Analisis Faktor Optimasi untuk Data Warehouse dengan Data Tabungan pada Bank XYZ

Aloysius Adhyatma Herfangsyah[1], Willy Sudiarto Raharjo[2], Antonius Rachmat Chrismanto[3]
Prodi Informatika, Universitas Kristen Duta Wacana
Jl. dr. Wahidin Sudirohusodo no. 5-25, Yogyakarta
[1]aloysius.adhyatma@ti.ukdw.ac.id
[2]willysr@ti.ukdw.ac.id
[3]anton@ti.ukdw.ac.id

Abstract*— The development of information technology processes is used by many companies to improve their business performance by presenting integrated and consistent data. Data warehouse is a scientific method that supports the company's analytic process, by providing integrated data from various database sources. As the number and complexity of data increases, companies that use harddisk or other physical memory to store the data will require more resources than the previous year. Data processing that consists of a lot of complex data and requires a lot of storage media will need an optimal database, so the company's analytic process can run quickly and efficiently. In this research, the author will build a data warehouse by creating a table with initial condition, make mappings to insert the data needed, make an optimization tables with 7 different conditions of a combination of partitioning, bucketing, and compression, then analyze the performance of each table by using some queries which will be often used for simple analytic. The performance that will be analyzed in this study are the query run time and storage space used by each table. Based on testing by using 3 queries, the author concluded that the use of partition and bucketing speeds up the average query run time by 28%. While the use of compression affects the average size of data storage by 8 to 30 times smaller when compared with uncompressed tables, the compression also slows the average query time by 77% or about twice the normal time.

Intisari— Berkembangnya proses teknologi informasi dimanfaatkan oleh banyak perusahaan untuk meningkatkan kinerja bisnisnya, yaitu dengan cara penyajian data yang terintegrasi dan konsisten. Data warehouse merupakan suatu ilmu yang menunjang proses analisis perusahaan, dengan cara menyediakan data yang terintegrasi dari berbagai sumber basis data. Seiring dengan bertambahnya jumlah dan kompleksitas data, perusahaan yang menggunakan media komputer untuk menyimpan datanya akan memerlukan resource yang lebih banyak dari tahun sebelumnya. Pemprosesan data yang banyak ini tentunya memerlukan media penyimpanan berupa basis data yang optimal, sehingga proses analisis dari perusahaan tersebut dapat berjalan secara cepat dan efisien. Pada penelitian ini, penulis akan membangun data warehouse dengan membuat tabel kondisi awal, membuat mapping untuk memasukkan data-data yang diperlukan, membuat tabel optimasi dengan 7 kondisi yang berbeda dari kombinasi partisi, bucketing, dan kompresi, lalu menganalisis performa dari tabel tersebut menggunakan query yang akan sering digunakan untuk analisis sederhana. Performa yang akan dianalisis pada penelitian ini adalah dari segi waktu jalan query dan ruang penyimpanan yang digunakan oleh masing-masing tabel. Berdasarkan pengujian dari penelitian ini dengan menggunakan 3 query, dihasilkan kesimpulan bahwa penggunaan partisi dan bucketing mempercepat rata-rata jalannya query sebesar 28%, sementara penggunaan kompresi data mempengaruhi rata-rata ukuran ruang penyimpanan data sebesar 8 hingga 30 kali lebih kecil jika dibandingkan dengan tabel yang tidak dikompresi, namun penggunaan kompresi ini memperlambat rata-rata jalannya waktu query sebesar 77% atau sekitar hampir dua kali lipat.

Kata Kunci*— basis data, *data warehouse*, optimasi, etl, *big data

I. PENDAHULUAN

A. *Latar Belakang Masalah*

Saat ini banyak perusahaan yang memanfaatkan teknologi informasi untuk meningkatkan kinerja bisnisnya dalam menghadapi persaingan bisnis yang ketat dan meraih pasar lebih luas, serta memperoleh keuntungan lebih besar, yaitu dengan cara mengoptimalkan penyajian informasi secara terintegrasi dan konsisten. Penerapan sistem *data warehouse* yang selama ini diterapkan oleh Bank XYZ, nyatanya, telah memberikan efek yang baik bagi pihak bank, contohnya proses analisis data dan pengelolaan informasi yang didapat dari data historis yang dilakukan di perusahaan akan menjadi lebih efisien, dan membuat proses pengambilan keputusan yang bersifat strategis dapat dilaksanakan dengan lebih optimal dan cepat karena berdasarkan proses penganalisisan yang akurat, dan data yang diperoleh berasal dari data historis yang telah diproses selama ini.

Data warehouse merupakan salah satu bagian dari teknologi informasi yang dapat menunjang proses analisis perusahaan, dengan membantu *user* bisnis perusahaan dalam membantu menentukan kebijakan – kebijakan perusahaan agar dapat menghasilkan keputusan yang cepat dan tepat berdasarkan hasil analisis dari *data mining* dengan menggunakan data dan fakta yang telah diproses oleh *data warehouse*, dengan cara melakukan proses ETL, yaitu proses yang bertujuan untuk mengumpulkan sumber data yang berasal dari berbagai macam data operasional agar dapat terkumpul ke dalam sebuah media penyimpanan berupa basis data.

Seiring dengan pertambahan jumlah dan kompleksitas data yang cukup pesat, pemrosesan data menggunakan *data warehouse* dirasa kurang optimal [6]. Hal ini dipengaruhi oleh beberapa faktor, yaitu *data warehouse* yang tidak bisa memproses data yang tidak terstruktur, serta tidak mendukung pemrosesan data secara *near real time*. Maka pendekatan *data warehouse* menggunakan konsep *big data* menjadi solusi atas permasalahan tersebut. Karena pada *big*

data, proses yang dilakukan akan memakan waktu lebih cepat karena menggunakan proses paralelisasi dan *map reduce*, dan dapat menerima data yang tidak terstruktur seperti gambar, video dan suara.

Kinerja basis data merupakan isu yang tidak kalah penting saat data bertambah besar dan kompleks. Jumlah data yang disimpan di piringan magnetik (disket, *harddisk*, dll) meningkat 100% pertahun, artinya setiap perusahaan di dunia yang menggunakan komputer sebagai tempat menyimpan data yang dimilikinya akan meningkat dua kali lipat per tahun [1]. Oleh sebab itu perlu suatu cara untuk menjaga kinerja sistem basis data agar tetap optimal seiring dengan bertambahnya jumlah dan kompleksitas data.

B. *Tinjauan Pustaka*

Penelitian tentang optimasi basis data dengan cara mengoptimalkan *query*, yaitu cara yang digunakan untuk meningkatkan strategi evaluasi dari suatu *query* untuk membuat nilai evaluasi dari *query* tersebut menjadi lebih efisien dan berdampak pada hasilnya [2]. Penelitian menggunakan metode algoritma subset query, menghasilkan kesimpulan jika data yang akan diproses relatif kecil, *query* menggunakan cross product akan memberikan hasil berupa waktu yang lebih cepat jika dibandingkan dengan metode subset query, sebaliknya, penggunaan metode subset query akan lebih cepat jika dibandingkan dengan metode cross product jika menggunakan data yang besar [3]. Penelitian sebelumnya yang telah dilakukan tentang optimasi yaitu membandingkan penggunaan file format yang digunakan untuk menyimpan data pada tabel Hive, tujuan dari penelitian ini yaitu mencari format terbaik yang digunakan untuk kondisi tertentu [4]. Penelitian selanjutnya yaitu membandingkan format berbentuk Text, Parquet, dan PCAP untuk menyimpan data tabel, kesimpulan yang didapat dari penelitian ini adalah penggunaan format *Parquet* adalah yang paling optimal jika dibandingkan dengan Text dan PCAP, karena jika dibandingkan dalam hal *storage*, file Parquet menggunakan *storage* sekitar 40% lebih kecil jika dibandingkan dengan Text, serta file PCAP yang tidak mendukung kebutuhan analisis karena guna file PCAP biasanya digunakan untuk menyimpan *log* [5].

