TUGAS AKHIR – KI141502

SINKRONISASI BASIS DATA SQL DENGAN BASIS DATA NOSQL MENGGUNAKAN *DATA ADAPTER* DENGAN PENDEKATAN *QUERY DIRECT ACCESS*

I GUSTI NGURAH ADI WICAKSANA
NRP 5113100110

Dosen Pembimbing I
Ir. Muchammad Husni, M.Kom.

Dosen Pembimbing II
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

SINKRONISASI BASIS DATA SQL DENGAN BASIS DATA NOSQL MENGGUNAKAN *DATA ADAPTER* DENGAN PENDEKATAN *QUERY DIRECT ACCESS*

I GUSTI NGURAH ADI WICAKSANA
NRP 5113100110

Dosen Pembimbing I
Ir. Muchammad Husni, M.Kom.

Dosen Pembimbing II
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

SYNCHRONIZATION BETWEEN SQL AND NOSQL DATABASES USING DATA ADAPTER WITH DIRECT ACCESS QUERY APPROACH

I GUSTI NGURAH ADI WICAKSANA
NRP 5113100110

Supervisor I
Ir. Muchammad Husni, M.Kom.

Supervisor II
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

SINKRONISASI BASIS DATA SQL DENGAN BASIS DATA NOSQL MENGGUNAKAN *DATA ADAPTER* DENGAN PENDEKATAN *QUERY DIRECT ACCESS*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Arsitektur dan Jaringan Komputer
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
I GUSTI NGURAH ADI WICAKSANA
NRP : 5113 100 110

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Ir. Muchammad Husni, M.Kom .....
   NIP: 19600221 198403 1 001 (Pembimbing 1)

2. Henning Titi Ciptaningtyas, S.Kom, M.Kom .....
   NIP: 19840708 201012 2 004 (Pembimbing 2)

SURABAYA
JULI, 2017

[Halaman ini sengaja dikosongkan]

SINKRONISASI BASIS DATA SQL DENGAN BASIS DATA NOSQL MENGGUNAKAN *DATA ADAPTER* DENGAN PENDEKATAN *QUERY* DIRECT ACCESS

Nama Mahasiswa	**: I Gusti Ngurah Adi Wicaksana**
NRP	**: 5113100110**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Ir. Muchammad Husni, M.Kom.**
Dosen Pembimbing 2	**: Henning Titi Ciptaningtyas, S.Kom., M.Kom.**

Abstrak

Basis data NoSQL, singkatan dari Not Only SQL, semakin banyak digunakan seiring dengan bertambahnya jumlah aplikasi big data. Kebanyakan sistem masih menggunakan relational databases (RDB), namun seiring dengan bertambahknya jumlah data tiap tahunnya, sistem menangani big data dengan basis data NoSQL untuk menganalisis dan mengakses data dengan lebih cepat. NoSQL muncul sebagai akibat dari pertumbuhan eksponensial dari internet dan perkembangan aplikasi web.

Sintaks query pada basis data NoSQL berbeda dengan basis data SQL, sehingga diperlukan perubahan kode pada aplikasi. Dibandingkan merubah kode sumber atau merubah basis data SQL menjadi NoSQL, penulis melakukan riset untuk mengintegrasikan kedua basis data. Data adapter memungkinkan aplikasi untuk tidak merubah sintaks query SQL-nya. Data adapter menyediakan metode yang dapat melakukan sinkronisasi basis data SQL dengan basis data NotSQL. Selain itu, data adapter menyediakan antar muka yang bisa diakses aplikasi untuk menjalankan query SQL. Oleh karena itu, dalam pengerjaan Tugas Akhir ini diterapkan sistem data adapter untuk melakukan sinkronisasi antara basis data SQL dengan NoSQL menggunakan pendekatan query direct

access, dimana sistem memungkinkan aplikasi untuk menerima query ketika proses sinkronisasi sedang berlangsung.

Dari hasil uji coba dengan menggunakan data adapter, didapatkan hasil jika data adapter dapat melakukan sinkronisasi antara basis data SQL, yaitu MySQL, dengan NoSQL, yaitu Apache HBase. Sistem ini menghabiskan persentase sumber daya memori pada kisaran 1.736,6 MB sampai dengan 2.431,8 MB dan persentase processor yang bergerak dari 14,1% sampai dengan 38,9%. Selain itu, dari sistem ini juga didapatkan hasil performa basis data NoSQL yang lebih baik dibandingkan dengan basis data SQL.

Kata kunci: RDB, NoSQL, Data adapter, Sinkronisasi

SYNCHRONIZATION BETWEEN SQL AND NOSQL DATABASE USING *DATA ADAPTER* WITH DIRECT ACCESS QUERY APPROACH

Nama Mahasiswa	**: I Gusti Ngurah Adi Wicaksana**
NRP	**: 5113100110**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Ir. Muchammad Husni, M.Kom.**
Dosen Pembimbing 2	**: Henning Titi Ciptaningtyas, S.Kom., M.Kom.**

Abstract

NoSQL databases, or Not Only SQL, are increasingly being used as the number of big data applications increases. Most systems still use relational databases (RDBs) nowadays, but as the number of data increases each year, the system handles big data with NoSQL databases to analyze and access data more quickly. NoSQL emerged as a result of the exponential growth of the internet and the development of web applications

The query syntax in the NoSQL database differs from the SQL database, therefore developer need to changes the code of application. Instead of change the source code or change the database from SQL to NoSQL, the author do research to integrate this two databases. Data adapter allow applications to not change their SQL query syntax. Data adapters provide methods that can synchronize SQL databases with NotSQL databases. In addition, the data adapter provides an interface which is application can access to run SQL queries. Hence, this undergraduated thesis applied data adapter system to synchronize data between MySQL database and Apache HBase using direct access query approach, where system allows application to accept query while synchronization process in progress.

From the test performed using data adapter, the results obtained that the data adapter can synchronize between SQL databases, MySQL, and NoSQL database, Apache HBase. This system spends the percentage of memory resources in the range of 1.990,0 MB to 3.523,8 MB, and the percentage of processor moving from 21,3% to 51,3%. In addition, from this system also obtained the performance of database NoSQL better than SQL database.

Keywords: RDB, NoSQL, Data adapter, Synchronization

KATA PENGANTAR

Puji syukur penullis panjatkan kepada Tuhan Yang Maha Esa, Ida Sang Hyang Widhi Wasa, yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"**Sinkronisasi Basis Data SQL dengan Basis Data NoSQL Menggunakan *Data adapter* dengan Pendekatan *Query Direct Access***".**

Pengerjaan Tugas Akhir ini adalah kesempatan yang sangat berharga bagi penulis, terutama untuk lebih banyak memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Tanpa mengurangi rasa hormat, penulis ingin mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa, Ida Sang Hyang Widhi Wasa atas berkah yang tiada habisnya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Kedua orang tua penulis, I Gusti Ngurah Putu Widyaarta dan I Gusti Ayu Suryaningsih yang telah memberikan dukungan moral dan material serta doa yang tak terhingga untuk penulis. Begitu pula adik penulis, I Gusti Ayu Arista Putri yang selalu memberikan semangat ketika penulis mengerjakan Tugas Akhir ini.
3. Bapak Ir. Muchammad Husni, M.Kom. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.

5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Teman-teman Administrator Laboratorium Arsitektur dan Jaringan Komputer (AJK), Daniel, Uul, Setiyo, Nindy, Zaza, Risma, Syukron, Fatih, Oing, Ambon, Thoni, Bebet, Vivi, Awan, Didin, Satria, dan Fuad yang selalu menghibur dan mendukung penulis dalam pengerjaan Tugas Akhir ini.
7. Mas Sam, Mas Rohmen, dan Kak Surya yang selalu memberikan dukungan, bimbingan, dan ilmunya dalam mengerjakan Tugas Akhir ini.
8. Bu Diana, Bu Yuhana, Pak Royyana, dan Pak Affandi yang selalu memberikan motivasi untuk menyelesaikan Tugas Akhir ini.
9. Teman-teman 'Piranha Academy' (a.k.a Uwus), Dwi, Igun, Iwan, Semara, Kusnanta, Dewak, dan Wahyu yang selalu memberikan keceriaan selama pengerjaan Tugas Akhir ini.
10. Teman-teman TC angkatan 2013 yang yang telah berbagi ilmu, dan memberi motivasi kepada penulis.
11. Rekan-rekan Tim Pembina Kerohanian Hindu (TPKH) ITS yang telah memberikan dukungan, dan semangat kepada penulis untuk menyelesaikan Tugas Akhir ini.
12. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2017

I Gusti Ngurah Adi Wicaksana

DAFTAR ISI

LEMBAR PENGESAHAN v
Abstrak vii
Abstract ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xv
DAFTAR TABEL xvii
DAFTAR KODE SUMBER xix
DAFTAR PSEUDOCODE xxi
BAB 1 BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Metodologi 3
1.7 Sistematika Penulisan Laporan Tugas Akhir 5
BAB 2 BAB II TINJAUAN PUSTAKA 7
2.1 Big Data 7
2.2 Basis Data 9
2.3 Basis Data Relasional 10
2.4 Basis Data NoSQL 11
2.5 Data Adapater 14
2.6 MySQL 15
2.7 Apache HBase 16
2.8 Apache Phoenix 19
2.9 Python 20
2.10 Flask 21
BAB 3 BAB III DESAIN DAN PERANCANGAN 23
3.1 Kasus Pengguna 23
3.2 Arsitektur Sistem 25
3.2.1 Desain Umum Sistem 25

3.2.2 Desain *Data adapter* 27
3.2.3 Desain Basis Data SQL 28
3.2.4 Desain Basis Data NoSQL 29
3.3 Proses Transformasi dan Sinkronisasi 30
3.4 Desain Antarmuka 33
BAB 4 BAB IV IMPLEMENTASI 35
4.1 Lingkungan Implementasi 35
4.2 Rincian Implementasi Server Basis Data 36
4.2.1 Instalasi Server Basis Data SQL 36
4.2.2 Instalasi Server Basis Data NoSQL 37
4.3 Rincian Implementasi Sistem *Data adapter* 43
4.3.1 Instalasi Paket 43
4.3.2 Implementasi DB Adapter 44
4.3.3 Implementasi DB Converter 50
BAB 5 BAB V UJI COBA DAN EVALUASI 57
5.1 Lingkungan Uji Coba 57
5.2 Dataset Uji Coba 60
5.3 Skenario Uji Coba 61
5.3.1 Skenario Uji Coba Fungsionalitas 62
5.3.2 Skenario Uji Coba Kapasitas dan Performa 66
5.4 Hasil Uji Coba dan Evaluasi 67
5.4.1 Hasil Uji Coba Fungsionalitas 67
5.4.2 Hasil Uji Coba Kapasitas dan Performa 76
5.4.3 Evaluasi 87
BAB 6 BAB VI KESIMPULAN DAN SARAN 89
6.1 Kesimpulan 89
6.2 Saran 90
DAFTAR PUSTAKA 91
LAMPIRAN A KODE SUMBER 95
LAMPIRAN B GAMBAR PENDUKUNG 111
BIODATA PENULIS 113

