Dynamic Graph Model for Consistent Data Integration and Genome Variants

Bilal Arshad and Sanna Aizad

University of Derby – Department of Engineering and Technology

UK Systems Research Challenges Seminar

December 16th 2020

Data Integration – Issues!

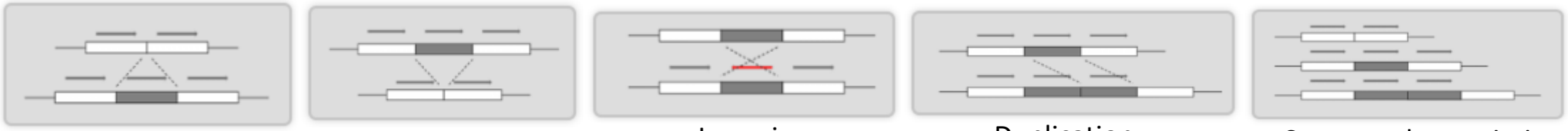
- Clinical data is distributed in disparate silos in various data formats
  - XML, CSV, SQL etc.
- In a distributed system clinical data needs to be brought together into a centralised warehouse for storage and reporting
- With various data sources involved structural consistency of data is an issue
- Genome variations
  - Differences in DNA between individuals within a population.

    Reference Genome: ACTGGTGACGTGCGAGCG
    Genome Variation: ACTGCTGACGAGCGTCGG

  - Variations represent mutations in germ cells and somatic cells.
  - The term *"**variant**"* is used for a section of the genome which differs between two genomes.
  - Different versions of the same variant are called *"**alleles**"*.

Types of clinical and genetic variations

- Structural variations (occurring over a period of time)



Insertion | Deletion | Inversion | Duplication | Copy number variation

- Single base-pair substitutions also known as Single Nucledotide Polymorphisms or SNPs.

ACTGTGACGTGCGG
ACTGTGACGAGCGG

- Insertions and Deletions (indels)

Reference	ACTGACGCATGCATCATGCATGC	
Insertion	ACTGACGCATGGTACATCATGCATGC	Indel
Deletion	ACTGACG--TGCATCATGCATGC	

Solution – Graphs!

Provide easy to use and intuitive graph models to support heterogeneous set of data

The graphs can be queried iteratively

Flexible and cost effective (in terms of computation)

Graphs allow efficient and iterative analytics

Graphs can be effectively optimised for efficient processing of interrelated datasets

Graph Model

Patient (PT)

pid	condition	pname
p01	cancer	Bill
p02	TB	Chris

Condition (CO)

pid	condition	gene_no	clinical_sig	last_reviewed
p01	cancer	GRCh11	X1	04.05.16
p02	TB	GRCh12	y1	09.03.12

Genes (GE)

gene_no	allele_id	GRCH37	GRCH38	attribute_name
GRCh11	314	TG	G	click
GRCh12	642	AD	A	min

Review Status (RS)

criteria	condition	pid	entries_submit	conflict
2D	cancer	p01	X1	yes
N1	TB	p02	y1	no

Allele (AL)

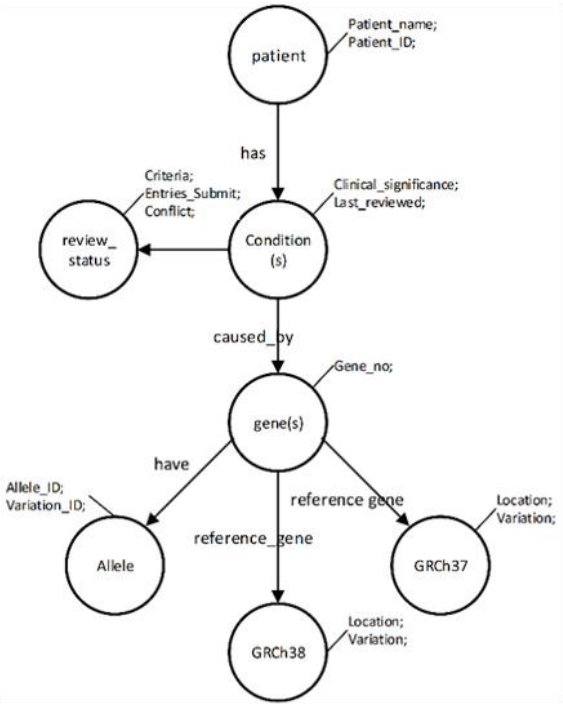
allele_id	gene_no	variation_id	attribute_name
314	GRCH11	111	DAG
642	GRCh12	211	FOW

Reference_GRCH37(GR37)

location_GRCH37	variation_GRCh37
3	T
8	G

Reference_GRCH38(GR38)