Penelitian lainnya tentang optimasi pada Hive adalah *Evaluating partitioning and bucketing strategies for Hive-based Big Data Warehousing systems*. Tujuan dari penelitian ini yaitu menganalisis optimasi *query* pada tabel yang telah di partisi dan di *bucketing*. Dari penelitian ini didapat kesimpulan bahwa melakukan partisi dan *bucketing* memberikan penambahan kecepatan pada saat *query data*, tetapi jika data yang digunakan tidak terlalu besar, maka penggunaan *bucketing* justru memperlambat kinerja *query* [6].

Berbeda dengan beberapa penelitian yang telah di sebutkan diatas, penelitian ini akan menganalisis *best practice* dari penggunaan kombinasi partisi, *bucketing*, serta kompresi pada tiap tabel yang akan dibuat, selain menerapkan *best practice* dari kombinasi tersebut, penelitian ini akan melakukan studi kasus dengan pemilihan kolom yang paling optimal saat melakukan partisi dan *bucketing* pada tiap tabel, serta analisis optimasi pemberian jumlah dari *bucket* itu sendiri.

C. *Tujuan Penelitian*

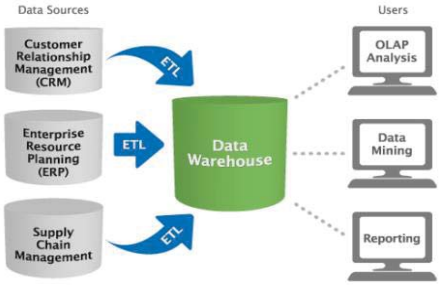
Adapun tujuan dari penelitian ini adalah sebagai berikut:

- Mengoptimalkan proses *data warehouse* pada Bank XYZ.
- Menganalisis pemrosesan *data warehouse* dengan desain yang telah di optimasi dan membandingkannya dengan desain yang belum di optimasi dalam hal kecepatan proses *query* pada tabel.

II. LANDASAN TEORI

A. *Data Warehouse*

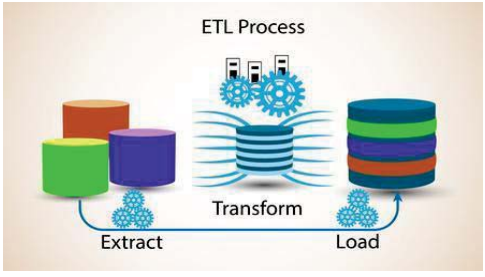
Data Warehouse merupakan sebuah sistem yang mengubah data dari berbagai sumber ke dalam bentuk *dimensional* yang kemudian dapat mendukung proses *query* dan analisis untuk tujuan mengambil keputusan lebih lanjut [4]. *Data warehouse* merupakan sebuah rangkuman data yang memiliki sifat berorientasi pada subjek, tetap, terintegrasi, dan *time-variant* yang telah didesain dan digunakan untuk mendukung pengambilan keputusan. Gambar 1 menunjukan arsitektur data warehouse.



Gambar 1. Arsitektur Data Warehouse (https://cdn.intellipaat.com/wp-content/uploads/2016/01/data-warehouse-460x329.png)

B. *Extract, Transform, Load*

Extract, Transform, and Load atau ETL dapat dikatakan sebagai kunci utama dalam *data warehouse*. ETL memegang peran penting dalam kinerja *data warehouse*. ETL terdiri dari tiga proses besar, yaitu *Extract* (sistem membaca data dari satu atau lebih sumber data), *Transform (*mengubah data yang telah diperoleh dari proses *Extract* dan kemudian mengubahnya ke dalam format tertentu sebelum kemudian disimpan dalam *data warehouse)*, dan *Load (*proses memasukkan data ke dalam *data warehouse*). ETL merupakan bagian sistem yang penting sehingga dalam mengumpulkan kebutuhan sistem ini perlu dilakukan dengan hati-hati. Selain itu, kebutuhan sistem ETL inilah yang akan menentukan seluruh sistem. Dengan kata lain, proses – proses berikutnya akan
menyesuaikan dengan proses ETL yang telah ada, proses ETL dapat dilihat pada Gambar 2



Gambar 2. Proses ETL

JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.191

(https://www.softbless.com/sites/all/themes/softbless/images/content/ETL-Extract-Transform-Load.jpeg)

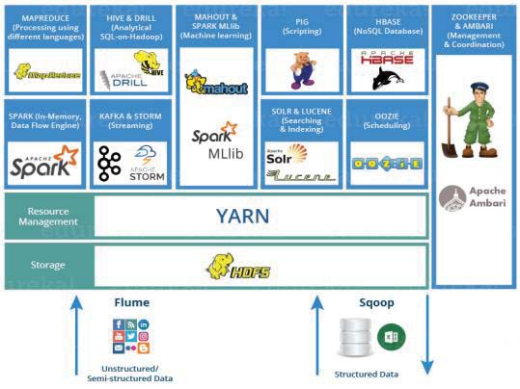
C. Big Data

Big Data merupakan *dataset* yang rumit dan tidak mudah untuk dikoleksi, disimpan, dikelola, dan dianalisis dengan menggunakan basis data konvensional karena cakupan yang luas dan ukuran yang besar. *Big Data* memiliki tiga karakteristik yaitu *volume, velocity, dan variety*.

1) *Volume: Volume* yang besar secara terus menerus terbentuk dari jutaan aplikasi dan perangkat. Setiap hari, data digital yang dihasilkan di seluruh dunia mencapai 2,5 exabyte. Jumlah ini akan bertambah dua kali lipat setiap 40 bulan [3]. Merujuk data yg dihasilkan oleh International Data Corporation (perusahaan yang menerbitkan report tentang penelitian), data yang telah dihasilkan, direplikasi, dan digunakan pada tahun 2012 saja sudah mencapai 4,4 zettabyte.
2) *Velocity*: Data yang dihasilkan secara cepat, sebaiknya diproses dengan cepat untuk memperoleh informasi yang berguna dan wawasan yang luas bagi perusahaan terkait. Sebagai contoh, perusahaan Walmart menghasilkan data sebesar 2,5 pentabyte setiap jam nya dari hasil transaksi pelanggan, maka dengan data sebesar itu, dibutuhkan alat yang dapat mengolah / memproses data tersebut.
3) *Variety: Big Data* dihasilkan dari berbagai sumber dan menghasilkan data dengan berbagai format (foto, video, suara, logs, dll). Dataset yang besar ini tersusun dari data yang terstruktur dan data yang tidak terstruktur.

