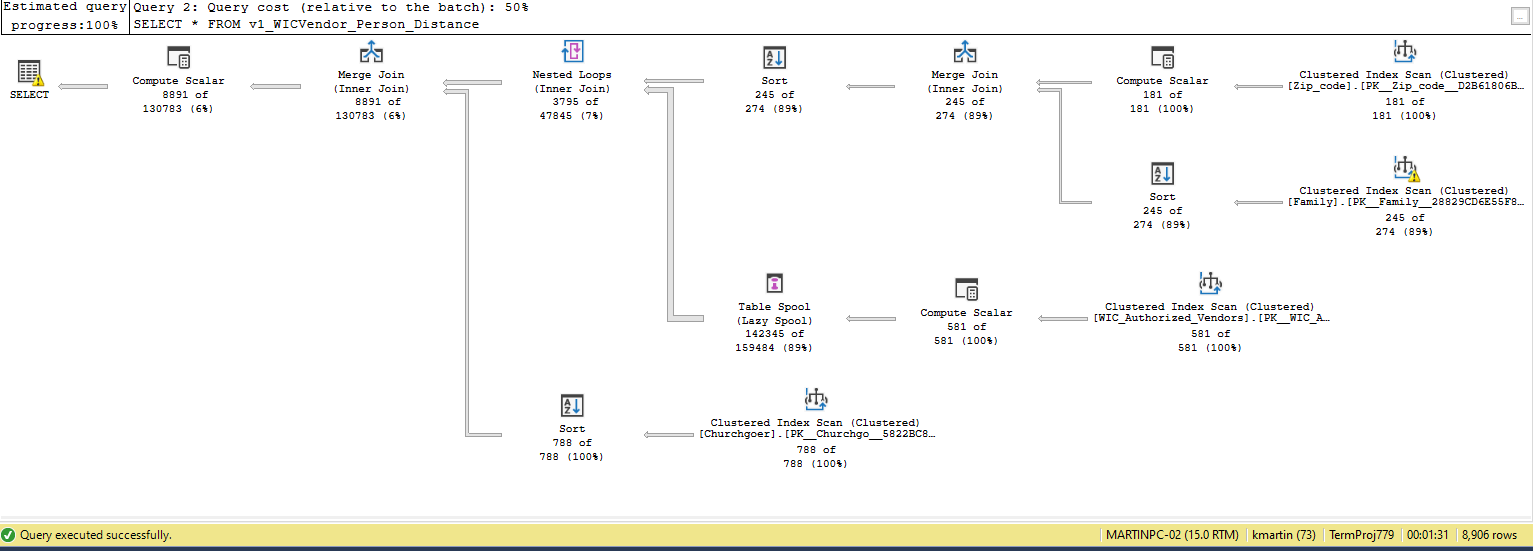


Figure 18 SQL Server Query Execution Plan

### SQL Server Query

--function to calculate distance between two zip codes

CREATE OR ALTER FUNCTION f\_distance\_math (@ZipCode1 decimal(5,0),@ZipCode2 decimal(5,0))

RETURNS decimal(12,2)

AS

BEGIN

--vars for getting lat/lng

DECLARE @Lat1 decimal(9,6),@Lng1 decimal(9,6), @Lat2 decimal(9,6), @Lng2 decimal(9,6);

--vars for arithmetic calc

DECLARE @X decimal,@Y decimal,@distance decimal (12,2);

--get lats/lngs

SET @Lat1 = (SELECT usz.lat FROM US\_Zip\_codes usz WHERE usz.zip = @ZipCode1);

SET @Lng1 = (SELECT usz.lng FROM US\_Zip\_codes usz WHERE usz.zip = @ZipCode1);

SET @Lat2 = (SELECT usz.lat FROM US\_Zip\_codes usz WHERE usz.zip = @ZipCode2);

SET @Lng2 = (SELECT usz.lng FROM US\_Zip\_codes usz WHERE usz.zip = @ZipCode2);

--calculations

SET @X = (69.1 \* (@Lat2 - @Lat1));

SET @Y = (69.1 \* (@Lng2 - @Lng1) \* cos(@Lat1/57.3));

SET @distance = SQRT(@X \* @X + @Y \* @Y);

RETURN @distance

END

GO

--transformation of WIC data

ALTER TABLE WIC\_Authorized\_Vendors

ADD vendor\_id decimal NOT NULL DEFAULT 0;

--add PK sequence

CREATE SEQUENCE WIC\_PK

START WITH 1

INCREMENT BY 1;

--add PK vals

UPDATE WIC\_Authorized\_Vendors

SET vendor\_id = NEXT VALUE FOR WIC\_PK

ALTER TABLE WIC\_Authorized\_Vendors

ADD PRIMARY KEY (vendor\_id);