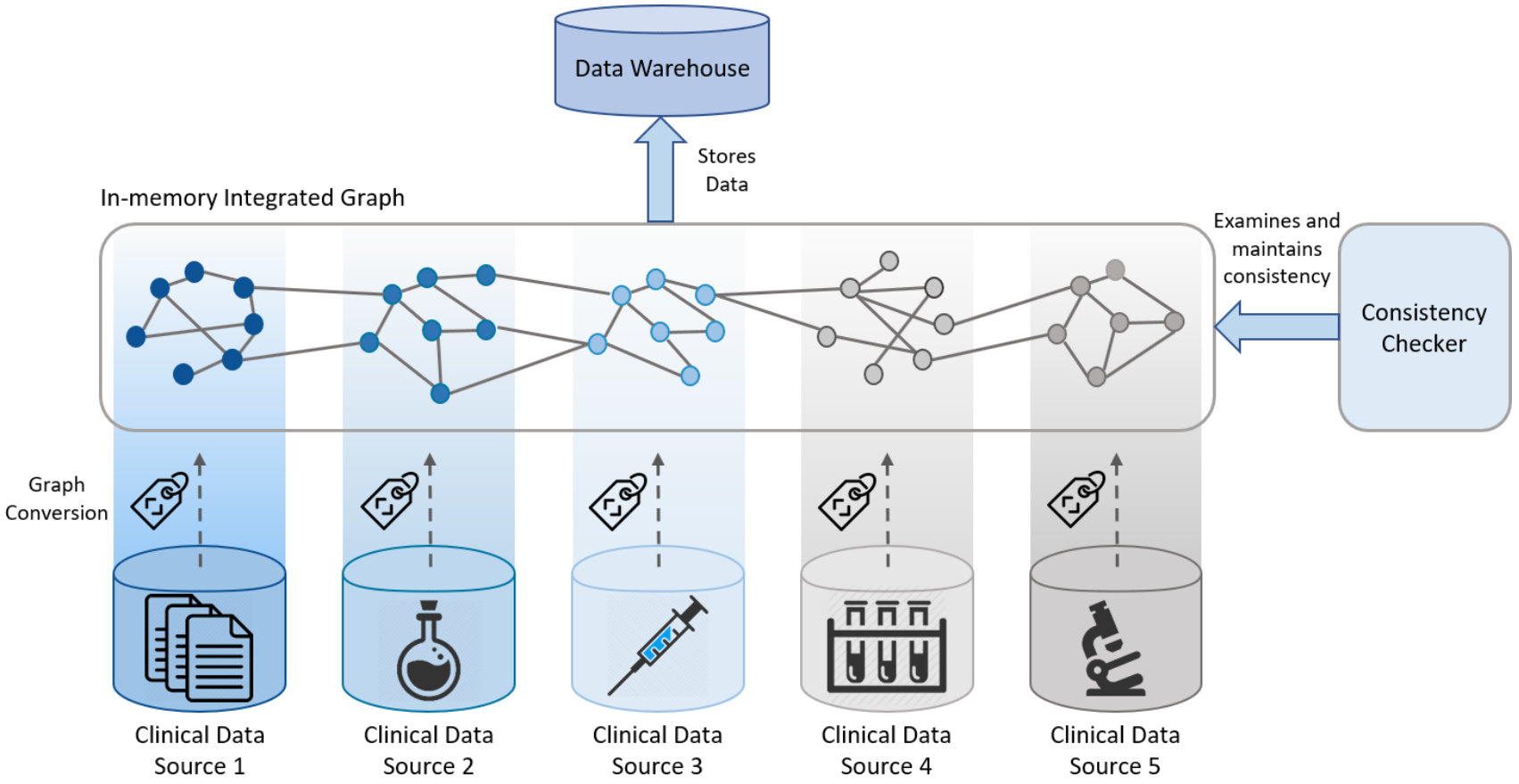
location_GRCH38	variation_GRCh38
5	G
20	A