D. Apache Hadoop

Apache Hadoop adalah suatu *framework* Java yang bersifat *open source* yang digunakan untuk menjalankan aplikasi yang berjalan pada *environment Big Data*. Hadoop biasanya digunakan pada lingkungan yang menyediakan *storage* dan komputasi secara terdistribusi ke kluster – kluster dari komputer/node yang saling terhubung (*parallel* computing) dengan ukuran data yang besar. Hadoop dapat digunakan untuk analisis data tidak terstruktur, semi-terstruktur, maupun data yang terstruktur, sehingga Hadoop banyak digunakan pada lingkungan *Big Data* yang menampung data yang besar serta berbagai macam tipe data dari berbagai sumber. Salah satu contoh aplikasi pada Hadoop adalah Hive. Ekosistem Hadoop dapat dilihat pada Gambar 3



Gambar 3. Ekosistem Hadoop
(https://medium.com/@stevanihalim/sedikit-tentang-hadoop-18d6ade32ae7)

E. Hadoop Distributed File System

Hadoop distributed file system atau HDFS merupakan sistem file yang digunakan untuk menampung data pada ekosistem *Hadoop*, HDFS mendukung *transfer* data secara cepat antar nodes, saat data disimpan dengan HDFS, sistem ini akan membagi informasi data tersebut ke beberapa blok dan membagi ke beberapa nodes di *cluster*, sehingga memberikan efisiensi yang tinggi untuk proses secara paralel.

F. Apache Hive

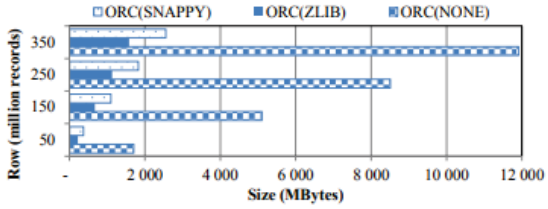
Hive adalah salah satu *tools* pada *data warehouse* yang dibuat dengan berbagai tujuan, contohnya yaitu untuk keperluan *query*, untuk memfasilitasi *summary* data dengan mudah, serta untuk analisis data besar yang tersimpan di berbagai media penyimpanan dan berbagai file sistem yang telah terintegrasi dengan Hadoop. Hive memiliki cara yang mudah untuk memproses data dengan melakukan query seperti SQL query pada data tersebut. Hive dapat dengan mudah mengintegrasikan teknologinya dengan teknologi konvensional seperti Oracle. Hive juga mampu memproses berbagai data seperti *flatfile* dengan berbagai *delimiter*, file JSON, maupun data berbentuk tabel pada basis data. Tabel pada Hive dapat dibedakan menjadi 2, yaitu tabel internal / *managed* tabel dan tabel eksternal, tabel internal adalah tabel yang memiliki format sama seperti tabel seperti pada basis data secara umum, sementara tabel eksternal adalah tabel yang dapat disimpan dengan format file lain seperti *textfile*, ORC, Avro, atau Parquet. Pada Hive, ada 3 hal penting yang sering dibahas, yaitu Tabel, Partisi, dan Bucket. Hive juga memiliki metastore yang digunakan untuk menyimpan metadata. Selain itu, Bahasa *query* yang digunakan pada Hive adalah HiveQL (HQL), Bahasa *query* ini sangat mirip dengan SQL yang biasa digunakan pada Oracle.

1) *Hive Partition:* Partisi pada Big Data bersifat sama dengan partisi yang ada pada RDBMS pada umumnya, yaitu membagi data dengan menggunakan kolom tertentu, sehingga kumpulan data yang memiliki kolom yang sama akan disimpan pada directory yang sama, maka pada saat melakukan query dengan kolom tersebut, kecepatan baca data akan menghasilkan waktu *query* yang lebih cepat jika dibandingkan dengan tabel yang tidak di partisi.
2) *Hive Bucketing: Bucketing* pada Hive berarti membagi data yang telah dipartisi atau dibagi sebelumnya menjadi bucket – bucket dalam bentuk file dengan size yang kurang lebih sama, hal ini dilakukan untuk mencegah Hive membuat file dengan jumlah banyak dan dengan ukuran yang tidak sama pada folder dalam partisi sebelumnya, karena semakin banyak file yang dihasilkan oleh partisi maka proses baca data akan semakin lama.

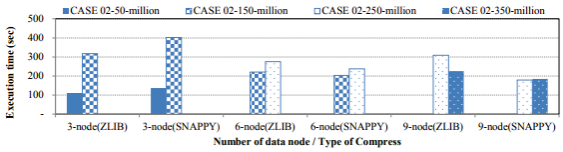
G. Kompresi Data

Kompresi adalah upaya untuk memperkecil ukuran dari satu atau banyak file. Ketika file dikompresi, file ini akan menggunakan ruang penyimpanan yang lebih sedikit jika dibandingkan dengan file yang tidak dikompres, dan dapat dipindahkan ke tempat lain dengan lebih cepat, maka dari itu, kompresi sering digunakan untuk menghemat ruang penyimpanan dan mengurangi waktu yang diperlukan untuk memindahkan data.

1) *Kompresi Snappy: Snappy* adalah salah satu library untuk kompresi / dekompresi. Snappy tidak berfokus pada maksimum efisiensi pada hasil kompresi, atau kompabilitas dengan library lainnya, melainkan berfokus pada kompresi dengan kecepatan yang tinggi. Sebagai contoh, pada Gambar 5 dan Gambar 6, dapat dilihat terdapat penelitian terhadap perbandingan kompresi snappy dan zlib dari segi waktu dan hasil.



Gambar 5. Perbandingan Snappy dan Zlib segi hasil (Rattanaopas et al, 2017, hal. 156) [15]



Gambar 6. Perbandingan Snappy dan Zlib segi waktu (Rattanaopas et al, 2017, hal. 157) [15]

H.Apache Parquet

Apache Parquet adalah tipe file berbentuk kolumnar yang dapat digunakan pada Hadoop (Hive, HBase, Pig). File data kolumnar adalah tipe data optimal untuk menyimpan data berbentuk tabel / data yang telah dibagi menjadi kolom – kolom

ID	Name	Department
1	emp1	d1
2	emp2	d2
3	emp3	d3

Gambar 7. File berbentuk tabel

Gambar 7 merupakan file sebuah tabel, jika file ini disimpan dalam bentuk *snappy*, maka file ini akan berubah menjadi seperti Gambar 8

1	2	3	emp1	emp2	emp3	d1	d2	d3

Gambar 8. File berbentuk *parquet*

file Parquet ini akan memberikan hasil optimal dalam waktu *query*, karena ketika sebuah *query* memerlukan kolom *Name*, file ini tidak perlu melihat kolom ID dan kolom *Department*, sehingga akan menghasilkan waktu *query* yang lebih cepat.