GO

--get WIC vendors within radius of person's home zip

GO

CREATE OR ALTER VIEW v1\_WICVendor\_Person\_Distance

AS

--fields to return

SELECT p1.first\_name,p1.last\_name,p1.email,wv.Vendor AS [WIC Authorized Vendor],wv.[Street Address],wv.City,

RIGHT('00000'+CAST(wv.[Zip Code] AS varchar(5)),5) AS [Zip Code],wv.Phone,

dbo.f\_distance\_math(p1.zip\_code,CAST(wv.[Zip Code] AS decimal)) AS distance

--subquery to get person in CT w/ email address)

FROM (SELECT p0.first\_name,p0.last\_name,p0.email, z.zip\_code

FROM dbo.Churchgoer p0

JOIN dbo.Family f

ON (p0.family\_id = f.family\_id)

JOIN dbo.Zip\_code z

ON (f.address\_id = z.zip\_id)

--only persons in CT

WHERE z.state\_id = 7

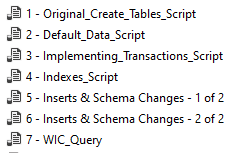
AND p0.email IS NOT NULL) AS p1

--cross join for all options

CROSS JOIN dbo.WIC\_Authorized\_Vendors wv

--filter by distance

WHERE dbo.f\_distance\_math(p1.zip\_code,wv.[Zip Code]) < 5;

NOTE: Numbered SQL Scripts needed to execute this with a similar result set are in the submitted files and are titled as follows:

# Analysis & Observed Outcomes

## Data-Specific Knowledge is Key

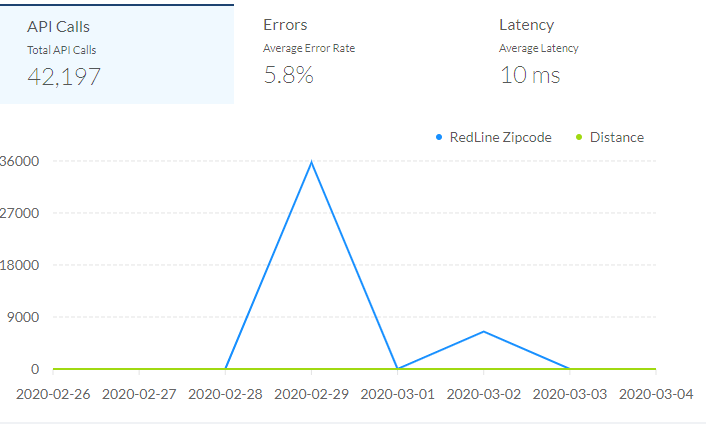
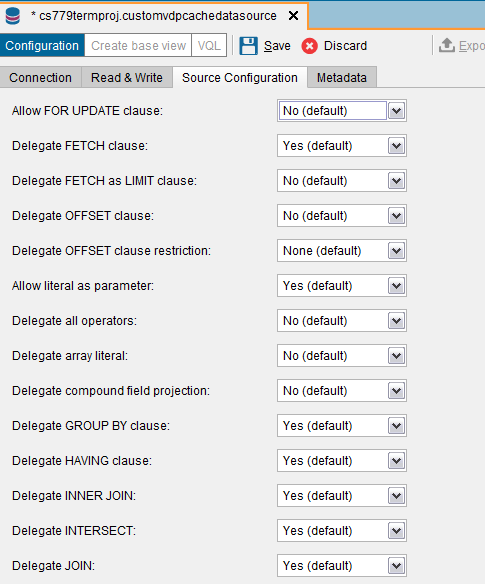
It is common knowledge in database management that understanding one’s data is key to efficient database management, and in a virtualized environment, this is particularly true. For example, in the original design of the WIC retailer query described herein, the initial configuration utilized an API to retrieve the distance between two points. Using an API for a distance calculator seemed an excellent idea in theory, removing any coding needed and off-loading the processing from Denodo’s JVM, but API calls ended up being expensive (relative to performance and actual cost) and were not more accurate than a distance calculator in Denodo. Because of the nature of the API, processing could not be delegated to Denodo (as other sources may) (Denodo Community, n.d.). The query created with the great circle distance calculator formula from Wayana Software (2015) within Denodo performed much more efficiently, querying the full test result set in about 8 seconds and querying a filtered subset in under 200 ms. The query using the distance calculator API took about 10 seconds to run when using that same filtered subset, and timed out on an attempted query of the full test result set. The distance calculator is one small example of the importance of knowing and understanding the data being utilized in a system. Understanding how data is coalesced, its schema, its subject matter, etc. are all vitally important in a virtualized system.