DAFTAR GAMBAR

Gambar 2.1 Diagram Tiga 'V' Big Data 9
Gambar 2.2 Terminologi Basis Data Relasional 10
Gambar 2.3 Tipe Basis Data NoSQL 13
Gambar 2.4 Contoh Sintaks SQL 16
Gambar 2.5 Perintah Dasar pada HBase Shell 18
Gambar 2.6 Skema penyimpanan pada Apache Hbase..... 18
Gambar 2.7 Contoh tampilan data yang disimpan dalam Hbase jika diakses melalui terminal 18
Gambar 2.8 Contoh Program Sederhana Menggunakan Flask 22
Gambar 3.1 Diagram Kasus Pengguna 23
Gambar 3.2 Desain Sistem *Data adapter* Secara Umum..... 26
Gambar 3.3 Desain Arsitektur *Data adapter* 28
Gambar 3.4 Diagram Arsitektur Basis Data SQL 29
Gambar 3.5 Diagram Arsitektur Basis Data NoSQL 30
Gambar 3.6 Diagram Alir Proses Sinkronisasi 31
Gambar 3.7. Diagram Alir Proses Mengambil Query dari Basis Data MySQL 32
Gambar 3.8 Diagram Alir Konversi Sintaks Query MySQL ke Query Apache Phoenix 33
Gambar 3.9 Diagram Alir Antarmuka *Data adapter* 34
Gambar 4.1 Diagram Alir DB Converter 51
Gambar 5.1 Desain Arsitektur Uji Coba Sistem 57
Gambar 5.2 Tampilan pada Terminal Ketika Proses Inisialisasi Berlangsung 68
Gambar 5.3 Tampilan pada Terminal Ketika Proses Transformasi Berlangsung 69
Gambar 5.4 Grafik Perbandingan Waktu Respon Rata-rata Basis Data SQL (MySQL) dengan NoSQL (HBase) 76
Gambar 5.5 Grafik Waktu Inisialisasi Berdasarkan Perbandingan Jumlah Baris 78
Gambar 5.6 Grafik Penggunaan Memori Berdasarkan Perbandingan Jumlah Baris 79

Gambar 5.7 Grafik Persentase Penggunaan Processor Berdasarkan Perbandingan Jumlah Baris 79
Gambar 5.8 Grafik Waktu Transformasi Berdasarkan Perbandingan Jumlah Tabel 81
Gambar 5.9 Grafik Penggunaan Memori Berdasarkan Perbandingan Jumlah Tabel 82
Gambar 5.10 Grafik Persentase Penggunaan Processor Berdasarkan Perbandingan Jumlah Tabel 82
Gambar 5.11 Grafik Perbandingan Waktu Transformasi pada Transformasi Query Insert, Update dan Delete 84
Gambar 5.12 Grafik Perbandingan Penggunaan Memori pada Transformasi Query Insert, Update dan Delete 85
Gambar 5.13 Grafik Perbandingan Persentase Processor pada Transformasi Query Insert, Update dan Delete 85
Gambar B. 1 Contoh Tangkapan Layar Uji Coba Performa Perbandingan Jumlah Baris 111
Gambar B. 2 Contoh Tangkapan Layar Uji Coba Performa Perbandingan Jumlah Tabel 111
Gambar B. 3 Contoh Tangkapan Layar Uji Coba Performa Transformasi Query Insert 111
Gambar B. 4 Contoh Tangkapan Layar Uji Coba Performa Transformasi Query Delete 112
Gambar B. 5 Contoh Tangkapan Layar Uji Coba Performa Transformasi Query Update 112

DAFTAR TABEL

Tabel 2.1 Beberapa Perbedaan Sintaks Antara Apache Phoenix dengan MySQL 20
Tabel 3.1 Penjelasan Diagram Kasus Pengguna 24
Tabel 4.1 Perangkat Lunak yang Digunakan 35
Tabel 4.2 Implementasi Rute Pada DB Adapter 44
Tabel 4.3 Daftar Sintaks SQL pada Setiap Rute 47
Tabel 5.1 Spesifikasi Server Basis Data SQL 58
Tabel 5.2 Spesifikasi Server Basis Data NoSQL 58
Tabel 5.3 Spesifikasi Server *Data adapter* 59
Tabel 5.4 Spesifikasi Komputer Penguji 59
Tabel 5.5 Aksi dan Hasil Harapan Setiap Fungsi Antar Muka 63
Tabel 5.6 Hasil Uji Coba Fungsionalitas 70
Tabel 5.7 Hasil Waktu Respons Uji Coba Fungsionalitas untuk Perubahan Data 72
Tabel 5.8 Waktu Respon Uji Fungsionalitas Proses Pengambilan Data pada Basis Data MySQL 73
Tabel 5.9 Waktu Respon Uji Fungsionalitas Proses Pengambilan Data pada Basis Data HBase 74
Tabel 5.10 Hasil Uji Kapasitas dan Performa pada Komputer *Data adapter* (DB Converter) dengan Perbandingan Jumlah Baris 77
Tabel 5.11 Hasil Uji Kapasitas dan Performa pada Komputer *Data adapter* (DB Converter) dengan Perbandingan Jumlah Tabel 80
Tabel 5.12 Hasil Uji Kapasitas dan Performa pada Komputer *Data adapter* (DB Converter) dengan Perbandingan Jumlah Query Insert 83
Tabel 5.13 Hasil Uji Kapasitas dan Performa pada Komputer *Data adapter* (DB Converter) dengan Perbandingan Jumlah Query Update 83

Tabel 5.14 Hasil Uji Kapasitas dan Performa pada Komputer *Data adapter* (DB Converter) dengan Perbandingan Jumlah Query Delete 84

DAFTAR KODE SUMBER

Kode Sumber 4.1 Konfigurasi MySQL pada Berkas my.cnf . 37
Kode Sumber 4.2 Konfigurasi pada Berkas .bashrc 40
Kode Sumber 4.3 Variable JAVA_HOME pada hadoop-env.sh 40
Kode Sumber 4.4 Konfigurasi pada Berkas core-site.xml 41
Kode Sumber 4.5 Konfigurasi pada Berkas mapred-site.xml 41
Kode Sumber 4.6 Konfigurasi pada Berkas hdfs-site.xml 42
Kode Sumber A. 1 Sinkronisasi 107
Kode Sumber A. 2 Converter Sintaks MySQL ke Sintaks Phoenix 109

[Halaman ini sengaja dikosongkan]

DAFTAR PSEUDOCODE

Pseudocode 4.1 Eksekusi Query untuk Proses Perubahan Data. 49
Pseudocode 4.2 Eksekusi Query untuk Proses Pengambilan Data 50
Pseudocode 4.3 Proses Sinkronisasi 53
Pseudocode 4.4 Konverter Sintaks Query Insert MySQL ke Sintaks Query Apache Phoenix..... 54
Pseudocode 4.5 Konverter Sintaks Query Update ke Sintaks Query Apache Phoenix 55

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Big data dan *hybrid database* menjadi semakin banyak digunakan seiring dengan berkembangnya servis *cloud computing*. The United Nations Economic Commission for Europe, atau UNECE, memprediksi jika peningkatan jumlah data akan mencapai 350% pada tahun 2019 jika dibandingkan dengan tahun 2015. [1] Basis data NoSQL, singkatan dari *Not Only SQL*, semakin banyak digunakan seiring dengan bertambahnya jumlah aplikasi *big data*.

Kebanyakan sistem dewasa ini masih menggunakan *relational database* (RDB). MySQL adalah salah satu *Database Management System* (DBMS) populer untuk aplikasi berbasis web. Dengan performanya yang terpercaya, keandalan, dan penggunaan yang mudah, MySQL telah menjadi basis data pilihan utama yang digunakan oleh perusahaan dengan profil tinggi seperti Facebook, Twitter, dan Youtube. [2] Tetapi seiring dengan jumlah data yang terus bertambah tiap tahunnya, sistem menangani *big data* dengan basis data NoSQL untuk menganalisis dan mengakses data dengan lebih cepat. NoSQL muncul sebagai akibat dari pertumbuhan eksponensial dari internet dan perkembangan aplikasi web. NoSQL mendukung pengembangan yang gesit, karena NoSQL adalah *schema-less* dan tidak tidak perlu mendefinisikan secara statik bagaimana data harus dimodelkan. [3]

Sistem yang menggunakan basis data NoSQL, ketika meng-*query* data tidak menggunakan sintaks yang sama seperti mengambil data pada basis data SQL karena sintaks *query* yang digunakan berbeda. Dibandingkan merubah kode sumber atau merubah basis data SQL menjadi NoSQL, penulis melakukan riset untuk mengintegrasikan kedua basis data. Aplikasi terkoneksi dengan SQL untuk menangani jumlah data dengan skala kecil dan menengah, server basis data NoSQL sebagai *back-end* sistem

untuk menganalisis data dan melakukan sejumlah operasi *read/write*, atau secara periodik melakukan *backup* data dari SQL.

Integrasi basis data dapat mempengaruhi desain sistem yang asli. Dalam sistem, aplikasi memperoleh data dari basis data relasional menggunakan *query* SQL, namun dalam NoSQL tidak dapat diakses dengan menggunakan SQL. Oleh karena itu aplikasi harus merubah desain untuk dapat mengakses basis data RBD dan NoSQL. Mekanisme transformasi data dari SQL ke NoSQL diperlukan ketika mengintegrasikan sistem dengan NoSQL. Selama proses transformasi, aplikasi dituntut untuk menunggu hingga proses sinkronisasi selesai. Proses transformasi ini dapat berlangsung sangat lama karena jumlah data yang diubah dalam skala besar. Hal ini menjadi permasalahan utama untuk beberapa kasus seperti *real-time* atau layanan *non-stop* seperti analisis ilmiah dan aplikasi web.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana cara mengimplementasikan *data adapter* untuk mensinkronisasikan SQL dan NoSQL?
2. Bagaimana hasil implementasi *data adapter* dalam sinkronisasi basis data SQL dan NoSQL?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan yaitu sebagai berikut:

1. Jumlah server yang digunakan untuk sistem *data adapter* adalah tiga komputer, masing-masing untuk basis data SQL, basis data NoSQL dan *data adapter*.
2. Perangkat lunak yang digunakan adalah MySQL sebagai basis data SQL, Apache HBase sebagai basis data NoSQL dan Apache Phoenix sebagai translator.

1.4 Tujuan

Tujuan Tugas Akhir ini adalah sebagai berikut:

1. Menyamakan isi data pada basis data SQL dan NoSQL dalam permasalahan ekspansi penggunaan basis data menggunakan mekanisme *data adapter*.
2. Mengetahui performa dari *data adapter* dalam melakukan sinkronisasi basis data SQL dan NoSQL.