III. PERANCANGAN SISTEM

A. Spesifikasi Sistem

Berikut adalah spesifikasi sistem yang akan digunakan pada penelitian ini:

1) *Kebutuhan Perangkat Keras:* Kebutuhan perangkat keras pada penelitian ini adalah sebagai berikut:
   - Windows10 64bit
   - Processor Intel® Xeon® CPU E5-2699 @ 2.40GHz (2 processors)
   - RAM 16GB

2) *Kebutuhan Perangkat Lunak:* Kebutuhan perangkat lunak pada penelitian ini adalah sebagai berikut:
   - Informatica Developer versi 10.2.1
   - Dbeaver versi 5.2.1
   - Microsoft Edge versi 38
   - SSH versi 3.2.9

B. Rancangan Database

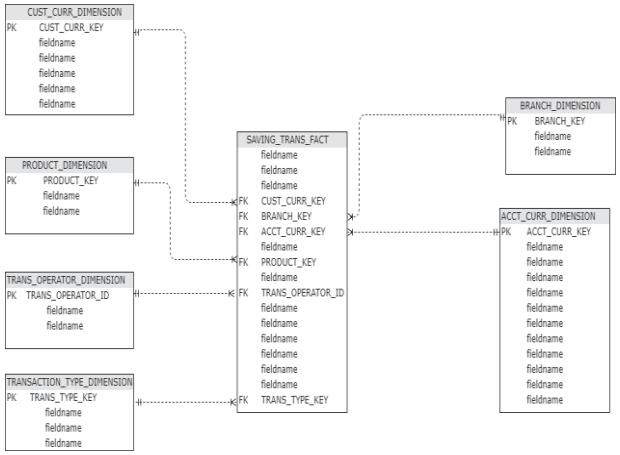
Data warehouse yang digunakan untuk membangun proses terdapat seperti pada Tabel 1. *Data warehouse* ini membutuhkan beberapa tabel dengan fungsi yang berbeda – beda. Tabel – tabel tersebut adalah tabel *customer* untuk mencatat nasabah yang ada, tabel *account* yang berfungsi mencatat akun dari nasabah, tabel *saving* untuk mencatat transaksi tabungan dari nasabah, tabel *checking* untuk mencatat transaksi giro, tabel *time deposit* untuk mencatat transaksi deposit, sedangkan tabel *acct_dly_bal_summ_fact* adalah transaksi yang dilakukan oleh nasabah perhari.

TABEL I
LIST TABEL PENELITIAN

Nama Tabel	Keterangan Tabel
CUST_CURR_DIMENSION	Tabel master kustomer
ACCT_CURR_DIMENSION	Tabel master akun
SAVING_TRANS_FACT	Tabel Fakta Tabungan
CHECKING_TRANS_FACT	Tabel Fakta Giro
TD_TRANS_FACT	Tabel Fakta Deposit
DLY_ACCT_BAL_FACT	Tabel Fakta Trans Harian

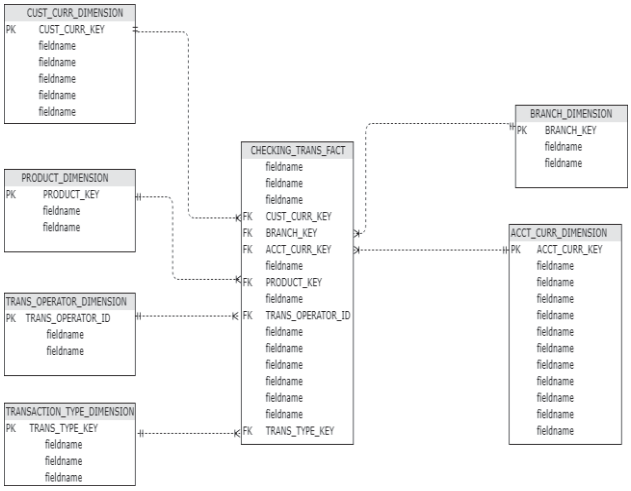
C. Skema Data Warehouse

Pada penelitian ini, skema yang digunakan adalah *star schema*, yaitu skema yang terdiri atas data faktual pada bagian tengah dan dikelilingi tabel dimensi yang berisi referensi data. Pada model ini, tiap tabel dimensi harus terhubung ke tabel fakta, Gambar 9 merupakan skema untuk tabel saving, sedangkan Gambar 10 adalah skema untuk tabel checking, Gambar 11 adalah skema untuk tabel deposit, dan Gambar 12 adalah skema untuk tabel summary

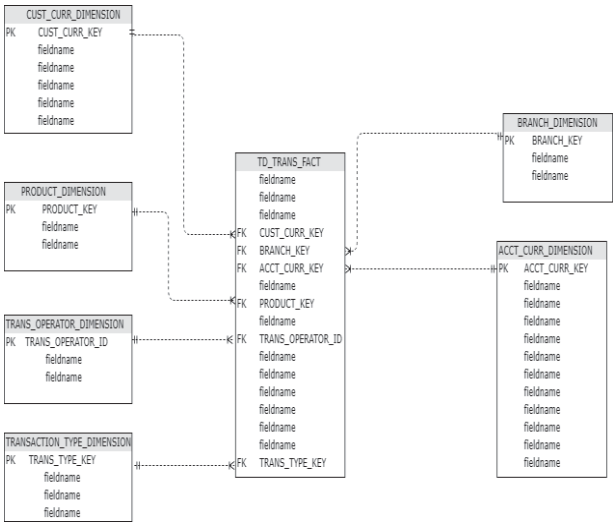


Gambar 9. Skema tabel saving

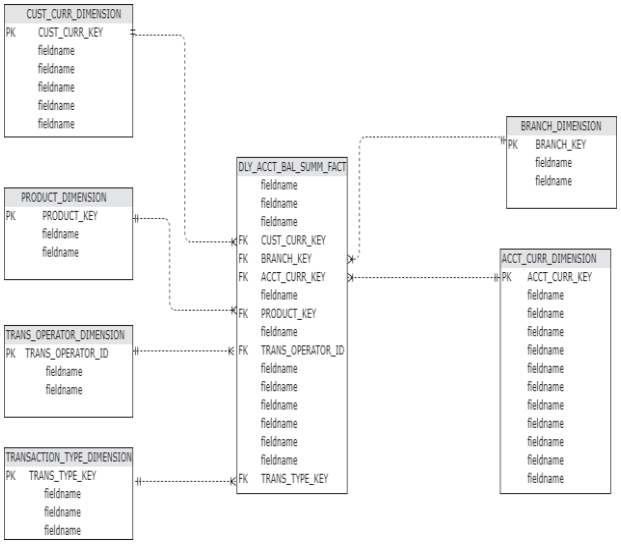
JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.191



Gambar 10. Skema tabel checking



Gambar 11. Skema tabel deposit



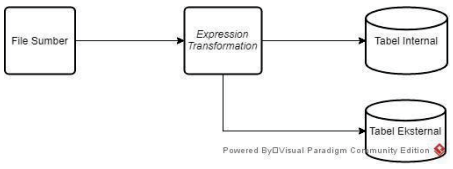
Gambar 12. Skema tabel summary

D. Langkah Penelitian

Pada penelitian ini, akan dilakukan dengan beberapa langkah, yaitu:

1)Migrasi Data Warehouse: Pada migrasi ini, dilakukan penyesuaian terhadap proses ETL yang awalnya menggunakan basis data Oracle, menjadi basis data Hive. Proses migrasi ini meliputi beberapa tahap, yaitu:

