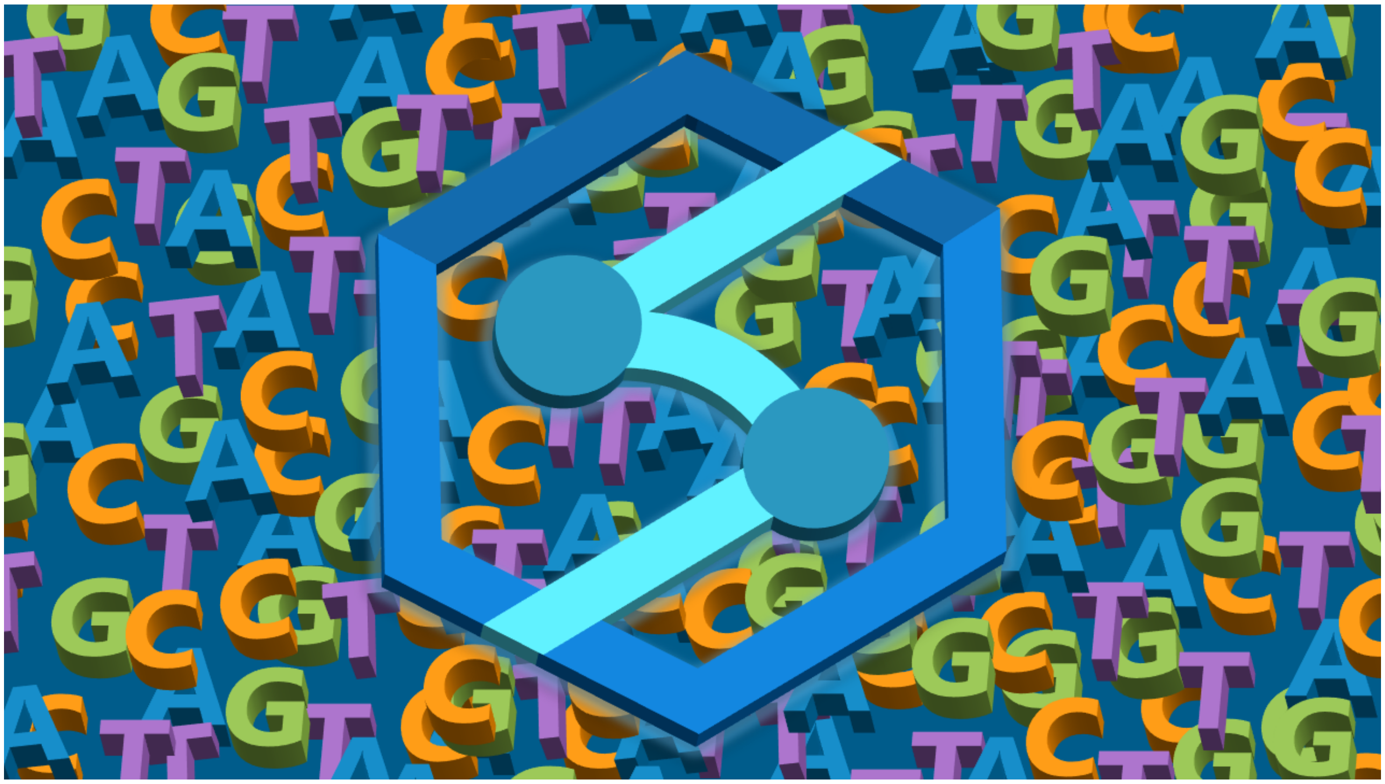
Query Millions of Genomic Variants At-Scale using Azure Synapse

Colby Ford

One struggle for genomics research is the ability to analyze the vast amounts of data in an efficient way. Previously, this would have been performed using large, on-premise high performance computing (HPC) cluster jobs. Today, the cloud offers us opportunities to perform bioinformatics analyses interactively and at-scale.