1.5 Manfaat

Manfaat yang diperoleh dari pembuatan Tugas Akhir ini adalah data pada basis data NoSQL memiliki konten yang sama dengan data pada basis data SQL setelah melalui proses sinkronisasi menggunakan sistem *data adapter* dengan pendekatan *query direct access*.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Penyusunan proposal Tugas Akhir dilaksanakan untuk merumuskan masalah serta melakukan penetapan rancangan dasar dari sistem yang akan dikembangkan dalam pelaksanaan Tugas Akhir ini.

2. Studi literatur
   Pada tahap ini dilakukan pencarian informasi dan literatur dari paper dan artikel di internet yang diperlukan untuk tahap implementasi program. Tahap ini diperlukan untuk membantu memahami konsep dan penggunaan komponen-komponen terkait dengan sistem yang akan dibangun, diantaranya *data adapter*, basis data MySQL dan Apache Hbase.

3. Desain dan Perancangan
   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain fungsi yang akan dibuat yang ditunjukkan melalui *pseudocode*.

4. Implementasi perangkat lunak
   Implementasi perangkat lunak merupakan tahap membangun rancangan program yang telah dibuat. Pada tahap ini akan direalisasikan mengenai rancangan apa saja yang telah didefiniskan pada tahap sebelumnya. Fungsi yang ada pada tahap ini merupakan fungsi hasil implementasi dari tahap analisis dan perancangan perangkat lunak.

5. Pengujian dan evaluasi
   Pada tahap ini dilakukan pengujian dan pencatatan hasil dari metode yang diimplementasikan. Pengujian akan dilakukan di Laboratorium Arsitektur dan Jaringan Komputer. Parameter yang akan diujikan adalah berapa waktu yang dibutuhkan untuk melakukan sinkronisasi basis data antara RDB dan HBase NoSQL database sehingga dapat diketahui efektifitas implementasi sistem *data adapter*. Banyak komputer server basis data yang digunakan adalah dua komputer.

6. Penyusunan buku Tugas Akhir
   Pada tahap ini disusun buku yang memuat dokumentasi mengenai perancangan, pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu perumusan masalah, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Desain dan Perancangan

Bab ini berisi tentang dasar dari algoritma yang akan diimplementasikan pada Tugas Akhir ini.

Bab IV Implementasi

Bab ini membahas mengenai implementasi dari rancangan yang telah dibuat pada bab sebelumnya.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan mengenai kemampuan sistem dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang telah dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Big Data

Setiap hari, manusia menghasilkan 2,5 triliun *byte* data, dan 90% dari semua data yang ada di dunia saat ini muncul pada dua tahun terakhir. Data ini berasal dari mana-mana: sensor yang digunakan untuk menangkap informasi perubahan cuaca, sejumlah post di media sosial, gambar digital dan video, catatan transaksi penjualan, dan dan data sinyal GPS dari perangkat bergerak. Jumlah yang sangat besar dan diproduksi secara terus-menerus ini dikenal dengan sebutan *Big Data*. Sejalan dengan peningkatan jumlah data ini, teknik yang ada sekarangpun menjadi semakin kuno. Perlu adanya keterampilan membuat kode yang komprehensif serta penguasaan pengetahuan dan statistik yang baik. [4]

Istilah "*Big Data*" muncul seiring dengan berkembangnya proses industri, personil dan teknologi untuk mendorong bidang baru apa yang tampaknya akan meledak. Perusahaan besar seperti Amazon dan Wal-Mart serta badan lain seperti pemerintahan Amerika Serikat dan NASA menggunakan *big data* untuk memenuhi tujuan bisnis dan strategi mereka. *Big data* dapat memainkan peran bagi perusahaan skala menengah keatas dan organisasi yang dimanfaatkan agar mendapatkan keuntungan.

Tidak ada hal yang baru terkait dengan gagasan *big data*, yang mana telah ada sejak tahun 2001. *Big data* adalah informasi

yang dimiliki perusahaan, didapatkan dan diproses dengan teknik baru untuk menghasilkan nilai dengan cara terbaik. [5]

Big data dan *hybrid database* menjadi semakin pupoler seiring dengan berkembangnya servis *cloud computing*. The United Nations Economic Commission for Europe, atau UNECE, memprediksi jika peningkatan jumlah data akan mencapai 350% pada tahun 2019 jika dibandinkan dengan tahun 2015. [1] Pada tahun 2001, Gartner's Doug Laney pertama kalinya memperkenalkan istilah yang dikenal dengan "Tiga V" untuk menggambarkan beberapa karakteristik yang membuat *big data* berbeda dengan pemrosesan data yang lain. 'Tiga' V itu ialah *Volume*, *Velocity*, dan *Variety*. [6]

Volume, atau kapasitas, mengacu pada jumlah data yang tersedia untuk dianalisis. Namun ketika data terkumpulkan dari pertumbuhan jumlah perangkat, seperti sensor, telepon seluler dan komputer, jumlah data yang terakumulasikan menjadi sangat banyak. Namun sudah ada beberapa teknologi, seperti Hadoop, yang memudahkan beban ini.

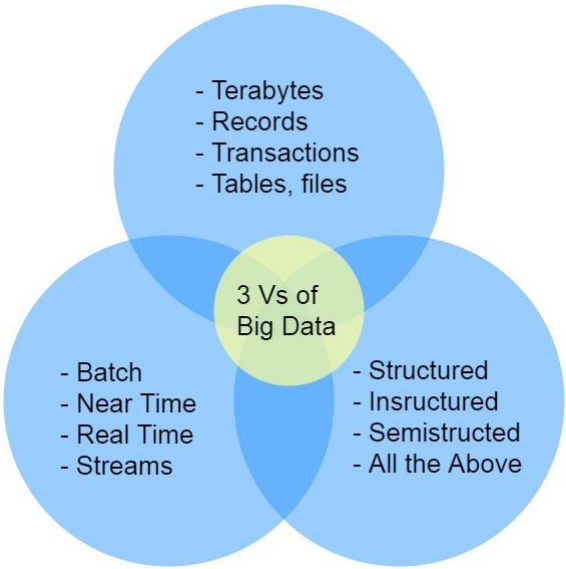
Velocity, atau kecepatan, mengacu pada kecepatan pengolahan data. Karena data datang dari berbagai sumber, hal itu bisa terkumpulkan dengan cepat. Tantangannya adalah mengumpulkannya secepat dan seefisien mungkin.

Variety, atau keberagaman, mengacu pada banyaknya tipe format data. Contohnya seperti tipe data terstruktur, numerik, dokumen, suara, video, *e-mail* dan data transaksi penjualan. Tantangannya adalah membandingkan dan membedakan banyak data sedemikian rupa sehingga menjadi berpola dan dapat dimanfaatkan. [7]

Hal terpenting dari big data adalah potensi untuk meningkatkan efisiensi dalam konteks volume data yang sangat besar, dari tipe data yang berbeda. Jika big data digunakan dengan tepat, suatu instansi dapat memiliki bayangan untuk bisnis mereka, oleh karena itu akan mengarah pada efisiensi di berbagai bidang seperti penjualan, peningkatan produk manufaktur dan sebagainya.

Big data dapat diterapkan secara efektif pada beberapa area, contohnya adalah sebagai berikut:

- Dalam teknologi informasi digunakan untuk meningkatkan keamanan dan penanganan masalah dengan menganalisis pola dari log yang ada.
- Meningkatkan servis dan produk berdasarkan penggunaan konten media sosial. Dengan mengetahui preferensi pelanggan, perusahaan dapat merubah produknya agar dapat menyebar ke orang yang lebih banyak. [8]



Gambar 2.1 Diagram Tiga 'V' Big Data

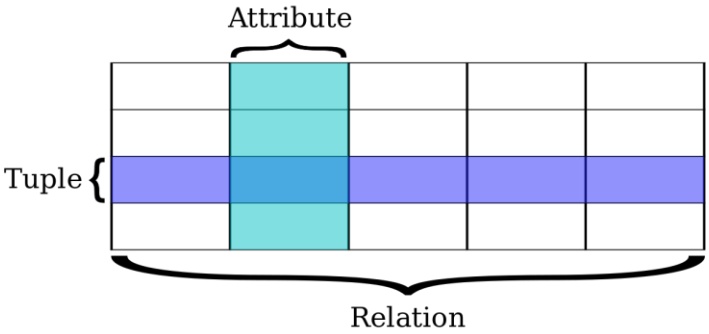
2.2 Basis Data

Basis data atau *database* merupakan sebuah koleksi atau kumpulan dari data yang bersifat mekanis, terbagi, terdefinisi secara formal serta terkontrol. Pengontrolan dari sistem *database*

tersebut adalah terpusat, yang biasanya dimiliki dan juga dipegang oleh suatu organisasi

Bentuk sebuah basis data, elektronik dan lainnya, harus di rencanakan. Proses dalam merancang basis data adalah aktivitas dalam merepresentasikan kelas, atribut, dan relasi antar basis data. Data adalah fakta. Informasi menggambarkan data. Informasi adalah data dengan konteks yang berarti. [9]

2.3 Basis Data Relasional



Gambar 2.2 Terminologi Basis Data Relasional

Relational database, atau disingkat RDB, menggambarkan suatu kumpulan dari banyak relasi. Konsep basis data relasional, bahasa indonesia dari *relational database*, pertama kali diperkenalkan oleh Dr. Codd pada tahun 1970. Sebuah sistem yang mengatur hal ini disebut dengan *Relational Database Management System* (RDBMS). Model relasional terdiri dari beberapa komponen, diantaranya kumpulan objek berelasi, kumpulan operasi yang bekerja pada relasi, dan integritas data.

Sebagian besar *database* yang digunakan pada aplikasi modern saat ini adalah relasional *database*. Basis data relasional adalah model *database* yang menyimpan data pada tabel. Setiap tabel terdiri dari baris (*record*) dan kolom (*field*). Dalam terminologi ilmu komputer, baris sering disebut dengan "*tuples*",

dan kolom dapat disebut dengan "*attribute*". Sebuah tabel dapat divisualisasikan sebagai sebuah matriks baris dan kolom, dimana setiap persimpangan dari baris dan kolom berisi nilai tertentu. Hal ini adalah relasional selama semua *record* berbagi bidang yang sama. Gambar 2.2 menunjukan terminologi basis data relasional.