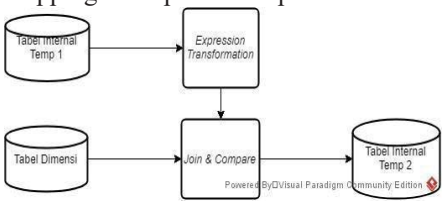
1. Menentukan sumber
   Pada tahap ini, ditentukan *source* apa saja yang akan digunakan untuk membuat *data warehouse* pada *Big Data*. Sumber data pada penelitian ini didapatkan dari dua sumber utama, yaitu IDS (*Integrated Deposit System*) dan CIS (*Customer Integrated System)*. Sumber yang majemuk ini akan dipilih dan dianalisis, dan selanjutnya akan ditentukan mana saja yang berguna sebagai data sumber untuk dimasukkan ke *data warehouse*.
2. Pembuatan tabel
   Setelah menentukan *source* yang akan digunakan, selanjutnya menentukan *source file* tersebut akan digunakan pada tabel dimensi atau tabel fakta dengan membuat tabel menggunakan aplikasi DBeaver. Pada tabel awal ini, tidak ada perlakuan optimasi yang dilakukan.
3. Pembuatan *mapping*
   Tahap ketiga yaitu menentukan *logic* dan *transformation* yang akan diaplikasikan untuk *source file* tersebut pada *mapping* dengan menggunakan aplikasi Informatica Developer. Pembuatan *mapping* untuk tabel dimensi pada *big data* ini dapat dikerjakan dengan empat tahapan, yaitu:
   a. Tahap 1 tabel dimensi
      Pada tahap ini, akan dilakukan transformasi *expression* berupa pembersihan data / *data cleansing*, contohnya adalah *trim* dan atau pengubahan tipe data, contohnya *padding* atau mengubah desimal menjadi *string* atau *string* menjadi *timestamp*, setelah dilakukan transformasi, maka data tersebut akan dimasukkan ke dua tabel, yaitu tabel *internal* dan tabel *external*, tabel *internal* adalah tabel *temporary* yang akan digunakan pada tahap selanjutnya, sedangkan tabel *external* ini akan disebut *raw table*, yaitu tabel yang berisi data yang akan digunakan sebagai dokumentasi *flatfile*. Tahap ini dapat dilihat pada Gambar 13



Gambar 13 Mapping Tahap 1 Dimensi

   b. Tahap 2 tabel dimensi
      Pada tahap ini, tabel *temporary* pada langkah sebelumnya akan di *join* dengan tabel akhir dimensi, dengan tujuan untuk mendapatkan *flag* berupa N, U atau D, yaitu data baru, data *update*, atau data *delete*, dengan cara membandingkan *primary key* untuk data baru dan data *delete* dan membandingkan semua

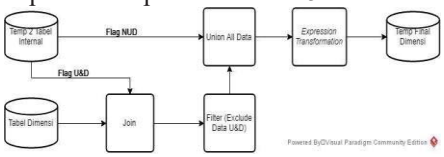
kolom untuk mendapatkan *flag update*, sementara data yang tidak memiliki *flag* / tidak mengalami perubahan, data tersebut akan di *exclude* pada *target* tabel *temporary 2* ini, sehingga pada tabel *temporary 2* hanya berisi data yang memiliki *flag*, contoh mapping ini dapat dilihat pada Gambar 14



Gambar 14 Mapping Tahap 2 Tabel Dimensi

c. Tahap 3 tabel dimensi

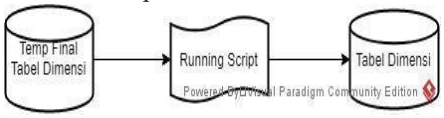
Pada tahap ketiga, tabel *temporary 2* pada langkah sebelumnya akan di *union* dengan data pada tabel dimensi akhir, dengan tujuan mengambil data terbaru (*new, update*, dan *delete*) dari tabel *temporary* 2, sedangkan data yang tidak mengalami perubahan akan diambil dari tabel dimensi akhir, setelah dua tabel tersebut di *union*, maka langkah selanjutnya adalah menambahkan kolom *inserted_dt*, yaitu kolom yang menunjukan tanggal data tersebut dimasukkan ke tabel *temporary final* seperti dapat dilihat pada Gambar 15



Gambar 15 Mapping Tahap 3 Dimensi

d. Tahap final tabel dimensi

Pada tahap ini hanya dilakukan *running script* berupa *refresh data* dari temp final ke tabel dimensi akhir, perbedaan dari *temp final* dan tabel dimensi akhir adalah pada jumlah datanya, pada tabel dimensi akhir, data akan diarahkan pada data dengan *inserted_dt* terbesar, dalam arti data yang akan ditampilkan pada tabel dimensi akhir adalah data – data terbaru, sedangkan pada *temp final*, data lama akan tetap ada, dengan tujuan untuk *tracking*, seperti dicontohkan pada Gambar 16

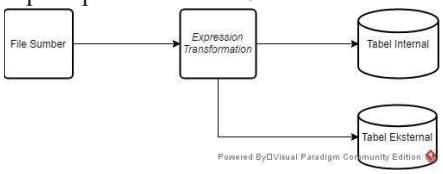


Gambar 16. Mapping Akhir Tabel Dimensi

Sementara untuk tabel fakta, *mapping* dijalankan 3 tahap, sebagai berikut:

a. Tahap 1 tabel fakta

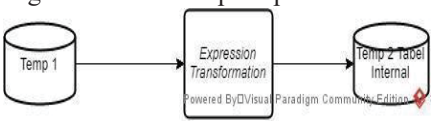
Pada tahap 1 untuk tabel fakta, langkah yang dilakukan sama seperti tabel dimensi, yaitu *load* data dari file sumber dan dimasukkan ke dua tabel, yaitu tabel *internal* dan tabel *eksternal*, seperti pada Gambar 17



Gambar 17. Mapping Tahap 1 Tabel Fakta

b. Tahap 2 tabel fakta

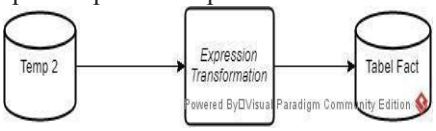
Pada tahap 2 ini, *expression* yang dilakukan pada tabel *temporary* 1 adalah menambahkan logika proses bisnis, agar pada tabel *temporary 2*, bentuk data yang ada sudah siap untuk dianalisis oleh divisi lain atau *user* bisnis. Pada tabel fakta, tidak ada *update* data, karena sifat dari tabel fakta yang merupakan data transaksi yang berbeda beda seperti pada Gambar 18



Gambar 18. Mapping Tahap 2 Tabel Fakta

c. Tahap akhir tabel fakta

Pada tahap akhir tabel fakta, akan ditambahkan transformasi *expression* berupa penambahan kolom *inserted_dt*, sebagai penanda data tersebut kapan dimasukkan ke dalam tabel fakta, seperti dapat dilihat pada Gambar 19



Gambar 19. Mapping Akhir Tabel Fakta

2)Optimasi Pembuatan Tabel: Setelah data pada source file telah dimasukkan ke dalam tabel big data, maka langkah selanjutnya yaitu membuat tabel dengan desain yang telah dioptimasi, optimasi tabel ini akan dilakukan dengan 3 cara, yaitu partisi, bucketing, dan kompresi. Terdapat 7 skenario untuk analisis optimasi tabel ini untuk tabel dimensi, yaitu dengan menggunakan kombinasi dari partisi, kompresi, dan bucketing. Pada tabel fakta, hanya terdapat perlakuan partisi dan kompresi, karena untuk data pada tabel fakta pada penelitian ini, tidak memungkinkan untuk dilakukan bucketing.