Figure 19 (Rapid API Developer Dashboard, 2020) shows the API calls made by Denodo for distance calculations during testing

## Cache is King: The Importance of Configuring Denodo Administrative Tools

Administrators should be sure to properly configure cache, indexes, statistics, user roles (not available in Denodo Express), and Version Control Systems (not available in Denodo Express) to optimize Denodo’s capabilities. The ability to store queried data in a cache results in fewer I/Os and API calls. Additionally, the options for cache indexes (either created on the cache or imported from the view) should be considered on a case by case basis (Denodo Community, n.d.).

## 

Figure 20 Cache Options

## Alternatives to Data Virtualization



### Data Lakes

Data lakes can be an alternative to data virtualization, where “the data lake acts as…an operational data store…replication or streaming technology is used to keep the data lake up to date” (van der Lans, Modernizing Data Architectures for a Digital Age Using Data Virtualization, 2019). However, this method is not well-equipped to access real-time data. However, when used in conjunction with Denodo, the data lake can serve as a central cloud-based repository of data that is transformed in real-time on an as needed basis (van der Lans, Data Virtualization:Selected Writings, 2019).

### Apache Spark

Like data virtualization, Spark presents a single logical database drawing from heterogeneous sources, and applications are not required to use a specific language. However, two key differentiators between Denodo and Spark are Denodo’s capabilities of *query push down* (where query processing is “pushed down” to the data source) and *distributed join optimization* (parallel processing and injection joins, ship joins, and parallel joins) (Van Der Lans, 2019). Although this report did not examine the validity of the claim, it stands to reason that the less data that has to be actually loaded into memory for queries, the better.

# Conclusion

This analysis sought to illustrate that, with data virtualization, ChMS clients would be able to access external data sources to facilitate access to community resources in a more efficient manner than in its current application environment. Although this implementation was merely a “proof of concept,” the research presented and evidence brought suggested that Denodo, or more generally data virtualization, offers an efficient option for business users that have increasingly complex and unmanageable data architectures. This implementation would likely benefit from more extensive testing with more big data oriented sources to determine if it would actually present a benefit for the ChMS user group. In conclusion, as we continue to see rising complexity of data architectures, data virtualization will pay an increasingly vital role in ensuring users can meet their data needs.

# References

Blomberg, B. (2015, June 11). *The Pros and Cons of Database Virtualization*. Retrieved from The Data Recovery Blog by Ontrack: https://www.ontrack.com/blog/2015/06/11/pros-cons-database-virtualization/

Denodo Community. (n.d.). *General Architecture*. Retrieved from Denodo.com: https://www.denodo.com/en/data-virtualization/how-it-works

Denodo Community. (n.d.). *Hardware Requirements*. Retrieved from Denodo Platform Installataion Guide: https://community.denodo.com/docs/html/browse/7.0/platform/installation/preinstallation\_tasks/hardware\_requirements/hardware\_requirements

Denodo Solutions. (n.d.). *Solutions By Use Case: Big Data*. Retrieved from Denodo.com: https://www.denodo.com/en/solutions/by-use-case/big-data

Engel, M. (2019, February 20). *Use Big Data to Engage Your People and Grow Your Church*. Retrieved from Gloo Blog: https://blog.gloo.us/big-data-for-churches-introduction

Martin, K. (2018, October 15). Church Management System: CS669 Term Project.

Microsoft Community. (2019, August 12). *Microsoft JDBC Driver for SQL Server*. Retrieved from Microsoft SQL Docs: https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15

Polnar, J., & Schudy, R. (2019). Module 4. *Advanced Database Management (MET CS 779)*. Boston, Massachusetts: Boston University Metropolitan College.

Pushpay. (2018, July 30). *The Crucial Church Metric You’re (Probably) Not Measuring—But Should!* Retrieved February 22, 2020, from Pushpay: https://www.vanderbloemen.com/blog/church-metric-not-measuring

*Rapid API Developer Dashboard*. (2020, March 4). Retrieved from Rapid API: https://rapidapi.com/developer/dashboard

van der Lans, R. (2019). *Data Virtualization:Selected Writings.* Lulu.com.

van der Lans, R. (2019, October). *Modernizing Data Architectures for a Digital Age Using Data Virtualization.* Retrieved from R20/Consultancy: https://www.denodo.com/en/document/analyst-report/modernizing-data-architectures-digital-age-using-data-virtualization

Van Der Lans, R. F. (2019, October 24). *Spark and Data Virtualization: Competitors or Cooperators?* Retrieved from Data Virtualization Blog: http://www.datavirtualizationblog.com/spark-and-data-virtualization-competitors-or-cooperators/

Wayana Software. (2015). *Great Circle Distance Calculator*. Retrieved from Geosats.com: http://www.geosats.com/0x2circ.html

*WIC Authorized Vendors*. (n.d.). Retrieved from Connecticut Data: https://data.ct.gov/Health-and-Human-Services/WIC-Authorized-Vendors/jk32-cd4i