Tabel juga sering terdapat *primary key*, yang menyediakan tanda pengenal unik untuk setiap baris dalam tabel. Kunci ini bisa ditunjuk ke kolom, atau dapat terdiri dari beberapa kolom yang bersama-sama membentuk kombinasi unik dari beberapa nilai. Disisi lain, *primary key* menyediakan cara yang efisien untuk pengindeksan dan dapat digunakan untuk berbagi nilai antar tabel dalam database. Sebagai contoh, nilai *primary key* dari satu tabel dapat digunakan pada baris pada tabel lainnya. Nilai yang disisipkan ini tabel lain ini disebut dengan *foreign key.*

Cara yang digunakan untuk mengakses data pada basis data relasional adalah dengan menggunakan *query* SQL (*Structured Query Language)*. *Query* SQL dapat digunakan untuk membuat, memodifikasi, dan menghapus tabel, serta memilih, *insert*, dan menghapus data dari tabel yang ada. [10]

2.4 Basis Data NoSQL

Basis data NoSQL, kependekan dari *Not Only SQL*, adalah sebuah pendekatan untuk manajemen data dan desain *database* yang berguna untuk kumpulan distribusi data yang sangat besar. NoSQL, meliputi berbagai teknologi dan arsitektur, berusaha untuk memecahkan masalah skalabilitas dan permasalahan performa big data pada kinerja database relasional. NoSQL sangat berguna ketika suatu perusahaan perlu untuk mengakses dan menganalisis data yang tidak terstruktur dalam jumlah yang sangat besar atau data yang disimpan dari jarak jauh pada beberapa server virtual. [11]

NoSQL mencakup berbagai teknologi basis data yang berbeda dan dikembangkan untuk menangani permintaan dalam membangan aplikasi modern:

- Pengembang bekerja dengan aplikasi yang membuat volume baru yang sangat besar, dengan cepat melakukan perubahan tipe data (terstruktur, tidak terstruktur, semi terstruktur dan polimorfik)
- Lama pengerjaan siklus pengembangan *waterfall* adalah dua sampai delapan belas bulan. Dijaman sekarang, tim bekerja dengan secepat-cepatnya, mengiterasi dan mem-*push* kode hampir setiap minggu bahkan setiap hari.
- Aplikasi yang pernah dibuat untuk khalayak tertentu, kini dibuat menjadi servis yang harus selalu menyala, dapat diakses dari berbagai perangkat dalam skala global oleh jutaan pengguna.
- Perusahaan kini beralih ke *scale-out architecture* menggunakan perangkat lunak *open source*, server komoditas dan komputasi awan ketimbang monolotik server dan infrastruktur penyimpanan.

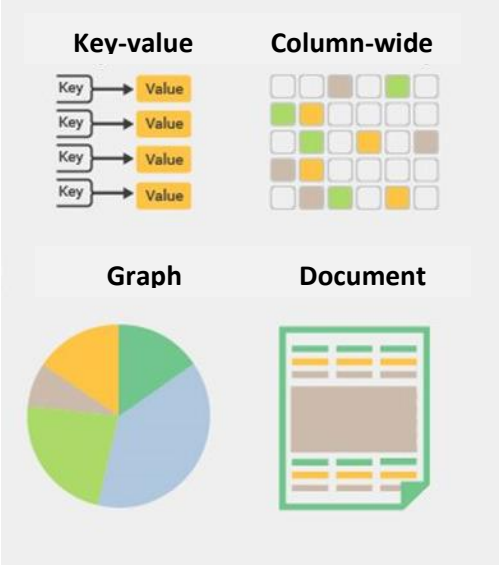
Basis data relasional tidak dirancang untuk mengatasi permasalahan dengan tantangan skala dan kegesitan yang dihadapi aplikasi modern. Basis data relasional juga tidak dibangun untuk mendapatkan keuntungan dari penyimpanan komoditas dan kemampuan pemrosesan yang tersedia saat ini. [12]

Basis data NoSQL adalah teknologi penyimpanan data yang lebih bervariasi sehingga sulit untuk membuat karakteristik mereka menjadi general. NoSQL memberikan performa dan skalabilitas yang lebih baik jika dibandingkan dengan basis data relasional. [13] Ada beberapa tipe basis data NoSQL yang ditunjukan pada Gambar 2.3. Penjelasan tipe-tipe basis data NoSQL adalah sebagai berikut:

3. *Document*

   Document dikenal dengan memasangkan tiap *key*-nya dengan struktur data yang kompleks. *Document* dapat terdiri dari banyak pasangan *key-value* yang berbeda, atau pasangan *key-array*, atau bahkan dokumen bersarang (*nested document*)

4. *Graph*
   Digunakan untuk menyimpan informasi tentang jaringan seperti koneksi sosial (social connection). Yang termasuk graph diantaranya Neo4J dan HyperGraphDB.
5. *Key-value*
   Key-value adalah basis data NoSQL paling sederhana. Setiap item didalam basis data disimpan dengan atribut nama (atau 'kunci'), bersama dengan nilainya. Contoh dari key-value adalah Riak dan Voldemort. Beberapa penyimpanan key-value, seperti Redis, memungkinkan setiap nilai memiliki tipe data, seperti integer, yang dapat menambahkan fungsionalitas.
6. *Wide-column*
   Tipe *wide-column*, seperti Cassandra dan HBase, adalah mengoptimasi *query* dengan kumpulan data yang besar, dan menyimpan data kolum secara bersama, termasuk barisnya.



Gambar 2.3 Tipe Basis Data NoSQL

2.5 Data Adapater

Data adapter sangat termodulasi, yang mana terletak diantara aplikasi dengan basis data. *Data adapter* bertanggung jawab untuk menerima *query* dari aplikasi dan mentransformasi data antar basis data dalam suatu waktu. Sistem menyajikan antarmuka parsing *query* SQL untuk mengakses RDB dan NoSQL.

Data adapter menawarkan mekanisme untuk mengontrol proses transformasi basis data dan memperbolehkan aplikasi untuk melakukan *query* meskipun data (tabel) yang dimaksud sedang transformasi atau tidak. Setelah data ditransformasi, *data adapter* menyediakan mekanisme penambalan untuk mensinkronisasikan ketidak konsistenan data pada tabel. *Data adapter* menawarkan *non-stoping service* selama proses transformasi berlangsung.

Terdapat tiga fitur utama pada *data adapter*. Yang pertama adalah *data adapter* menyediakan antarmuka SQL untuk basis data RDB dan NoSQL. Di dalamnya terdiri atas Apache Phoenix sebagai penerjemah SQL untuk terhubung dengan HBase, dan MySQL JDBC driver untuk menghubungkan dengan RDB yaitu MySQL. Dengan ini aplikasi tidak perlu mengubah *query* untuk menangani *query* NoSQL.

Berikutnya adalah *Database Converter* yang digunakan untuk menangani transformasi basis data. *Database Converter* merubah data dari MySQL ke HBase menggunakan Apache Sqoop dan Apache Phoenix. Mekanisme ini mensinkronisasi data setelah transformasi selesai dilakukan dengan penambalan blok *query*.

Fitur terakhir adalah *Query Approach* atau pendekatan *query*. Pendekatan *query* yang digunakan adalah *direct access (DA mode)*. Pendekatan ini memperbolehkan aplikasi untuk melakukan *query* secara langsung ke basis data tanpa harus menunggu proses transformasi selesai dilakukan atau mengambil data pada server lain. [14]

Pada pengerjaan Tugas Akhir ini, *data adapter* menjadi komponen utama untuk melakukan sinkronisasi antara basis data SQL dan NoSQL. Tipe transformasi yang digunakan adalah

dimana pada kedua basis data, SQL dan NoSQL, memiliki salinan data yang sama, dan arah transformasi adalah dari basis data SQL ke NoSQL.

2.6 MySQL

MySQL adalah aplikasi manajemen basis data, terjemahan dari *Database Management System* (DBMS), *open-source* yang paling terkenal di seluruh dunia. MySQL termasuk kedalam basis data relasional. Dengan performa yang terbukti, kehandalan dan penggunaanya yang mudah, membuat MySQL menjadi basis data dengan pilihan terdepan untuk aplikasi bebrbasis web. MySQL telah digunakan oleh beberapa situs bersar dunia seperti Facebook, Twitter, Youtube, Yahoo dan masih banyak lagi. Oracle menjadi penggerak inovasi MySQL, menyajikan kapabilitas baru untuk memperkuat web, *cloud*, *mobile* dan *embedded application* dimasa mendatang. [15]

Sebagai basis data yang relasional, MySQL menyimpan data dalam tabel yang terpisah dibandingkan dengan menyimpannya kedalam satu tabel. Model logika, dengan objek seperti basis data, tabel, *view*, baris dan kolom, menawarkan lingkungan pemrograman yang fleksibel. Basis data relasional memiliki pengaturan dasar diantara antara tabel yang berbeda seperti *one-to-one, one-to-many*, unik, diperlukan atau opsional, dan 'pointer' di antara tabel yang berbeda. Basis data dengan pengaturan seperti ini dirancang agar aplikasi tidak mengakses data yang tidak konsisten, duplikat, orphan, *out-of-date*, atau data yang hilang. [15]

MySQL memiliki beberapa terminologi dasar. MySQL membuat, mengkonfigurasikan dan berkomunikasi dengan basis data. Sebuah basis data adalah kumpulan data yang teratur. Tabel terdiri dari beberapa *records* (biasa disebut dengan baris), dan record mengandung *field* (biasa disebut dengan kolom). [16] Untuk mengakses dan mengubah data yang tersimpan, MySQL menggunakan sintaks *Structured Query Language* (SQL) sebagai standar bahasanya. Selain MySQL, ada beberapa aplikasi DBMS

yang menggunakan sintaks SQL sebagai bahasa standar untuk memanipulasi data didalamnya, seperti SQL Server, MS Access, Sybase, Informix, dan Postgres. Beberapa sintaks sederhana dapat dilihat pada Gambar 2.4. Tabel 2. 1 adalah contoh gambaran sederhana bentuk tabel pada basis data MySQL.

Dalam pengerjaan Tugas Akhir ini, MySQL berperan sebagai basis data relasional. Basis data MySQL menyimpan data asli sistem, dimana pada percobaan penelitian ini, basis data yang ditambahkan menggunakan basis data NoSQL yaitu Apache Hbase. Kedua basis data dinsinkronisasi menggunakan *data adapter*.

```
1  INSERT INTO table_name (column1, column2) VALUES (value1, value2);
2
3  DELETE FROM table_name WHERE condition;
4
5  UPDATE table_name SET column1=value1, column2=value2 WHERE condition;
6
7  SELECT column1, column2 FROM table_name;
```

Gambar 2.4 Contoh Sintaks SQL

Tabel 2. 1. Contoh tabel pada basis data MySQL

	Field 1	*Field 2*	*Field 3*
Field Names:	**Nama**	**Umur**	**Warna Favorit**
Record 1	Bruce Callow	13	Tidak punya
Record 2	Frank Wright	37	Merah
Record 3	Seymour Hawthorne	82	Hitam, Putih

2.7 Apache HBase

Apache Hbase dibuat pada tahun 2007 di Powerset dan awalnya merupakan bagian dari Hadoop. Sejak saat itu, proyek ini menjadi proyek tingkat atas dibawah Apache Software Foundation. HBase tersedia dibawah lisensi Apache Software License, versi 2.0. [17]

Apache HBase adalah salah satu basis data NoSQL yang *open source* dimana menyajikan akses *real-time read/write* untuk sebuah basis data yang besar. HBase berjalan dalam sebauh file sistem yang bernama Hadoop. Skala penggunaan HBase adalah untuk menangani data set yang sangat besar dengan berjuta-juta baris dan kolom, dan basis data ini dengan mudah mengkombinasikan sumber data yang menggunakan struktur dan skema dengan variasi yang berbeda. HBase terintegrasi dengan Hadoop dan disamping itu bekerja dengan baik pada *data engine* YARN.