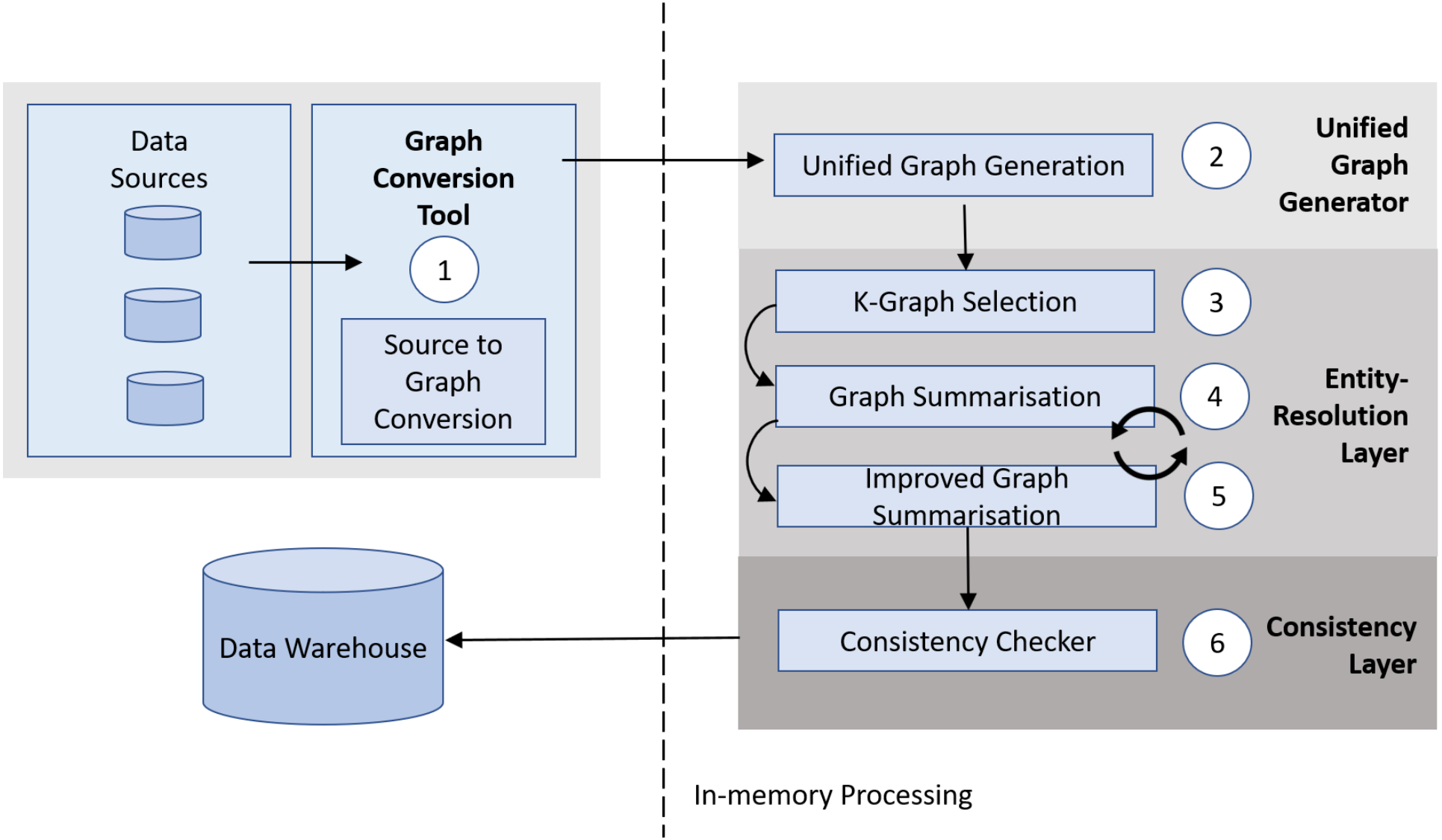
The data within the source file is mapped onto a graph model. The nodes capture the different types of the data present within the file, whereas the edges capture the relationships between the nodes and hold biological significance.

Consistency in evolving sources

- Given a native graph environment, it becomes easy to manipulate the graph model.
- The nodes and edges can be updated and changed using Cypher queries.
- The graph model can be accessed and updated iteratively.
- It also becomes easier to pull out information based on the knowledge of the position (particularly useful in terms of studying genes, where the gene positions are known)



