Poster Paper

Proposal of a Database Type and Aggregation Function for Accelerating Medical Genomics Study on RDBMS

Yoshifumi Ujibashi
Fujitsu Laboratories Ltd.
Kawasaki, Japan
ujibashi@jp.fujitsu.com

Motoyuki Kawaba
Fujitsu Laboratories Ltd.
Kawasaki, Japan
kawaba@jp.fujitsu.com

Lilian Harada
Fujitsu Laboratories Ltd.
Kawasaki, Japan
harada.lilian@jp.fujitsu.com

ABSTRACT

Next generation sequencing (NGS) and the recent development of efficient algorithms for genomic analysis are contributing to the understanding of human genetic variation and thus to personalized medicine. Among those genomic analysis, disease-causal gene analysis that finds genes relevant to specific diseases has received much attention. In this paper, we present our work on extending the PostgreSQL open source relational database management system (RDBMS) to efficiently handle genomic analysis. We introduced a new genome data type and a genome type aggregation function that drastically improved the performance of a typical query for disease-causal gene analysis by a factor of 50 to 360.

1. INTRODUCTION

Human beings have a sequence of three billions deoxyribonucleic acid (DNA) molecules which contains some millions of variants called "gene variants" that cause individual variations. Each individual has personal types of gene variants called "genotypes" which are combinations of nucleotide derived from chromosome dipoles. Figure 1 illustrates an example of genotypes. Individual 0 has genotype 'C/C' at gene variant 0, and genotype 'T/C' at gene variant 1. On the other hand, individual 1 has genotype 'A/C' at gene variant 0, and 'T/T' at gene variant 1.

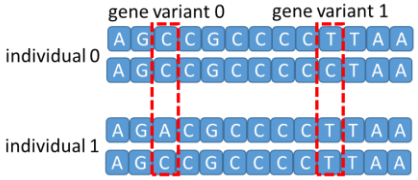

Figure 1: Chromosome dipoles of two individuals

The improvement of processing performance and the reduction of running cost of NGS, and the recent development of efficient algorithms for genomic analysis have resulted in an enormous increase in the amount of data of gene variants like SNP (single nucleotide polymorphism) and INDEL (insertion-deletion polymorphism). These gene variants are leveraged in a variety of studies such as cohort study, inheritance history study and disease-causal gene study. Disease-causal gene study aims to find the genes relevant to specific diseases and clarify the reasons of these diseases. The typical processing in such a study is to first filter the individuals by some patient clinical condition (e.g. case-control), and then, find the genes whose frequency of its genotype is different between the filtered group and the rest. Normally the genetic data is delivered in flat-files (VCF [4]), and the patient information data, for instance, demographic information as gender/race/age, clinical information, and lifestyle information, are usually stored/managed in RDBMS. Recently, some studies integrate the genetic data and the patient data to be managed in RDBMS [1] [2]. Although the RDBMS approach improves the data manageability and usability, the improvement of the procedure and performance of the analysis processing in finding out the genes of interest remains a big challenge because of the very huge amount of gene variants.

In this paper, we present our work on extending the PostgreSQL [3] open RDBMS with a new data type and a new aggregation function called "genome type" and "genome type aggregation function", respectively. Some preliminary examination shows that our approach is promising and can improve the execution time of disease causal gene analysis processing by a factor of 50 to 360.

2. Conventional Methods

2.1 Database Schema

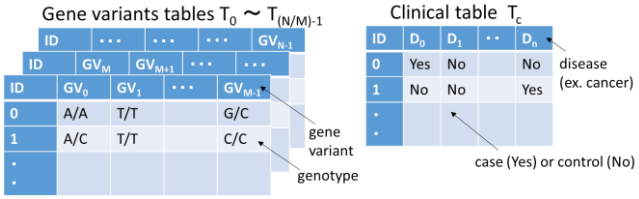

Figure 2: Conventional database schema

Figure 2 shows the database schema composed of tables that contain the N gene variants ($GV_{0...N-1}$) with the associated genotype for each individual. Note that since there is a limit on the number of columns a table can contain, in this case M, there are N/M tables ($T_{0...N/M-1}$) to store all the M gene variants of the individuals. Figure 1 shows also a clinical table (T_c) with information related to n diseases ($D_{0...n-1}$) for each individual. Other tables containing information about the patients (lifestyle, demographics, etc.) can also be necessary to properly describe the individuals.

2.2 Naïve Method

A naïve method counts the number of occurrence of each genotype for all the gene variants of patients with a specified disease. SQL 1 shows the SQL statement that calculates the distribution of genotype of a gene variant (GV_0) for patients who have a specified disease (D_0). This query is executed N times (i.e. for each $GV_{0..N-1}$). It takes 1,530ms on PostgreSQL to execute the above query on 150,000 individuals for each gene variant GV_i. For

```
SELECT count(T0.GV0) FROM T0, Tc
GROUP BY T0.GV0
WHERE T0.ID = Tc.ID and Tc.D0 = 'Yes'
```

SQL 1: SQL for naïve method

the usual case of 3,000,000 gene variants, it would take more than 50 days. There are two major reasons for such a long execution time. One is the huge number of gene variants N and thus, the huge number of corresponding SQL queries that have to be processed. The other is the long processing time of the query for the join between T_i and T_c. Note that in real cases the query could contain more joins to include lifestyle and demographics that are usually stored in other tables.