3)Pengujian Performa Tabel: Setelah data – data telah berhasil dimasukkan kedalam tabel akhir dimensi / fakta pada semua tabel, maka langkah selanjutnya adalah menganalisis kecepatan baca data tabel akhir dengan query yang akan digunakan oleh user. Perbedaan waktu query dari tiap tabel yang belum dan telah dioptimasi dengan masing – masing perlakuan akan dicatat dan akan dibuatkan graf hasil perbedaan. Pada tahap ini, akan dilakukan pengujian performa dari tiap tabel dengan 3 query yang akan didapat dari user dengan menjalankan query tersebut ke semua tabel. Dari hasil query yang didapat dari tiap tabel dan tiap skenario, diharapkan query yang dilakukan pada tabel yang telah dioptimasi (tabel skenario optimasi 1 - 7) akan menghasilkan waktu query

JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.191

yang lebih cepat dari tabel yang tidak dioptimasi (tabel kondisi awal). Kondisi optimasi untuk tabel dimensi dapat dilihat pada TABEL 2, dan kondisi optimasi untuk tabel fakta dapat dilihat pada Tabel 3, huruf V pada Tabel 2 berarti perlakuan tersebut dilakukan, dan huruf X berarti perlakuan tersebut tidak dilakukan pada skenario tersebut.

TABEL II
OPTIMASI TABEL DIMENSI

	Partisi	*Bucketing*	Kompresi
Skenario 1	V	X	X
Skenario 2	X	V	X
Skenario 3	X	X	V
Skenario 4	X	V	V
Skenario 5	V	X	V
Skenario 6	V	V	X
Skenario 7	V	V	V

TABEL III
OPTIMASI TABEL FAKTA

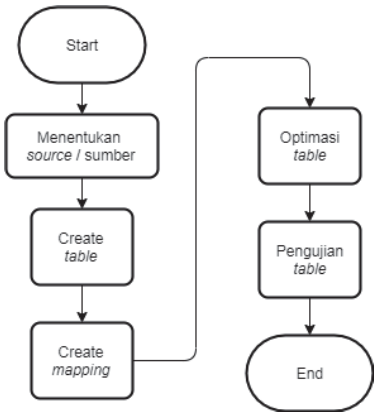
	Partisi	Kompresi
Skenario 1	V	X
Skenario 2	X	V
Skenario 3	V	V

Dalam skema pengujian tabel ini, akan dijalankan 3 contoh query yang sekira nya akan menjadi sampel dari query – query yang akan dieksekusi oleh user bisnis dalam pengambilan keputusan pada Bank XYZ. Contoh query tersebut antara lain:

- Menghitung total deposito di tiap cabang
- Menghitung total setoran per tipe transaksi
- Menghitung total setoran giro per hari

E. Flowchart Penelitian

Pada penelitian ini, penulis akan melakukan beberapa langkah atau metode pengerjaan, langkah pertama yaitu menentukan *source* atau sumber, yaitu menentukan tabel-tabel yang akan digunakan dalam penelitian ini, berikutnya adalah membuat tabel tersebut pada aplikasi Hive, setelah menentukan tabel-tabel yang akan digunakan, akan dibuat *mapping* yang berguna untuk memasukkan data-data ke tabel-tabel penelitian tersebut, yang selanjutnya yaitu membuat tabel dengan beberapa skenario optimasi, langkah berikutnya adalah pengujian tabel-tabel tersebut. Langkah penelitian ini dapat dilihat pada flowchart pada Gambar 20



Gambar 20. Flowchart Penelitian

IV. IMPLEMENTASI SISTEM

A. Persiapan Implementasi

Pada penelitian ini, perlu disiapkan persiapan implementasi berupa pembuatan tabel dan pembuatan *mapping*.

1) Script query create table: Pada bagian ini, penulis akan membuat tabel dengan kondisi yang tidak diberikan perlakuan apapun (tidak di partisi, tidak di bucketing, dan tidak di kompresi). Berikut adalah script pembuatan tabel yang dilakukan untuk seluruh tabel yang akan digunakan pada penelitian ini di aplikasi Hive, terdapat 2 contoh query, yaitu untuk tabel internal dan untuk tabel eksternal, seperti pada Tabel 4.

TABEL IV
TABEL PADA HIVE

Tabel Internal	Tabel Eksternal
`Create table schema.table_name ( column_name datatype column_name datatype ) row format delimited fields terminated by '\021';`	`Create external table schema.table_name ( column_name datatype column_name datatype ) location 'path/to/location' row format delimited fields terminated by '\021';`

2) Script query analisis data

```
SELECT
sum(a.total_amount),
b.nama_customer,
c.nama_cabang,
d.nama_produk,
date_add(from
unixtime(unix_timestamp('01-01-1600',
"dd-MM-yyyy")),
(a.tgl_proses - 2305448)
FROM XYZDW.DLY_ACCT_BAL_SUMM_FACT a
INNER JOIN XYZDW.CUST_CURR_DIMENSION b
ON a.cust_curr_key = b.cust_curr_key
INNER JOIN XYZDW.BRANCH_DIMENSION c ON
a.branch_key = c.branch_key
INNER JOIN XYZDW.PRODUCT_DIMENSION d
ON a.product_key = d.product_key
WHERE BRANCH_NAME <> 'UNKNOWN'
GROUP BY b.nama_customer,
c.nama_cabang, d.nama_produk,
a.tgl_proses
```

```
SELECT
sum(a.total_amount),
b.nama_customer,
c.jenis_transaksi,
date_add(from
unixtime(unix_timestamp('01-01-1600',
"dd-MM-yyyy")),
(a.tgl_proses - 2305448)
FROM XYZDW.SAVING_TRANSACTIONS_FACT a
```

```
INNER JOIN XYZDW.CUST_CURR_DIMENSION b
ON a.cust_curr_key = b.cust_curr_key
INNER JOIN
XYZDW.TRANSACTION_TYPE_DIMENSION c ON
a.trans_type_key = c.trans_type_key
GROUP BY b.nama_customer,
c.jenis_transaksi,a.tgl_proses
```

```
SELECT
sum(a.total_amount),
b.nama_customer,
c.jenis_transaksi,
date_add(from
unixtime(unix_timestamp('01-01-1600',
"dd-MM-yyyy")),
(a.tgl_proses - 2305448)
FROM XYZDW.CHECKING_TRANSACTIONS_FACT
a
INNER JOIN XYZDW.CUST_CURR_DIMENSION b
ON a.cust_curr_key = b.cust_curr_key
INNER JOIN
XYZDW.TRANSACTION_TYPE_DIMENSION c ON
a.trans_type_key = c.trans_type_key
WHERE a.trans_type_key IN (214,208)
GROUP BY b.nama_customer,
c.jenis_transaksi, a.tgl_proses
```

B. Implementasi Optimasi Data Warehouse

Pembuatan mapping ini adalah proses migrasi dari Informatica PowerCenter menuju Informatica Developer, dimana pada PowerCenter, basis data yang digunakan adalah Oracle, sedangkan pada Developer, basis data yang digunakan adalah Hive, logika yang digunakan untuk pembuatan mapping ini sama dengan yang ada pada Informatica PowerCenter, hanya saja, beberapa fungsi seperti lookup, non-equal join, dan update strategy tidak dapat diaplikasikan pada Informatica Developer, sehingga untuk fungsi-fungsi tersebut, digunakan logika yang sedikit berbeda.