Apache HBase memberikan beberapa fitur seperti secara acak dan *real-time* mengakses data pada Hadoop. HBase dibuat untuk menampung tabel yang berukuran sangat besar, menjadikannya pilihan yang tepat untuk menyimpan *multi-structure* atau data yang jarang. Pengguna dapat melakukan *query* HBase dalam waktu tertentu, untuk melakukan "*flashback*" *query*. Karakteristik ini membuat HBase menjadi pilihan yang tepat untuk menyimpan data *semi-structured* seperti data log dan memberikan data yang sangat cepat untuk pengguna atau aplikasi yag terintegrasi dengan HBase. [18]

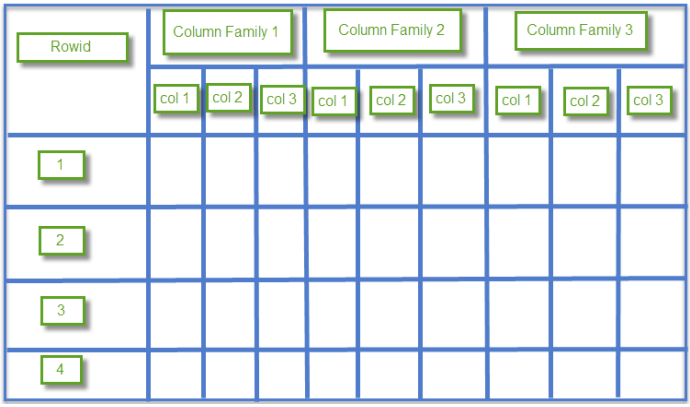
HBase adalah basis data yang berorientasi kolum dan data disimpan dalam tabel. Tabel diurutkan berdasarkakn *row id*. HBase memiliki *row id* yang merupakan kumpulan dari beberapa *column family* yang ditampilkan dalam bentuk tabel. *Column family* yang ditampilkan dalam skema adalah pasangan *key-value*. Jika dilihat secara rinci, setiap *column family* memiliki beberapa kolom. Nilai kolom ini disimpan dalam memori disk. Setiap sel dalam tabel memiliki data meta sendiri, seperti stempel waktu dan data lainnya. [19] Skema penyimpan di Hbase dapat dilihat pada Gambar 2.6. HBase memiliki perintah *query* yang berbeda dengan MySQL. Perintah *query* ini dapat dijalankan didalam HBase Shell. Gambar 2.5 menunjukan bentuk dasar perintah pada HBase. [20]

Pada pengerjaan tugas akhir ini, Apache HBase digunakan sebagai basis data NoSQL. Apache HBase disinkronisasikan dengan basis data SQL, dalam tugas akhir ini menggunakan

MySQL, melalui *data adapter*. Contoh bentuk data yang disimpan pada Apache HBase jika dilihat melalui terminal ditunjukan pada Gambar 2.7.

```
create '<table name>', '<column family>'

put '<table name>', 'row1', '<column family:column name>', '<new value>'

get '<table name>', 'row1'

delete '<table name>', '<row>', '<column name>', '<time stamp>'

scan '<table name>'
```

Gambar 2.5 Perintah Dasar pada HBase Shell



Gambar 2.6 Skema penyimpanan pada Apache Hbase

```
hbase(main):002:0* scan 'Academp'
ROW                                   COLUMN+CELL
 1                                    column=emp_details:name, timestamp=1487671596452, value=Melody
 1                                    column=emp_details:salary, timestamp=1487671596452, value=200
 2                                    column=emp_details:name, timestamp=1487671596452, value=Kinal
 2                                    column=emp_details:salary, timestamp=1487671596452, value=100
 3                                    column=emp_details:name, timestamp=1487671596452, value=Ronaldo
 3                                    column=emp_details:salary, timestamp=1487671596452, value=300
 4                                    column=emp_details:name, timestamp=1487671596452, value=Grecot
 4                                    column=emp_details:salary, timestamp=1487671596452, value=400
 5                                    column=emp_details:name, timestamp=1487671596452, value=Naomi
 5                                    column=emp_details:salary, timestamp=1487671596452, value=500
 6                                    column=emp_details:name, timestamp=1487671596452, value=Nadhifa
 6                                    column=emp_details:salary, timestamp=1487671596452, value=400
 7                                    column=emp_details:name, timestamp=1487671596452, value=Messi
 7                                    column=emp_details:salary, timestamp=1487671596452, value=300
7 row(s) in 0.2790 seconds
```

Gambar 2.7 Contoh tampilan data yang disimpan dalam Hbase jika diakses melalui terminal

2.8 Apache Phoenix

Banyak perkakas kecerdasan bisnis dan perkakas analisis data memiliki kemampuan yang kurang untuk bekerja dengan basis data HBase secara langsung. Apache Phoenix dapat membuat Anda untuk berinteraksi dengan HBase menggunakan SQL. Dengan menggunakan ODBC *drivers*, Anda dapat menghubungkan aplikasi ODBC dengan HBase melalui Apache Phoenix. [21] Beberapa perusahaan besar telah menggunakan Apache Phoenix seperti salesforce.com, Bloomberg, Sogou, Hortonworks, Alibaba.com, CertusNet, ebay.com, NGDATA, eHarmony dan PubMatic. [22]

Apache Phoenix adalah lapisan SQL yang efisien untuk Apache HBase. Apache Phoenix menambahkan SQL ke HBase, aplikasi *big data* yang terdistribusi dan *scalable* dan berjalan di atas file sistem Hadoop. Phoenix, bertujuan untuk mempermudah mengakses basis data HBase dengan mendukung sintaks SQL dan memperbolehkan *input* dan *output* dengan menggunakan JDBC API standar dibandingkan dengan menggunakan *HBase's Java client APIs*. Hal ini membuat kita dapat melakukan operasi CRUD dan DDL seperti membuat tabel, memasukan data baru, dan *query* data. SQL dan JDBC mengurangi jumlah kode yang harus dituliskan oleh pengguna, dapat melakukan optimasi performa yang transparan untuk pengguna, dan membukakan perkakas lain yang ada untuk menggunakan dan mengintegrasikannya.

Lebih mendalam, Phoenix mengambil *query* SQL, meng-*compile*-nya menjadi rangkaian pemanggilan API HBase, dan mendorongnya kedalam kluster untuk eksekusi pararel. Phoenix secara otomatis membuat *metadata* yang menyediakan tipe akses untuk data yang disimpan ke tabel HBase. Phoenix secara langsung menggunakan HBase API, bersama dengan *coprocessor* dan penyaringan secara *custom*, yang menghasilkan performa seper seribu detik untuk *query* data kecil, atau beberapa detik untuk 10 juta baris. [23]

Apache Phoenix menyediakan bahasa *query* SQL yang sudah dikenal dan jauh lebih mudah dibandingkan dengan bahasa shell HBase atau HBase Java API. Manfaat utamanya adalah menjalankan semua *query* secara pararel si semua server wilayah. Akibatnya, HBase bekerja lebih cepat dan lebih efisien. Lapisan ini kompatibel dengan driver JDBC sehingga migrasi dari sistem basis data relasional biasa tidak memerlukan perubahan kode utama. Phoenix memerlukan beberapa perubahan sintaks, beberapa sintaks yang mengalami perubahan dapat dilihat pada Tabel 2.1.

Dalam pengerjaan tugas akhir ini, Apache Phoenix digunakan pada *data adapter* sebagai data *converter* untuk merubah data dari MySQL ke HBase dengan menjalankan sintaks *query* MySQL. Selain itu, Apache Phoenix juga digunakan oleh aplikasi untuk mengambil data dari HBase melalui *data adapter*.

Tabel 2.1 Beberapa Perbedaan Sintaks Antara Apache Phoenix dengan MySQL

Phoenix	MySQL
UPSERT INTO t (col) VALUES (val);	**INSERT** INTO t (col) VALUES (val);
UPSERT INTO t (PK, col1) **SELECT** PK, 'val1' FROM t WHERE col2 = 'val2';	**UPDATE** t SET col1 = 'val1' WHERE col2 = 'val2';

2.9 Python

Python dikembangkan oleh Guido van Rossum di akhir tahun 80-an di *National Research Institute for Mathematics and Computer Science Netherlands*. Python berasal dari beberapa bahasa pemrogramman lain seperti ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix Shell dan bahasa pemrograman lainnya. Seperti Perl, Python berada dibawah GNU *General Public License* (GPL). [24]

Python adalah salah satu bahasa pemrograman tingkat tinggi yang *interpreted* dan *object-oriented* dengan semantik yang

dinamis. Dibuat di dalam struktur data, dikombinasikan dengan tipe dan ikatan yang dinamis, membuatnya menjadi atraktif untuk mengembangkan aplikasi secara sepat. Bahasa python adalah bahasa pemrograman yang sederhana, mudah untuk dipelajari dan karena itu dapat mengurangi biaya perawatan. Python mendukung penggunaan modul dan paket, yang mendorong modularitas dan penggunaan kode kembali. Python *intepreter* dan *library* standar yang tersedia dapat digunakan secara gratis dan dapat didistribusikan dengan bebas. Bahasa Python banyak digunakan karena dapat meningkatkan produktivitas seperti tidak perlu adanya langkah kompilasi, dan siklus *edit-test-debug* menjadi sangat cepat. [25]

2.10 Flask

Flask adalah sebuah kerangka kerja berbasis Python yang dipelopori oleh Armin Ronacer. Flask berada dibawah lisensi BSD. Flask menyediakan alat, beberapa library, dan teknologi yang membantu dalam membuat aplikasi web. Aplikasi web ini bisa terdiri dari beberapa halaman, blog, wiki atau e-commerce.

Flask termasuk kedalam kategori micro-framework. Micro-framework adalah kerangka kerja biasa dengan sedikit atau tanpa ketergantungan dengan libraries eksternal. Hal ini menimbulkan pro dan kontra. Kelebihannya adalah framework menjadi ringan, memiliki ketergantungan yang sedikit untuk mengupdate dan mengamati bug keamanan. [26]

Untuk dapat menggunakan Flask, yang perlu dilakukan adalah menginstallnya dengan menggunakan pip. Pip akan secara otomatis melakukan pemasangan berdasarkan versi Python yang ada di komputer. Secara sederhana, contoh penggunaan Flask dapat dilihat pada Gambar 2.8. [27]

Pada pengerjaan tugas akhir ini, Flask dipasang di sistem *data adapter* pada bagian DB Adapter. DB Adapter berfungsi untuk melakukan komunikasi dengan aplikasi dengan menerima

beberapa permintaan seperti perubahan data, pengambilan data dan memasukan data baru. Flask menghasilkan antar muka dengan bentuk data JSON yang digunakan oleh aplikas untuk mengakses data di basis data. Selain itu, pada Flask juga dipasang pada DB Converter sebagai antar muka untuk berkomunikasi dengan DB Adapter.