In this demo (see the video at the end of this post), I'll be showing how Azure Synapse can be used to analyze millions of variants quickly using basic structured query language (SQL) commands. I retrieved >80 million variant records from over 2,500 individuals from the 1000 Genomes Project (Phase 3 release). In its variant call format (VCF) form, this data is about 168GB in size, which encompasses all 22 autosomes, X and Y sex chromosomes, and mitochondrial variants.

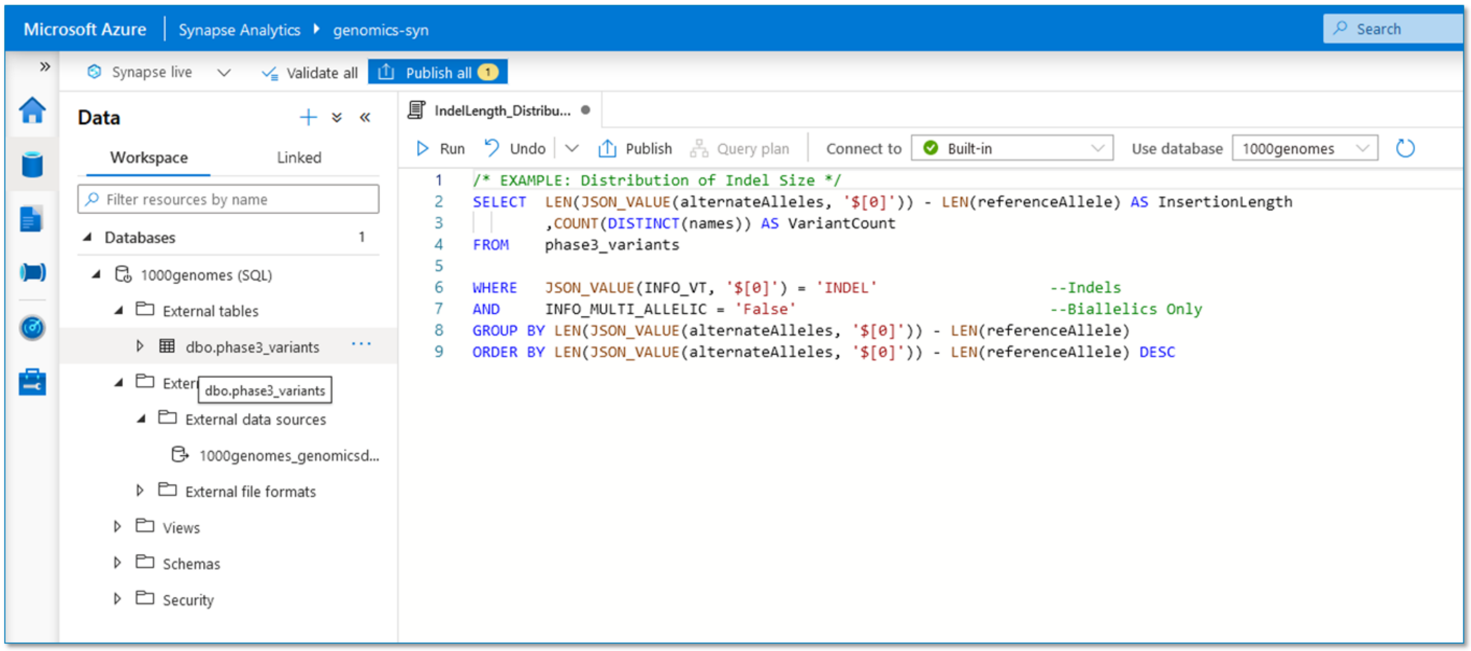


What's Azure Synapse?

Azure Synapse Analytics is a recent amalgamation/rebranding of some services in Azure, including Azure SQL Data Warehouse with connectors to Power BI and other services. Synapse includes serverless pool, dedicated SQL pool, and Apache Spark pool options for flexible and scalable data workloads in Azure.

Learn more about Azure Synapse Analytics here.