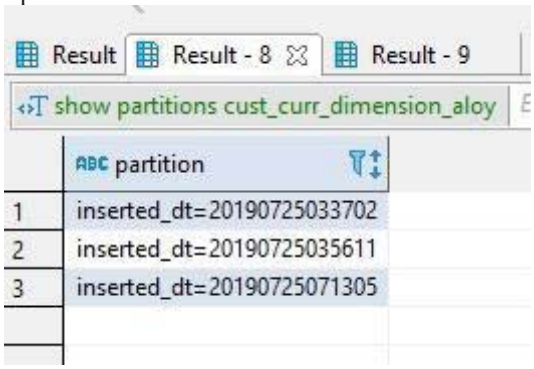
▪ Hasil partisi pada tabel dapat dilihat dengan menjalankan *query* pada Hive dengan sintaks sebagai berikut:

```
Show partitions tableName;
```

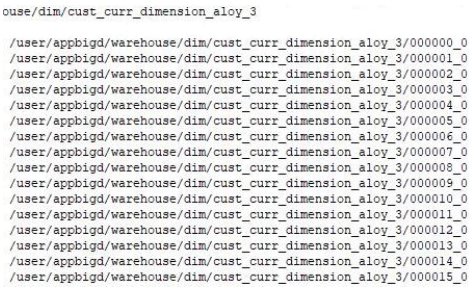
Sementara hasil *bucketing* pada tabel dapat dilihat pada konsol dengan menjalankan sintaks sebagai berikut:

```
Hadoop fs –ls path/to/directory
```

▪ Contoh hasil partisi dan bucketing pada tabel dapat dilihat pada Gambar 21 dan Gambar 22



Gambar 21. Hasil Partisi Tabel



Gambar 22. Hasil *Bucketing* Tabel

C. Pengujian Performa Tabel

Dengan menggunakan *query* yang ada, pada tahap ini akan dilakukan pengujian performa dari segi waktu dan ruang penyimpanan yang digunakan, pada Tabel 5, Tabel 6, dan Tabel 7 adalah hasil waktu dari eksekusi *query* dengan berbagai skenario, sedangkan Tabel 8 adalah hasil kompresi data pada tabel.

- Skenario 1: Skenario awal
- Skenario 2: Skenario partisi
- Skenario 3: Skenario *bucketing*
- Skenario 4: Skenario kompresi
- Skenario 5: Skenario partisi dan *bucketing*
- Skenario 6: Skenario partisi dan kompresi
- Skenario 7: Skenario *bucketing* dan kompresi
- Skenario 8: Skenario partisi, *bucketing* dan kompresi

TABEL V
HASIL QUERY 1

Skenario	1	2	3	4	5	6	7	8
Waktu (detik)	183	173	186	342	305	173	145	283

TABEL VI
HASIL QUERY 2

Skenario	1	2	3	4	5	6	7	8
Waktu (detik)	243	554	454	540	391	407	227	378

TABEL VII
HASIL QUERY 3

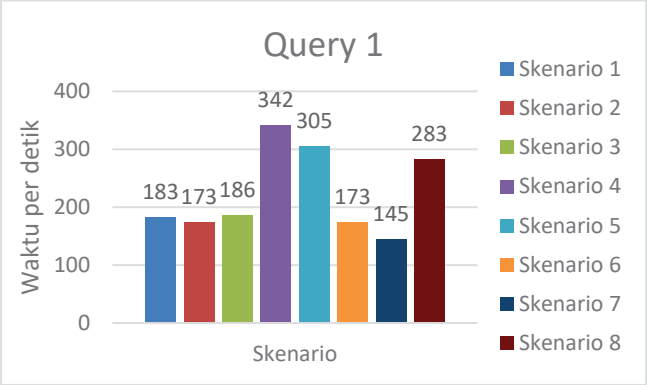
Skenario	1	2	3	4	5	6	7	8
Waktu (detik)	272	416	415	366	222	250	114	273

TABEL VIII
HASIL KOMPRESI DATA

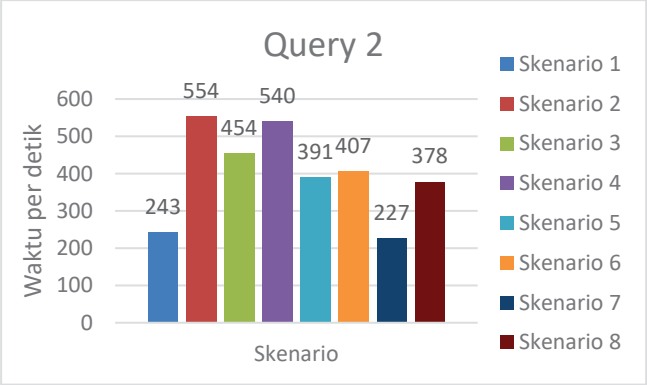
	Cust	Acct	Check	Saving	Deposit	Summary
1	17.8G	6.1M	16.2G	3.4G	2.4M	11.6G
2	17.3G	5.9M	2.1G	1.6G	684.5M	8G
3	17.8G	6.1M	-	-	-	-
4	1.1G	148.9K	1.3G	958.4M	327.8K	1.7G
5	1.4G	148.9K	-	-	-	-
6	1.1G	148.7K	1.3G	957.3M	328.8K	1.7G
7	17.3G	5.9M	-	-	-	-
8	1.2G	148.9K	-	-	-	-

JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.191

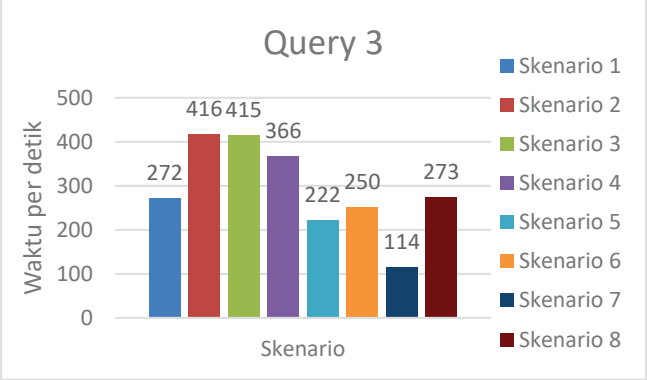
D. *Analisis Hasil Performa*



Gambar 23. Grafik Query 1



Gambar 24. Grafik Query 2



Gambar 25. Grafik Query 3

Pada *query* pertama atau pada Gambar 23, dihasilkan rata – rata waktu *query* sebesar 223.75 detik, dapat dilihat bahwa skenario optimasi ke tujuh menghasilkan waktu tercepat, dengan waktu sebesar 145 detik, jika dibandingkan dengan skenario optimasi lain, dan skenario ke empat, menghasilkan waktu terlama dengan waktu 342 detik. Jika dibandingkan dengan skenario 1 atau skenario tanpa optimasi, skenario 7 mengalami kenaikan waktu sebesar 21%. Skenario tujuh merupakan skenario dengan partisi dan *bucketing*, tetapi tanpa kompresi, sedangkan skenario empat merupakan skenario dengan *bucketing* dan kompresi.