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route("/")
5  def hello():
6      return "Hello World!"
7
8  if __name__ == '__main__':
9      app.run()
10
```

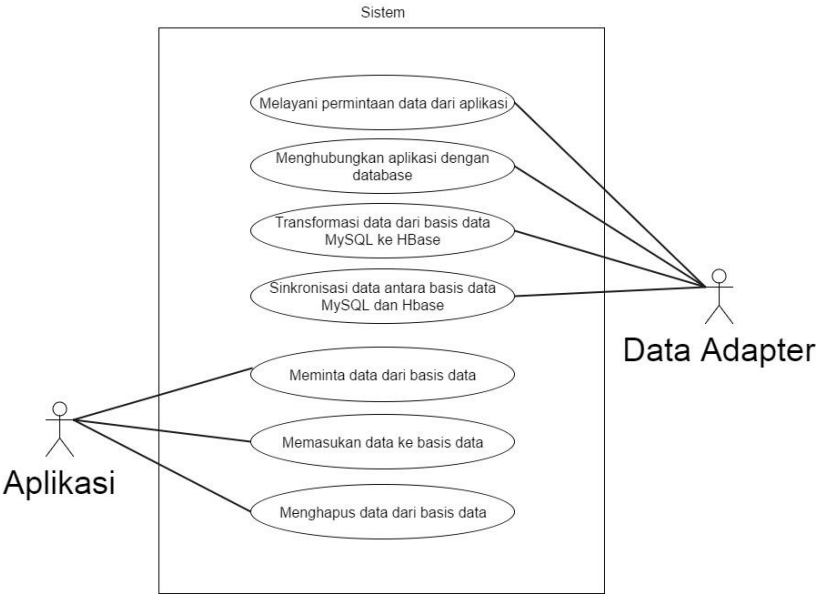
Gambar 2.8 Contoh Program Sederhana Menggunakan Flask

BAB III
DESAIN DAN PERANCANGAN

Pada bab ini akan dijelasakan mengenai analisis dan perancangan sistem. Perancangan di bagi menjadi perancangan arsitektur sistem dan perancangan proses utama sistem menggunakan *data adapter*.

3.1 Kasus Pengguna

Terdapat dua aktor dalam diagram kasus pengguna yaitu aplikasi dan *data adapter*. Pada sistem, aplikasi memiliki tiga aktifitas dan *data adapter* memiliki empat aktifitas yang di gambarkan pada Gambar 3.1.



Gambar 3.1 Diagram Kasus Pengguna

Diagram kasus pengguna pada Gambar 3.1 dijelaskan secara rinci pada tabel Tabel 3.1.

Tabel 3.1 Penjelasan Diagram Kasus Pengguna

Kode	**Nama Kasus Penggunaan**	**Aktor**	**Deskripsi**
UC01	Melayani permintaan dari aplikasi	*Data adapter*	Melayani permintaan data dari aplikasi, kemudian *data adapter* akan meneruskannya ke basis data
UC02	Menghubungkan aplikasi dengan basis data	*Data adapter*	*Data adapter* meneruskan permintaan aplikasi ke basis data. *Data adapter* menentukan dari basis data mana permintaan akan dijalankan.
UC03	Mentransformasi data dari basis data MySQL ke HBase	*Data adapter*	*Data adapter* bertugas melakukan transformasi data dari MySQL ke HBase.
UC04	Melakukan sinkronisasi data antara basis data MySQL dan HBase	*Data adapter*	Ketika aplikasi dijalankan oleh pengguna, *data adapter* akan melakukan proses sinkronisasi berdasarkan

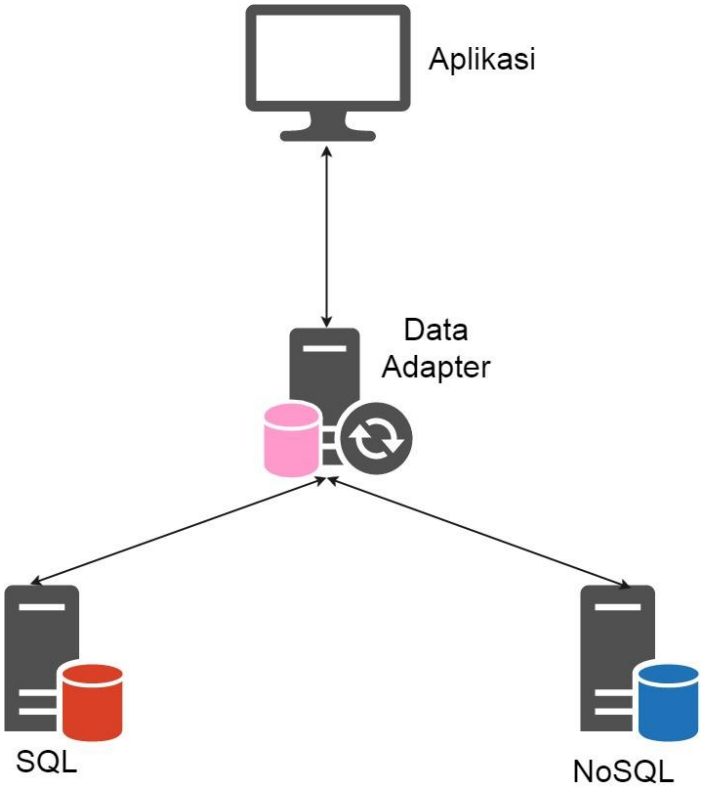
Kode	Nama Kasus Penggunaan	Aktor	Deskripsi
			perubahan yang terjadi di basis data MySQL.
UC05	Meminta data ke basis data	Aplikasi	Pengguna melalui aplikasi meminta data ke basis data.
UC06	Memasukan data ke basis data	Aplikasi	Pengguna melalui aplikasi memasukan data baru ke basis data.
UC07	Menghapus data dari basis data	Aplikasi	Pengguna melalui aplikasi menghapus data pada basis data.

3.2 Arsitektur Sistem

Sub-bab ini akan membahas mengenai analisis kebutuhan dan desain dari sistem yang akan diimplementasikan.

3.2.1 Desain Umum Sistem

Sistem yang dibangun adalah untuk mensinkronisasi dua database SQL dan No SQL. Untuk basis data tipe SQL, Database Management System (DBMS) yang digunakan adalah MySQL. Sedangkan untuk basis data tipe NoSQL, DBMS yang digunakan adalah Apache HBase. Jumlan total server yang digunakan pada penelitian ini adalah berjumlah tiga server, satu server untuk MySQL, satu server untuk Apache HBase dan satu server untuk *Data adapter*. Secara umum, visualisasi arsitektur yang digunakan sistem *data adapter* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Desain Sistem Data Adapter Secara Umum

Data adapter adalah penghubung antara aplikasi dengan basis data. *Data adapter* berperan untuk menerima permintaan dari aplikasi dan melakukan transformasi data dari MySQL ke Apache Hbase. Komponen utama dari sistem *data adapter* ini terdiri dari empat, yaitu basis data relasional, basis data NoSQL, DB Adapter dan DB Converter. DB Adapter berfungsi untuk menerima permintaan dari aplikasi seperti insert data, pembaruan data, menghapus data dan mengambil data. Untuk dapat melakukan perubahan dan pengaksesan data, *data adapter* menyediakan *interface* yang dapat diakses oleh aplikasi yang terhubung.

Interface yang disediakan oleh *data adapter* adalah dalam bentuk *Application Programming Interface* (API). *Interface*, atau antar muka, ini menghasilkan bentuk tipe data JSON yang dibuat dengan menggunakan Flask. Flask adalah kerangka kerja mikro yang berjalan menggunakan bahasa pemrograman Python. Antar muka ini dapat diakses melalui *port* 5000 oleh aplikasi. Dengan antar muka ini, aplikasi dapat melakukan perubahan dan pengaksesan data pada basis data.

Sementara itu, DB Converter bertanggung jawab dalam transformasi data dan pelaporan hasil transformasi yang dicatat di basis data. Transformasi data dilakukan dari basis data RDB ke basis data NoSQL. Untuk menerima permintaan dari DB Adapter, DB Converter berjalan pada port 5001. Proses transformasi ini dilakukan dengan bantuan Apache Phoenix. Pada DB Converter ini juga dipasang sebuah basis data menggunakan MySQL untuk menyimpan log sinkronisasi. Komponen utama dari *data adapter* ini dibangun dengan menggunakan bahasa pemrograman Python.

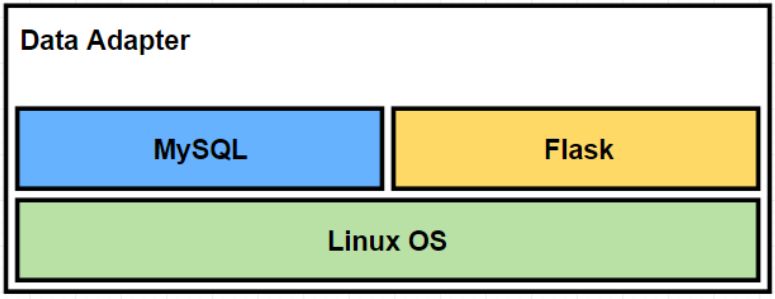
3.2.2 Desain *Data adapter*

Data adapter memiliki peran paling penting di dalam sistem. Setiap permintaan yang dilakukan oleh aplikasi akan diolah terlebih dahulu di *data adapter*, yang kemudian akan diteruskan ke basis data. Ada beberapa contoh antarmuka yang digunakan oleh aplikasi, dalam pengerjaan tugas akhir ini, antarmuka yang digunakan adalah API dalam bentuk JSON. Diagram arsitektur *data adapter* dapat dilihat pada

Data adapter terdiri dari dua komponen utama, yaitu DB Adapter dan DB Converter. DB Adapter berfungsi untuk menerima semua permintaan dari aplikasi seperti pengajuan *query*, pengambilan data, pembaruan data, dan penghapusan data. Untuk dapat melakukan hal ini, *data adapter* menyediakan antar muka dalam bentuk API. API menghasilkan keluaran data denga tipe data JSON. Semua komunikasi data yang dilakukan oleh aplikasi adalah

melalui antar muka ini. Flask adalah kerangka kerja mikro yang digunakan untuk membuat fitur antar muka.

Sementara, DB Converter berfungsi untuk melakukan transformasi data dari basis data SQL ke basis data NoSQL. Dalam hal ini, transformasi yang dilakukan adalah dari MySQL ke Apache HBase. Secara *default*, permintaan perubahan data dari aplikasi akan diarahkan ke MySQL, sedangkan untuk permintaan pengaksesan data, permintaan dari aplikasi akan di arahkan ke HBase. Proses transformasi data ini dilakukan dengan bantuan Apache Phoenix. Apache Phoenix memungkinkan untuk menerjemahkan *query* SQL, mengkompilasinya menjadi beberapa rangkaian perintah HBase, kemudian mengeksekusinya ke HBase. Python menjadi bahasa pemrograman yang digunakan untuk mengimplementasikan semua proses yang ada pada *data adapter*. Gambar 3.3 menampilkan diagram arsitektur *data adapter*.

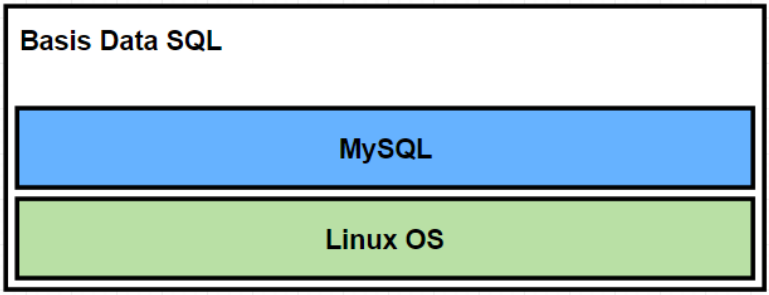


Gambar 3.3 Diagram Arsitektur Data Adapter

3.2.3 Desain Basis Data SQL

Basis data SQL digunakan untuk menyimpan semua data aplikasi yang bersifat relasional. Dalam perancangan dan pengembangan suatu aplikasi dituntut kemampuan aplikasi untuk dapat diakses oleh banyak pengguna. Banyak pengembang yang memisahkan server aplikasi dengan server basis data atas dasar kebutuhan ini. Pada pengerjaan tugas akhir ini, satu server yang

terpisah dari server aplikasi digunakan sebagai basis data SQL. Aplikasi DBMS yang digunakan adalah MySQL. Agar dapat digunakan aplikasi, perlu dibuatkan akun pengguna yang memiliki kewenangan untuk melakukan perubahan data pada server basis data. Semua permintaan dari aplikasi yang berkaitan dengan perubahan data, seperti *insert, update* dan *delete,* dan pengambilan data dapat dieksekusi di MySQL. Diagram arsiterkur basis data SQL tertera pada Gambar 3.4.



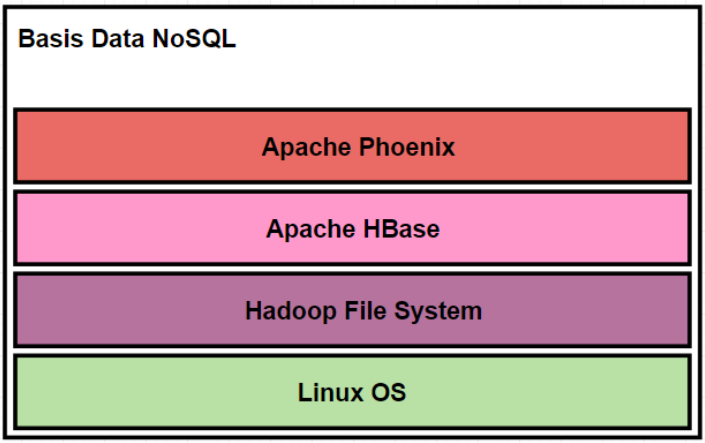
Gambar 3.4 Diagram Arsitektur Basis Data SQL

3.2.4 Desain Basis Data NoSQL

Basis Data NoSQL digunakan untuk menyimpan data dengan format yang berbeda dengan basis data NoSQL. Basis data ini adalah basis data tambahan karena jumlah data yang semakin bertambah. Aplikasi yang digunakan sebagai basis data NoSQL adalah Apache HBase. Untuk dapat menggunakan Apache HBase, terlebih dahulu harus memasang Apache Hadoop, sistem berkas milik Apache yang khusus menangani *big data*, pada server karena Apache Hbase berjalan diatas Apache Hadoop.

Pemasangan basis data NoSQL dilakukan pada satu server lain yang terpisah dari aplikasi dan basis data SQL. HBase akan menerima hasil transformasi data dari MySQL yang dikontrol melalui *data adapter*. Proses transformasi ini membutuhkan