System Architecture

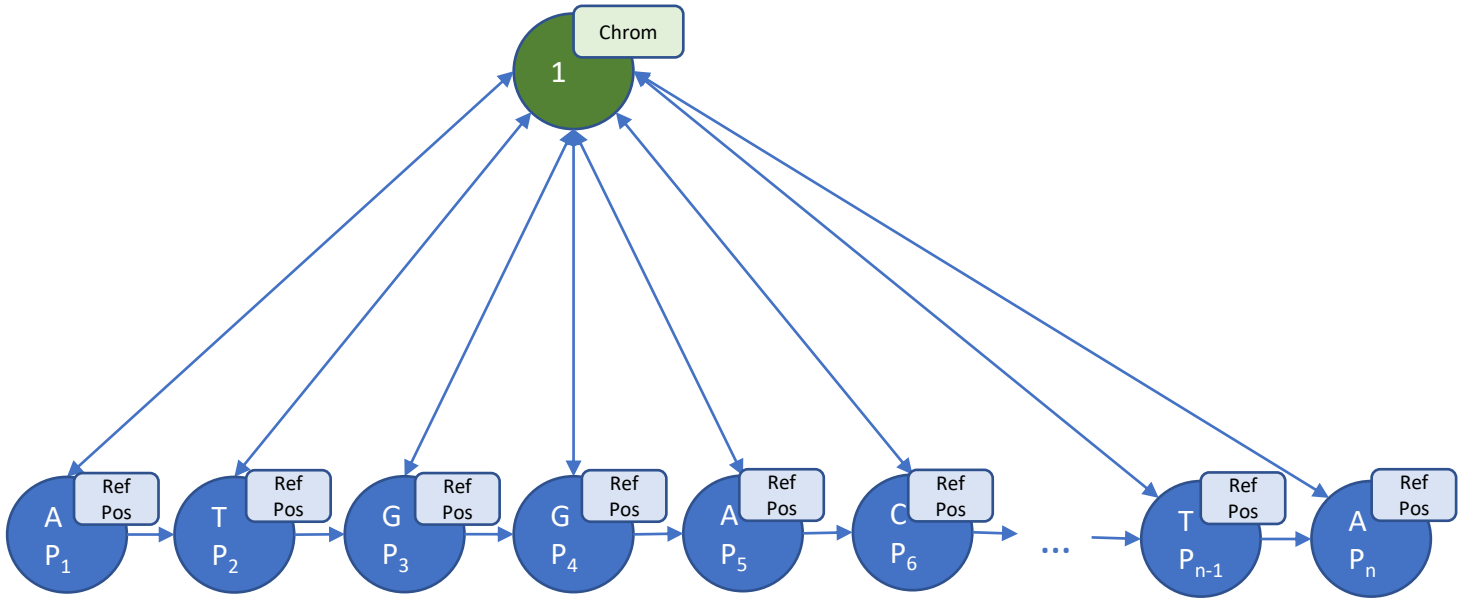


VCF Graph Model: Reference Genome

>Reference Genome

ATGGACTGAGGAGAGTG
AGTAGCGCATTGTCGTATA
TATCGTGCTAGCTAGCTGA
TGCCAGAGTGCTAGTCGA
TCGTTGTGCATGTCAGTAC
GACACAAAACGGCTAGTC
GTCGTCGTCGTCATAGTAC
GTAGTGCTGATAGTTCATG
ACTGCTCTCAGTA...

FASTA file



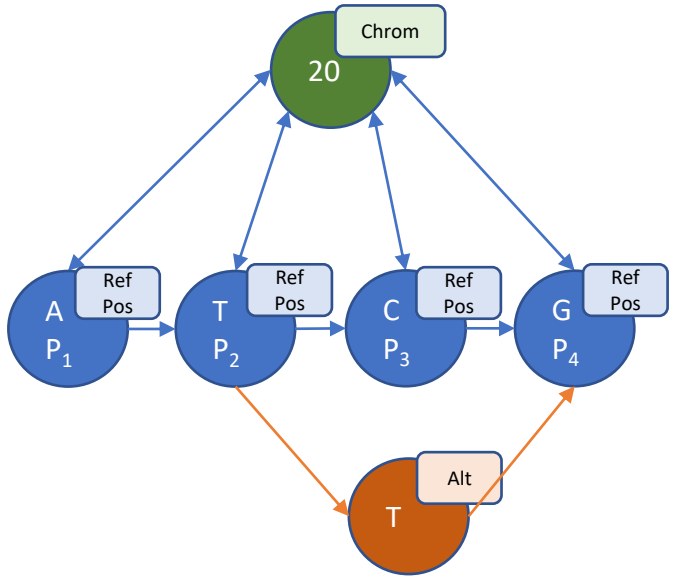
VCF Graph Model: Variations

- Substitution

  a nucleotide base is replaced by another.

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO
20	3	.	C	G	.	PASS	DP=100

Position:	P_1	P_2	P_3	P_4
Allele 1:	A	T	C	C
Allele 2:	A	T	T	C



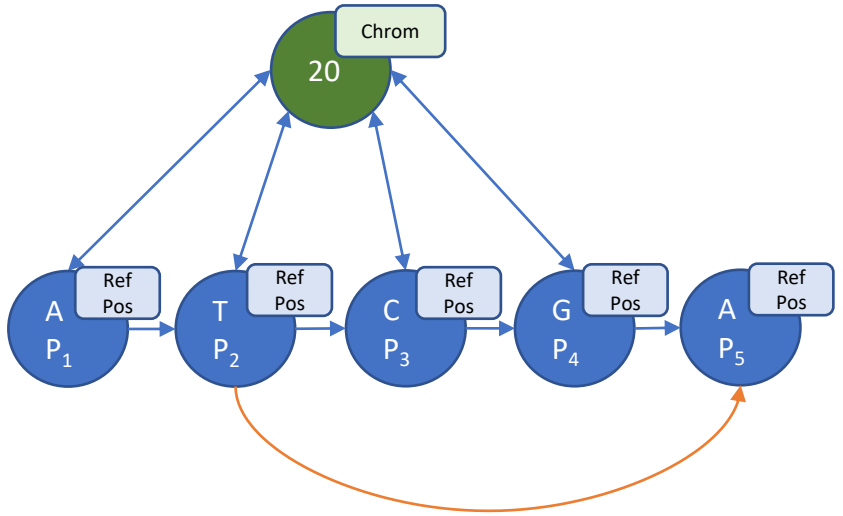
VCF Graph Model: Variations

- Deletion

  a nucleotide base is removed altogether.