Sedangkan pada *query* kedua atau pada Gambar 24, dihasilkan rata – rata waktu *query* sebesar 433.25 detik, pada *query* kedua, skenario 7 masih menghasilkan waktu *query* paling cepat, yaitu dengan waktu 227 detik, sementara skenario 2 menghasilkan waktu terlama, dengan waktu 554 detik. Pada *query* kedua ini, dari skenario 1 (skenario yang tidak dioptimasi), jika dibandingkan dengan skenario 7 (skenario dengan waktu *query* tercepat), terdapat peningkatan sebesar 7%.

Pada *query* ketiga yang dapat dilihat pada Gambar 25, dihasilkan rata – rata waktu *query* sebesar 291 detik, pada *query* ketiga, skenario 7 menghasilkan waktu *query* tercepat, dengan waktu 114 detik, sementara skenario 2 menghasilkan waktu *query* terlama, dengan waktu 416 detik,
pada *query* ketiga ini, jika dibandingkan antara skenario 1 dan skenario 7, optimasi ini mengalami kenaikan kecepatan sebesar 59%.

Pada ketiga *query* yang telah dijalankan diatas, *query* skenario 7 menghasilkan waktu *query* tercepat jika dibandingkan dengan skenario optimasi lain, hal ini disebabkan oleh skenario 7 yang dilakukan partisi dan *bucketing*, tetapi tidak dilakukan kompresi, karena ketika data pada Hadoop di kompresi dan *query* dijalankan, Hadoop akan membuka kompresi tersebut terlebih dahulu sebelum dapat membaca data yang ada. Namun, jika dilihat dari segi penyimpanan data, data yang tidak dikompresi akan menggunakan penyimpanan yang lebih besar jika dibandingkan dengan data yang dikompresi.

V. KESIMPULAN DAN SARAN

A. *Kesimpulan*

Optimasi yang dilakukan dengan cara analisis penggunaan partisi, *bucketing*, dan kompresi pada penelitian ini telah berhasil.

Perlakuan partisi dan *bucketing* pada tabel akan memberikan waktu *query* yang lebih cepat, hal ini dibuktikan dari hasil skenario 7 (skenario dengan partisi dan *bucketing*) yang dibandingkan dengan skenario 1 (tanpa partisi, tanpa *bucketing*), pada *query* 1, waktu yang diperlukan untuk menyelesaikan *query* pada skenario 7 menjadi lebih cepat 38 detik atau sekitar 20.7% jika dibandingkan dengan skenario 1, begitu juga pada *query* 2, waktu yang diperlukan bertambah cepat sebesar 16 detik atau sekitar 6.5%, sedangkan pada *query* 3, waktu yang diperlukan bertambah cepat sebesar 158 detik atau sekitar 58%.

Kompresi data akan mengecilkan ukuran data pada ruang penyimpanan, jika skenario 4 (skenario dengan kompresi) dibandingkan dengan skenario awal, terdapat penyusutan ukuran sebesar 8 kali hingga 30 kali, tetapi ada kelemahan dari kompresi ini, saat data di kompresi, rata-rata waktu *query* yang dibutuhkan akan menjadi lebih lama, hal ini dibuktikan dari membandingkan skenario 1 (skenario awal), dengan skenario 4 (skenario berupa kompresi saja), pada *query* 1, waktu yang diperlukan oleh skenario 4 untuk menyelesaikan *query* ini melambat sebesar 86% jika dibandingkan dengan skenario 1, sementara pada *query* 2, waktu yang diperlukan untuk menyelesaikan *query* ini melambat sebesar 122%, sedangkan untuk *query* 3, waktu yang diperlukan melambat sebesar 34%.

B. *Saran*

Berdasarakan hasil kesimpulan diatas, berikut saran yang diperoleh dari penelitian ini:

Basis data yang digunakan pada penelitian ini berfokus pada Hive, yaitu basis data yang ditujukan untuk melakukan proses analisis data, terdapat contoh lain basis data pada ekosistem Hadoop, contohnya HBase, HBase berfokus pada *real time querying*, yang ideal digunakan jika memerlukan

data yang acak saat *write* atau *read*, sehingga perlu dilakukan penelitian faktor optimasi lebih lanjut tentang basis data HBase atau basis data lain yang terintegrasi dengan Hadoop, sehingga pada penelitian lain, dapat dipilih basis data yang sesuai dan dibuat secara optimal untuk keperluan pemrosesan data secara efisien dan cepat.

DAFTAR PUSTAKA

[1] A. Abdel, "*Optimizing Join in HIVE Star Schema Using Key/Facts Indexing*", *IETE Technical Review, pp* 132-144,2018.

[2] Siallagan M, Sabariah MK, dan Sontya M. "*Optimasi Query Database Menggunakan Algoritma Genetik*", Jurnal Fakultas Hukum UII,2008.

[3] A. Ahmad, "*Multilevel Data Processing Using Parallel Algorithms for Analyzing Big Data in High-Performance Computing", International J Parallel Prog*,2018.

[4] R. Kaur dan R. Chadha, *"Comparative Analysis of Various File Formats in HIVE", International Journal of Technology and Computing. Vol 3 Issue* 6,2017.

[5] MZN. Saavedra dan S. Yu, *"A Comparison between Text, Parquet, and PCAP Formats for Use in Distributed Network Flow Analysis on Hadoop", Journal of Advances in Computer Network. Vol 5*,2017.

[6] E. Costa, "*Evaluating partitioning and bucketing strategies for Hive-based Big Data Warehousing systems", Journals of Big Data*,2019.

[7] DJ. Dean, *"Efficient Data Retrieval in Big-Data Processing Systems", International Business Machines Corporation*,2017.

[8] AT. Hashem, "*Multi-objective scheduling of MapReduce jobs in big data processing", Multimed Tools Appl*,2018.

[9] W. Li, *"Data mining optimization model for financial management information system based on improved genetic algorithm", Inf Syst E-Bus Manage*,2019.

[10] MM. Rathore, *"Real-Time Big Data Stream Processing Using GPU with Spark Over Hadoop Ecosystem", Int J Parallel Prog*,2018.

[11] P. Basanta-Val, "*An Efficient Industrial Big-Data Engine", IEEE Transactions on Industrial Informatics, pp* 1361-1369,2017.

[12] .Sachin, *"Nuts and Bolts of ETL in Data Warehouse", Emerging Trends in Expert Appliactions and Security Advances in Intelligent Systems and Computing. Volume 841*,2019.

[13] L. Wang, *"High performance cloud computing for remote sensing big data management and processing", Future Generation Computer System, pp* 353-368,2018.

[14] M. White, *"Big Data vs Data Warehousing", International Journal of Scientific Research in Computer Science, Engineering and Information Technoloy. Volume 3 Issue 5*,2018.

[15] K. Rattanaopas, S. Kaewkeerat dan Y. Chuchuen, "*A Comparison of ORC-Compress Performance with Big*

JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.191