2.3 External Count Method

In order to decrease the number of queries and join operations, a method composed of an SQL that first retrieves all the genotypes of the gene variants for the individuals with a specified disease, and then an external application that counts the number of each genotype using the result of the query, is analyzed.

```
SELECT T0.GV0, T0.GV1, …T0.GVM-1 FROM T0, Tc
WHERE T0.ID = Tc.ID and Tc.D0 = 'Yes';
```

SQL 2: Pseudo SQL for external count method

SQL 2 shows the SQL statement that retrieves the genotypes of gene variants ($GV_0..GV_{M-1}$) on Table T_0 for the patients with disease D_0. Note that a similar query is executed N/M times, i.e., for all gene variants tables $T_0..T_{N/M-1}$. The result of the SQL queries is input into an external program that counts the distribution of each genotype for all gene variants. It still takes about 5,500s, including both the PostgreSQL processing and the external processing, for the case of 1,000 individuals and 3,000,000 variants. This method still has two problems. First, since there is a limit on the number of columns a table can contain, many tables are necessary to store millions of variants and thus, the costly join processing of those tables with clinical and other tables cannot be avoided. Second, external processing causes a huge amount of data flow from the RDBMS to external application and thus, a high cost transfer time is necessary.

3. Proposed Method

In order to solve these problems, we proposed a method that integrates all the genotypes of the gene variations in a single table, making possible an efficient counting of the genotypes. We created a special database type and a special aggregation function called "genome type" and "genome aggregation function", respectively.

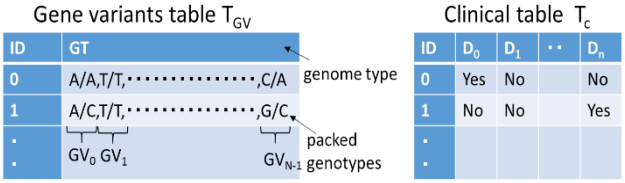

Figure 3: Proposed database schema

Figure 3 illustrates the gene variants table (T_{GV}) with the genome type (GT) column that packs all genotypes of all gene variants for each individual, enabling an efficient storing and scanning of the genotype data for its aggregation. SQL 3 shows the SQL statement with the proposed genome type aggregation function fjgeno_count(). Since all the genotypes of the gene variations are contained in GT, the genome aggregation function can efficiently count up through all genotypes of each individual at once. And only

```
SELECT fjgeno_count(TGV.GT) FROM T0
WHERE TGV.ID = Tc.ID and Tc.D0 = 'Yes';
```

SQL 3: SQL for proposed method

a single execution of the join processing with other tables as the clinical table T_c is necessary.

4. Evaluation

We implemented our proposed method on PostgreSQL, and compared its performance with the conventional methods presented in Section 3. We used the machine whose CPU is Xeon CPU E5-2680 0 @2.70GHz x2 and memory is DDR3 128GB.

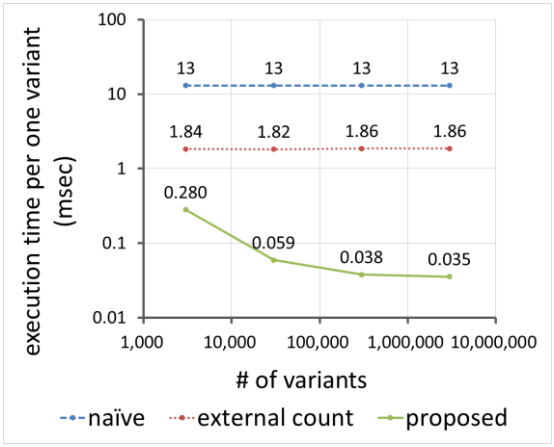

Figure 4: Execution time for the three methods

Figure 4 shows the execution time per gene variant for the case of 1,000 individuals for the naïve, the external count, and our proposed method, when varying the number of gene variants packed in GT. For the naïve method, the execution time per variant is 13ms, and for the external method is 1.86ms. On the other hand, the execution time for our method improves when increasing the number of packed variants, and it is reduced to 0.035ms which represents an improvement factor of about 50 to 360 over the conventional methods.

5. Conclusion

We proposed a new database type and new aggregation function as extensions to PostgreSQL for genomic analysis. Our preliminary evaluation showed that it can greatly improve the processing time of a typical query in medical genomics study. We are now working on further performance improvements for the genome aggregation function using dictionary and vectorization techniques, which we plan to report in detail in a future paper.

6. REFERENCES

[1] Ameur, A., Bunkikis, I., Enroth, S., et al. (2014) CanvasDB: a local database infrastructure for analysis of targeted- and while genome resequencing projects. *Database*, Vol. 2014, Article ID bau098

[2] Paila, U., Chapman, B.A., Kirchner,R. (2013) GEMINI: integrative exploration of genetic variation and genome annotations. *PLOS Comput. Biol.*,**9**,e1003153

[3] The PostgreSQL Global Development Group. (1996-2015) *PostgreSQL* http://www.postgresql.org/

[4] 1000 Genomes Project (2015) *The Variant Call Format (VCF) Version 4.2 Specification* http://samtools.github.io/hts-specs/VCFv4.2.pdf