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO
20	3	.	C	G	.	PASS	DP=100
20	2	.	TC	T	.	PASS	DP=100

Position:	P_1	P_2	P_3	P_4	P_5
Allele 1:	A	T	C	G	A
Allele 2:	A	T	-	-	A

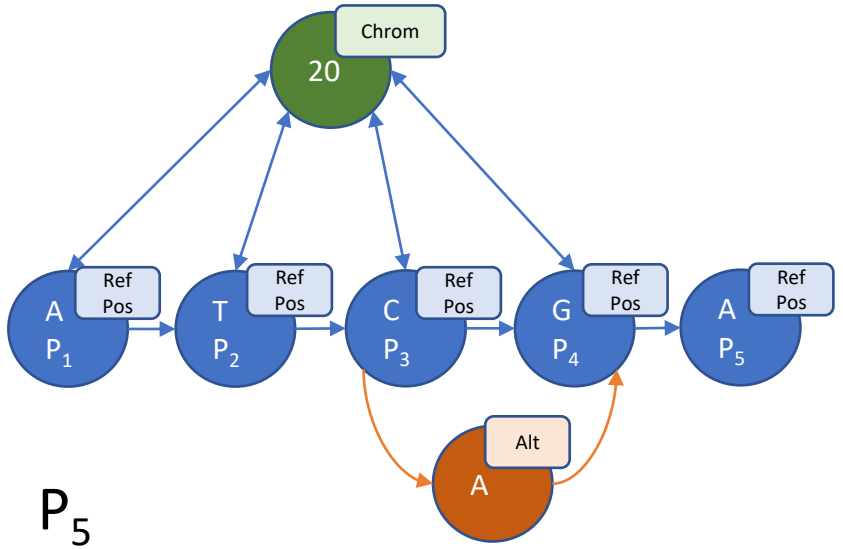


VCF Graph Model: Variations

- Insertion

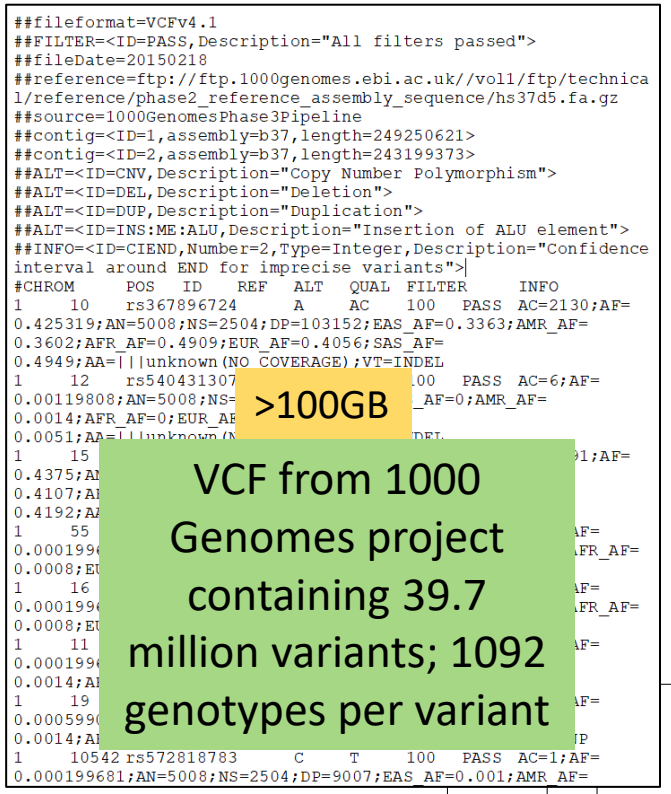
  an extra nucleotide base is added.

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO
20	3	.	C	G	.	PASS	DP=100
20	2	.	TC	T	.	PASS	DP=100
20	2	.	TC	TCA	.	PASS	DP=100