This service isn't marketed for any specific industry, but its immense scalability makes it a perfect tool for browsing tons of genomics data. All we need to do is get our data in a format that's usable in Synapse.



```sql
/* EXAMPLE: Distribution of Indel Size */
SELECT  LEN(JSON_VALUE(alternateAlleles, '$[0]')) - LEN(referenceAllele) AS InsertionLength
        ,COUNT(DISTINCT(names)) AS VariantCount
FROM    phase3_variants

WHERE   JSON_VALUE(INFO_VT, '$[0]') = 'INDEL'                   --Indels
AND     INFO_MULTI_ALLELIC = 'False'                            --Biallelics Only
GROUP BY LEN(JSON_VALUE(alternateAlleles, '$[0]')) - LEN(referenceAllele)
ORDER BY LEN(JSON_VALUE(alternateAlleles, '$[0]')) - LEN(referenceAllele) DESC
```

Converting from VCF to Parquet

VCF files, though popular in bioinformatics, are a mixed file type that include a metadata header and a more structured table-like body. Using the Glow package in Apache Spark, we can convert VCF files into the Parquet format, which works excellently in distributed contexts like a Data Lake or in Azure Synapse.

We can perform this conversion at scale in Spark (either in Azure Databricks or Azure Synapse) using only four lines of code (plus two more lines for calculating optional summary statistics). The following code reads a VCF file from a mounted genomics data lake location.

```
input_vcf_path = "/mnt/1000genomes/phase3_vcfs/chr1.vcf.gz"
output_parquet_path = "/mnt/1000genomes/phase3_parquets/chr1.parquet"

vcf_df = spark.read.format("vcf").load(input_vcf_path) \
               .withColumn("hardyweinberg", expr("hardy_weinberg(genotypes)")) \
               .withColumn("stats", expr("call_summary_stats(genotypes)"))

vcf_df.write.format("parquet").save(output_parquet_path)
```

When this VCF file is read into Spark, it converted into a Spark DataFrame. Then, after calculating some optional summary statistics, the cluster will write out multiple partitioned Parquet files back to the data lake. For the 1000 Genomes data I'm using here, the Parquet format reduces the ~168GB of VCF data down to ~74GB.

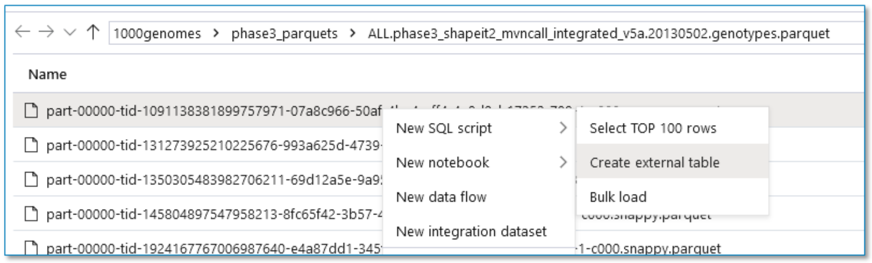
External Tables

In Azure Synapse, external tables are a really awesome capability to connect to data that lives in your data lake. Once you create an external table, you can query the data just as if it were a real table in your database. This works on delimited text files (like CSV or TSV), Hive Orc files, or Parquet.

A snippet of the CREATE EXTERNAL TABLE script for VCF data looks like this:

```
CREATE EXTERNAL TABLE phase3_variants (
        [contigName] varchar(50),
        [start] bigint,
        [end] bigint,
        [names] varchar(1000),
        [referenceAllele] varchar(1000),
        [alternateAlleles] varchar(1000),
        [qual] float,
    ...
        [hardyweinberg] varchar(8000),
        [stats] varchar(8000)
        )
        WITH (
        LOCATION = 'phase3_parquets/ALL.phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.parquet',
        DATA_SOURCE = [1000genomes_genomicsdls_dfs_core_windows_net],
        FILE_FORMAT = [SynapseParquetFormat]
        )
GO
```

While you could write this yourself, Synapse will help you generate scripts automatically. Find the Parquet file(s) that you want to load, right-click it, click New SQL Script, and then click Create external table. This will generate a basic script, which you should edit to fit your specific VCF file needs. (Specifically, make sure the data types are correct.)