bantuan dari Apache Phoenix sebagai penerjemah *query* SQL. Desain arsitektur basis data NoSQL dapat dilihat pada Gambar 3.5.

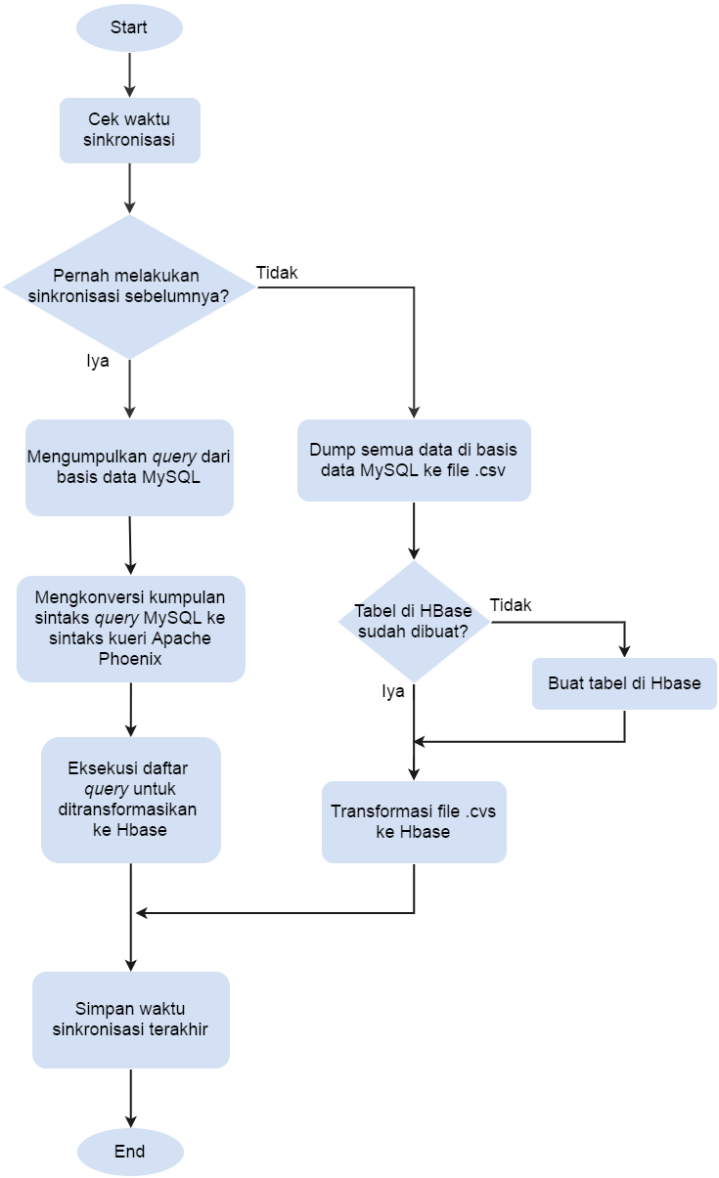


Gambar 3.5 Diagram Arsitektur Basis Data NoSQL

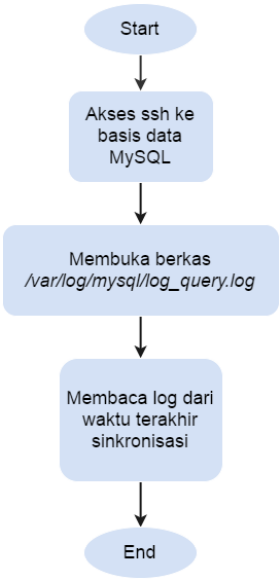
3.3 Proses Transformasi dan Sinkronisasi

Proses transformasi dilakukan secara manual ketika user mengeksekusi aplikasi sinkronisasi. Hal pertama yang dilakukan ketika melakukan sinkronisasi adalah mengecek di basis data log sinkronisasi, apakah pernah melakukan sinkronisasi sebelumnya atau tidak. Jika belum maka proses yang dilakukan adalah proses inisialisasi. Diagram alir proses sinkronisasi secara umum dapat dilihat pada Gambar 3.6.

Proses inisialisasi adalah proses menyalin semua data di basis data SQL ke basis data NoSQL. Proses ini diawali dengan men-*dump* atau mengekspor semua data pada tabel di MySQL menjadi berkas berformat .csv. Kemudian sistem akan membuat tabel baru yang sama dengan tabel yang ada pada basis data di MySQL. Setelah selesai, proses selanjutnya adalah melakukan transformasi data ke HBase. Proses transformasi data ini ditangani dengan menggunakan bantuan Apache Phoenix.



Gambar 3.6 Diagram Alir Proses Sinkronisasi

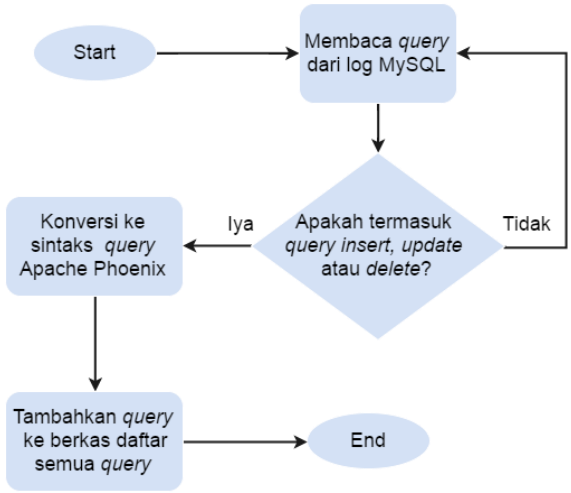


Gambar 3.7. Diagram Alir Proses Mengambil Query dari Basis Data MySQL

Jika pernah melakukan sinkronisasi sebelumnya, maka inisialisasi tidak akan dilakukan. Proses yang akan dilakukan adalah sistem akan mengambil semua *query* dari log pada basis data MySQL. Gambar 3.7 menunjukan diagram alir proses pengambilan *query* pada server basis data SQL. Sistem me-*remote* basis data MySQL, kemudian membaca daftar log pada berkas */var/log/mysql/log_query.log*. Log yang dibaca adalah berdasarkan waktu terakhir sinkronisasi. Dari log yang dibaca, sistem mengecek semua *query* yang melakukan perubahan pada basis data yaitu perintah *insert*, *update* dan *delete*. *Query* tersebut diambil dan dikonversi menjadi sintaks *query* Apache Phoenix. Semua sintaks baru kemudian disimpan didalam satu berkas dengan format .sql. Gambar 3.8 menampilkan diagram alir konveri sintaks *query* MySQL ke *query* Apache Phoenix. Dengan bantuan Apache Phoenix, semua *query* pada berkas .sql akan ditransformasi ke

HBase, sehingga semua perubahan data yang terjadi pada basis data SQL akan terjadi juga di basis data NoSQL.

Jika proses sinkronisasi berhasil, sistem akan mencatat log dan status sinkronisasi tersebut ke basis data log sinkronisasi. Basis data yang digunakan untuk mencatat log ini menggunakan MySQL dan terpisah dari server basis data SQL, server basis data NoSQL maupun server aplikasi.

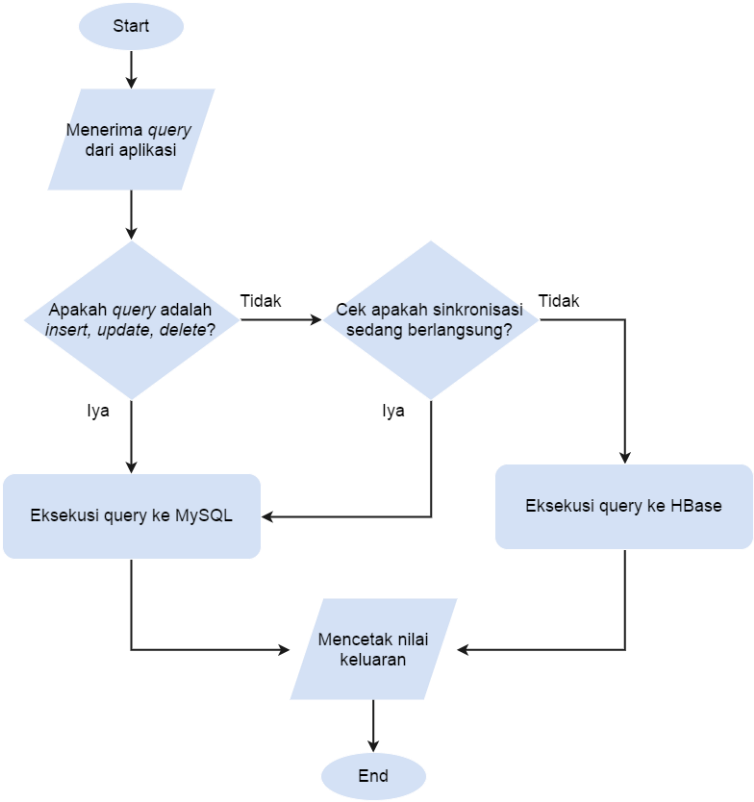


Gambar 3.8 Diagram Alir Konversi Sintaks Query MySQL ke Query Apache Phoenix

3.4 Desain Antarmuka

Aplikasi mangakses basis data dengan menjalankan *query* SQL melalui antarmuka atau rute-rute yang disediakan oleh *data adapter*. Dengan pendekatan *query direct access*, *data adapter* akan langsung meneruskan sintaks *query*, pada antar muka yang diakses, ke basis data yang dimaksud dan mengembalikan nilai dari keluaran basis data. Dari rute yang diakses aplikasi, sistem pertama kali akan melakukan pengecekan apakah *query* adalah merubah data pada basis data atau mengambil data. Jika *query* adalah *insert*,

update atau *delete*, maka perintah *query* akan dieksekusi ke basis data MySQL. Jika tidak, atau dalam hal ini *query* yang dijalankan adalah *query select*, maka sistem akan memastikan apakah proses sinkronisasi sedang berlangsung atau tidak. Secara default, *query select* akan diarahkan ke basis data HBase, namun jika proses sinkronisasi sedang berlangsung maka *query* akan diarahkan ke basis data MySQL. Sistem kemudian mencetak hasil keluaran yang didapat dari basis data dalam bentuk JSON. Gambar 3.9 menunjukan diagram alir antarmuka sistem *data adapter*.



Gambar 3.9 Diagram Alir Antarmuka Data adapter

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi sistem *data adapter* berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Pembahasan dilakukan secara rinci untuk setiap komponen yang ada yaitu basis data SQL, basis data NoSQL dan *data adapter*.

4.1 Lingkungan Implementasi

Lingkungan implementasi dan pengembangan dilakukan menggunakan komputer dengan spesifikasi Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz dengan memory 5.8 GB di Laboratorium Arsitektur dan Jaringan Komputer, Jurusan Teknik Informatika ITS. Perangkat lunak yang digunakan dalam pengembangan dipaparkan pada Tabel 4.1.

Tabel 4.1 Perangkat Lunak yang Digunakan

No	Perangkat Lunak	Versi	Keterangan
1	Linux Ubuntu	14.04	Sistem Operasi
2	Sublime	2	*Text Editor*
3	Python	2.7.6	Bahasa Pemrograman
4	Flask	0.12.1	Pustaka Python
5	Virtual Environment	15.1.0	Lingkungan Kerja Python
6	Apache Hadoop	2.7.3	Sistem Berkas
7	Apache HBase	1.2.4	Basis Data NoSQL
8	Apache Phoenix	4.10.0	Translator
9	MySQL	14.14	Basis Data SQL
10	Git	1.9.1	Pengelola Versi Program
11	Postman	4.10.7	REST *client*
12	Heidi SQL	9.3.0.4984	Antarmuka MySQL
13	Microsoft Word	2013	Pengolah kata

4.2 Rincian Implementasi Server Basis Data

Basis data digunakan untuk menyimpan semua data yang digunakan aplikasi. Pada pengerjaan Tugas Akhir ini, basis data yang akan disinkronisasikan adalah basis data SQL dan NoSQL. Sistem manajemen basis data SQL yang digunakan adalah MySQL Server dan untuk NoSQL menggunakan Apache HBase. Kedua basis data ini dipasang pada server yang terpisah. Pada sub bab ini akan dijelaskan secara rinci mengenai implementasi masing-masing basis data.

4.2.1 Instalasi Server Basis Data SQL

Pada pengerjaan tugas akhir ini hanya menggunakan satu server untuk Basis data SQL. Perlu dilakukan konfigurasi lebih lanjut setelah melakukan pemasangan MySQL pada server agar *data adapter* dapat terhubung dengan server.

Alamat IP server diatur sesuai dengan subnet di Laboratorium Arsitektur dan Jaringan agar server *data adapter* dan server basis data NoSQL dapat saling terhubung dan bekomunikasi. Server yang digunakan untuk basis data SQL menggunakan alamat IP 10.151.36.22. Konfigurasi basis data agar berjalan dengan menggunakan IP yang tertera dengan mengganti variabel *bind-address*.

Default-nya MySQL hanya akan mencatat log *error*, log yang berfungsi untuk mencatat kegagalan yang terjadi saat MySQL beroperasi. Pada pengerjaan Tugas Akhir ini, MySQL perlu mencatat log *query* apa saja yang dieksekusi karena *data adapter* akan membaca log ini untuk melakukan sinkronisasi dengan basis data HBase. Oleh karena itu perlu ditambahkan baris konfigurasi agar MySQL mencatat log *query* yang dieksekusi ke MySQL. Untuk memperbolehkannya, sunting berkas /etc/mysql/my.cnf dan tambahkan baris *log = /var/log/mysql/log_query.log*. Kode Sumber 4.1 adalah rincian konfigurasi tambahan pada berkas my.cnf.

Secara default, server MySQL hanya bisa diakses oleh *localhost* saja. Oleh karena itu perlu ditambahkan konfigurasi agar *data adapter* dapat mengakses server MySQL. Dalam pengerjaan tugas akhir ini, karena tingkat keamanan saat ini tidak terlalu dibutuhkan, maka perintah untuk menambahkan akses ke server diperbolehkan untuk semua alamat IP. Perintah ini dijalankan seperti dengan mengeksekusi *query* di MySQL. Terlebih dahulu harus masuk sebagai *user root*. Langkah-langkah menambahkan akses *user* lebih detailnya adalah sebagai berikut:

1. Masuk ke MySQL dengan menggunakan username dan password default. Jalankan *perintah mysql –u root –p*, kemudian tekan enter dua kali.
2. Buat pengguna baru dari semua alamat dengan nama ‘wicak’ dan password ‘w’ dengan menjalankan perintah *CREATE USER ‘wicak’@’%’ IDENTIFIED BY ‘w’*;
3. Berikan akses kepada pengguna baru yang kita buat agar pengguna yang dimaksud dapat mengakses basis data dari alamat IP lain. Jalankan perintah *GRANT ALL PRIVILEGES ON *.* TO ‘wicak’@’%’ IDENTIFIED BY ‘w’*;
4. Muat ulang konfigurasi dengan menjalankan perintah *FLUSH PRIVILEGES*;