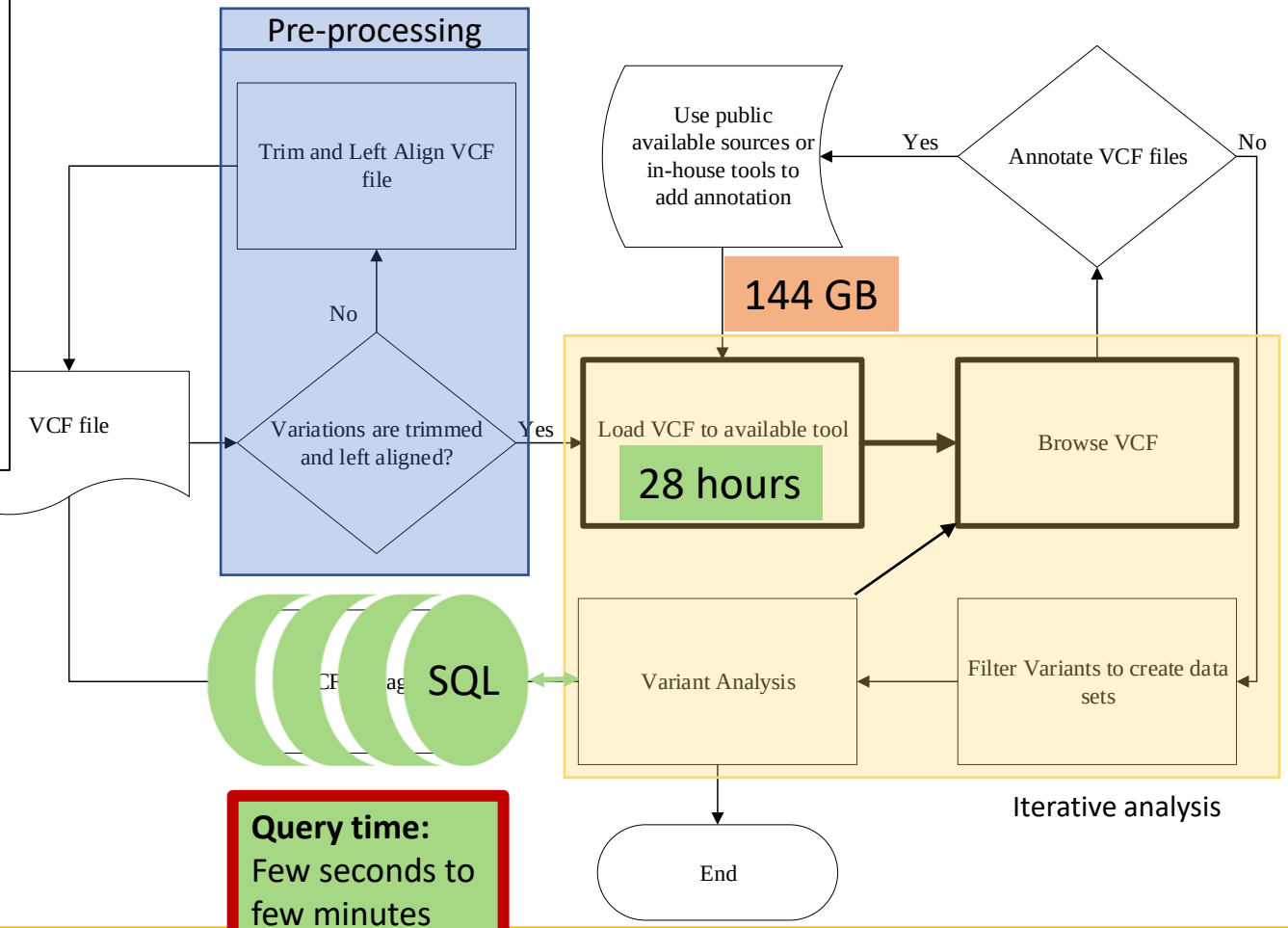
Position:	P_1	P_2	P_3		P_4	P_5
Allele 1:	A	T	C	-	G	A
Allele 2:	A	T	C	A	G	A

Variant Analysis Pipeline



>100GB

VCF from 1000 Genomes project containing 39.7 million variants; 1092 genotypes per variant



Query time: Few seconds to few minutes

GEMINI

SnpEff

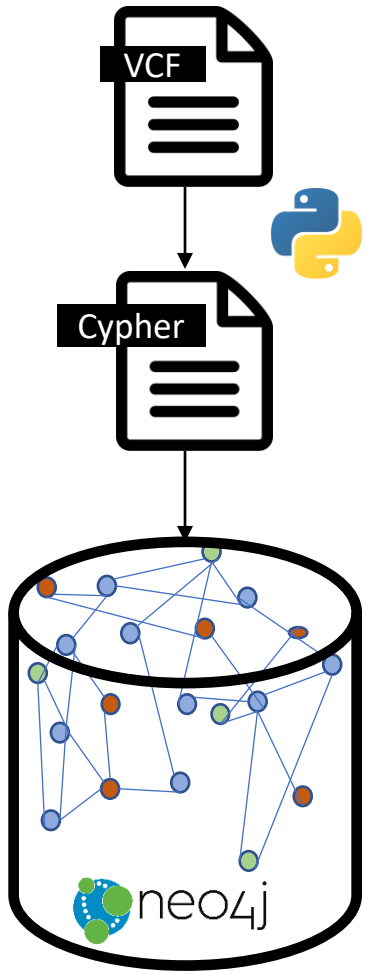
Implementation

Nodes from Reference Genome:
CREATE (cr20:CHROM {id:20})
CREATE (p2:POSRES {p:10657, r: 'G'})
CREATE (p3:POSRES {p:10966, r:'A'})
CREATE (p4:POSRES {p:10967, r:'C'})

Nodes from VCF:
CREATE (a1:ALT {a:'G'})
CREATE (a2:ALT {a:'T'})

Edges from Reference Genome:
CREATE (cr20)-[:HAS]->(p2)
CREATE (cr20)-[:HAS]->(p3)
CREATE (cr20)-[:HAS]->(p4)
CREATE (p2)-[:TO]->(p3)
CREATE (p3)-[:TO]->(p4)

Edges from VCF:
MODIFY (p2)-[:ALTPATH]->(a1)
MODIFY (p2)-[:ALTPATH]->(a2)
MODIFY (a1)-[:ALTPATH]->(p4)
MODIFY (a2)-[:ALTPATH]->(p4)