1000genomes > phase3_parquets > ALL.phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.parquet
Name
part-00000-tid-1091138381899757971-07a8c966-50af
part-00000-tid-131273925210225676-993a625d-4739
part-00000-tid-1350305483982706211-69d12a5e-9a9
part-00000-tid-145804897547958213-8fc65f42-3b57-4
part-00000-tid-1924167767006987640-e4a87dd1-345
c000.snappy.parquet
-1-c000.snappy.parquet
New SQL script
New notebook
New data flow
New integration dataset
Select TOP 100 rows
Create external table
Bulk load

Speedy Queries

Azure Synapse can perform queries on this data very quickly. In fact, for some of my example queries, it only took a couple minutes to return the results from >80 million variant records. Here are some example queries:

Sample Task	**Query Time (in minutes)**
Find Minor Structural Variants SELECT * FROM phase3_variants WHERE JSON_VALUE(INFO_VT, '$[0]') = 'SV' --Structural variants AND qual > 95 --High quality AND alternateAlleles LIKE '%ALU%' --Transposable elements AND JSON_VALUE(hardyWeinberg, '$.hetFreqHwe') >= 0.05 --Higher heterozygous frequency AND JSON_VALUE(stats, '$.alleleFrequencies[1]') <= 0.05 --Rare minor alleles (variants)	1:11

Calculate Indel Size Distribution
 	
SELECT LEN(JSON_VALUE(alternateAlleles, '$[0]')) - LEN(referenceAllele) AS InsertionLength
        ,COUNT(DISTINCT(names)) AS VariantCount
FROM   phase3_variants
 	
WHERE   JSON_VALUE(INFO_VT, '$[0]') = 'INDEL'   --Indels
AND   INFO_MULTI_ALLELIC = 'False'   --Biallelics Only
GROUP BY LEN(JSON_VALUE(alternateAlleles, '$[0]')) - LEN(referenceAllele)
ORDER BY LEN(JSON_VALUE(alternateAlleles, '$[0]')) - LEN(referenceAllele) DESC
	1:34

Find Motif Matches
 	
DECLARE @motif varchar(1000)
Set   @motif = 'TA'
 	
SELECT contigName
        ,[start]
        ,[end]
        ,names
        ,referenceAllele
        ,JSON_VALUE(alternateAlleles, '$[0]') AS alternateAllele
        ,(LEN(JSON_VALUE(alternateAlleles, '$[0]')) - LEN(REPLACE(JSON_VALUE(alternateAlleles, '$[0]'), @motif, ''))) / LEN(@motif) AS MotifMatches
FROM   phase3_variants
 	
WHERE   JSON_VALUE(INFO_VT, '$[0]') = 'INDEL'   --Indels
AND   INFO_MULTI_ALLELIC = 'False'   --Biallelics Only
AND   JSON_VALUE(alternateAlleles, '$[0]') LIKE '%' + @motif + '%'   --TA Matches
	1:49

Note: Processing times may vary. These are based on the serverless built-in pool that comes standard with Azure Synapse. If you create a dedicated SQL pool, you may reduce query times even further.

To learn more about best practices around making dedicated SQL pools, click here.

Resources

To view the full demonstration, check out the video below.

All demo code and sample queries used in this post can be found at: https://github.com/BlueGranite/azure-synapse-vcf-analysis

How can BlueGranite Help?

If you're reading this post, you're probably no stranger to BlueGranite. We have industry experts with experience in both healthcare and life sciences, plus a deep expertise in all things cloud. Contact us today to find out how we can help you with your data and analytics solutions.

Also, we recently created a Genomics-centric page to highlight solutions and technologies that are specific to genomics.