Algorithm 1: FASTA to Reference Genome Graph Model

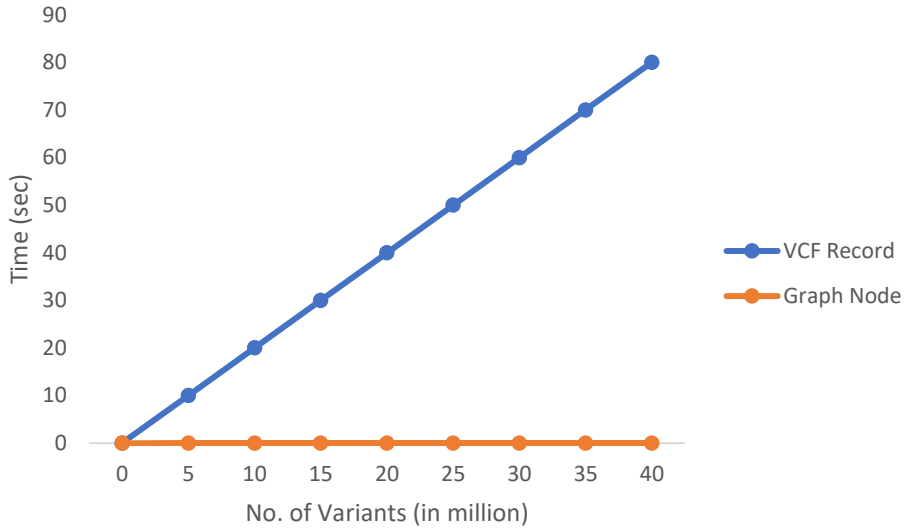
Input: FASTA file
Output: Nodes and edges in Graph Query Language
create chromosome node c;
foreach *char* m *in FASTA file* **do**
  find position p;
  create position node n;
  add attributes position p, nucleotide m to node n;
  create edge from n to c;
  if $n+1$ *is not last node* **then**
    create edge from n to $n+1$;

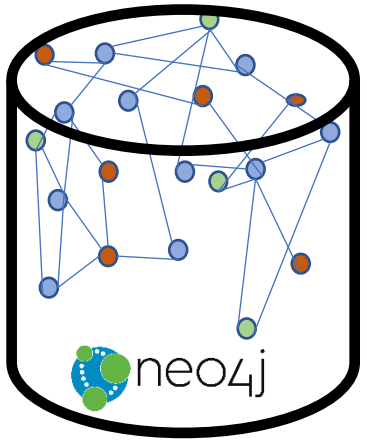
Algorithm 2: VCF to Variation Graph Model

Input: VCF file, Reference Genome Graph Model
Output: Nodes and Edges in Graph Query Language
foreach *record* m *in VCF* **do**
  find chromosome c;
  find position p;
  modify position node p by creating an edge to the alt node a;
  add altNucleotide an to node a;
  create edge from alt node a to position node $pEnd$;
  create edge from alt node a to vcfRecord node v;
  create edge from vcfRecord node v to header node h;
  create edge from header node h to headerMeta node hm;

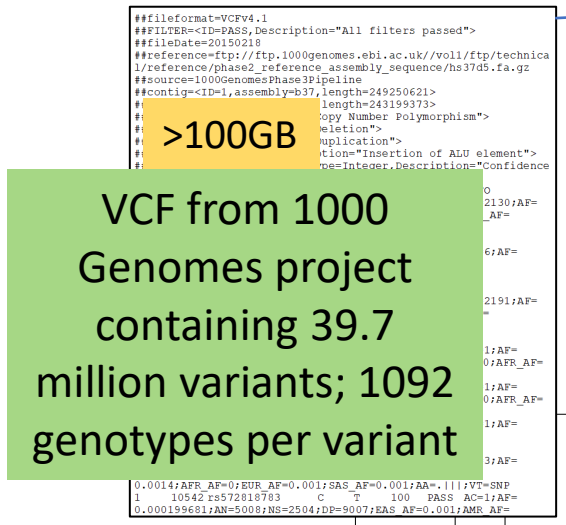
Browsing

	1000 Genomes 1,092 Samples Runtime	
	GEMINI	Neo4j
Return all novel variants`select * from variants where in_dbsnp = 0`	11 sec.	4 sec
Return all loss-of-function variants`select * from variants where is_lof = 1`	177 sec.	10 sec
Return all rare, loss-of-function variants`select * from variants where is_lof = 1 and aaf <0.01`	152 sec.	8 sec
Return all loss-of-function variants and filter on a specific sample's genotype.`select * from variants where is_lof = 1 --gt-filter "gt_types.NA12878 = = HET"` (trio)`--gt-filter "gt_types.NA20814 = = HET"` (1092)	194 sec.	2 sec



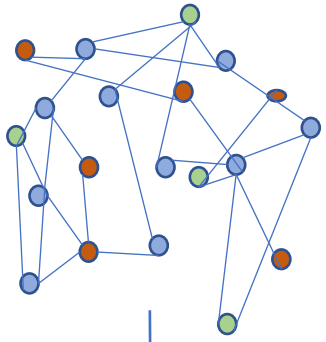


Results

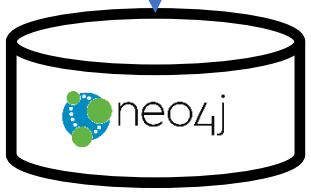


>100GB

VCF from 1000 Genomes project containing 39.7 million variants; 1092 genotypes per variant

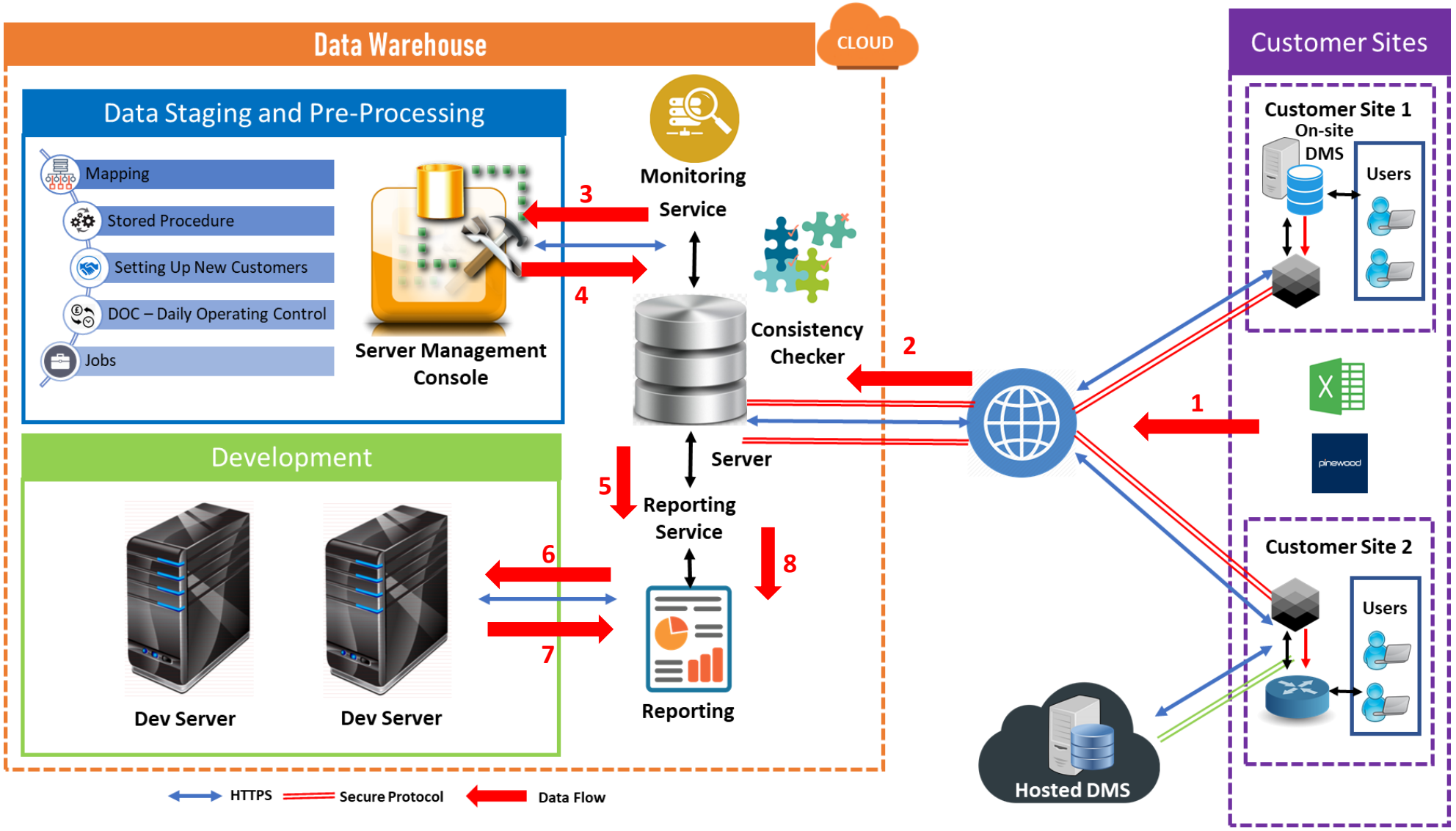


32 hours



Query time: 2ms

Another use-case – DMS (Dealer Management System)



Conclusion

- All underlying approaches in data integration settle on consistency
- Due to the iterative nature of analysis, the clinical datasets need to be parsed several times to extract the desired information
- In order to improve the query and processing times, we introduced a graph model as an alternative data structure
- The data from VCF and source files was converted to nodes and edges in our graph.
- The native graph database, Neo4j was used to construct our graph.
- This reduced our query time significantly as it takes a constant of 2ms to reach any node, no matter how many nodes are present

Future Direction

- Benchmark the system against bigger datasets for further testing
- The graph model will be tweaked to a high performance in-memory environment, which will make the model more efficient.
- New variations could be added to the model, without inserting the graph model to the graph database again
- The use of in-memory and HPC frameworks will make it easy to parse variations which will in turn make analysis faster, less expensive and more accurate
- The approach will be examined in other use-cases to test the efficacy of the approach in other domains

Thank you! Questions?

✉ b.arshad@derby.ac.uk
s.aizad@derby.ac.uk
@bilalarshadali
@sannaaizad