```
1  bind-address  =  10.151.36.129
2  log = /var/log/mysql/log_query.log
```

Kode Sumber 4.1 Konfigurasi MySQL pada Berkas my.cnf

4.2.2 Instalasi Server Basis Data NoSQL

Basis data yang dipasang sebagai basis data NoSQL adalah Apache HBase. HBase dipasang pada satu server yang terpisah dengan server basis data MySQL. Untuk memasang HBase, terlebih dahulu harus melakukan instalasi Hadoop karena HBase berjalan diatas sistem berkas Hadoop.

4.2.2.1 Instalasi Hadoop

Pada bagian ini akan dipaparkan tahapan pemasangan sistem berkas Hadoop. Langkah-langkahnya adalah sebagai berikut :

1. Pastikan pada server yang digunakan telah terpasang java. Kemudian install OpenJDK dengan mengetikan perintah *apt-get install default-jdk*.
2. Kemudian buat pengguna untuk Hadoop. Namun terlebih dahulu buat group dengan dengan nama hadoop. Ketikan *sudo addgroup hadoop*.
3. Tambahkan pengguna baru pada group hadoop dengan menjalankan perintah *sudo adduser --ingroup hadoop hduser*.
4. Tambahkan pengguna hduser ke akses sudo dengan perintah *sudo adduser hduser sudo*.
5. Hadoop memerlukan SSH untuk memanajemen node-nya secara remote dan lokal. Lakukan instalasi ssh dengan menjalankan perintah *sudo apt-get install ssh*.
6. Hadoop menggunakan SSH (untuk mengakses node-nya) biasanya mengharuskan penggunanya untuk memasukan kata sandi. Namun, persyaratan ini bisa ditiadakan dengan membuat dan mengatur sertifikat SSH dengan menjalankan perintah *ssh-keygen –t rsa –P ".*
7. Selanjutnya adalah menambahkan kunci baru ke daftar kunci yang berwenang sehingga Hadoop dapat menggunakan ssh tanpa meminta password. Ketikan perintah *cat $HOME/ .ssh/id_rsa.pub >> $HOME/ .ssh/authorized_keys*.
8. Unduh berkas instalasi Hadoop dengan perintah *wget http://mirrors.sonic.net/apache/hadoop/common/stable/hadoop-2.7.3.tar.gz.*
9. Untuk mengekstrak berkas Hadoop jalankan tar xvzf hadoop-2.7.3.tar.gz.
10. Kemudian pindahkan semua berkas instalasi ke direktori */usr/local/hadoop*.

11. Selanjutnya adalah melakukan konfigurasi pada Hadoop. Yang pertama adalah melakukan konfigurasi pada berkas bashrc dengan menjalankan perintah *vim ~/.bashrc*. Konfigurasi yang ditambahkan pada berkas .bashrc dapat dilihat pada Kode Sumber 4.2.
12. Tambahkan variabel JAVA_HOME kedalam berkas hadoop-env.sh agar ketika Hadoop dijalankan maka nilai JAVA_HOME tersedia. Ketikkan perintah *vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh* untuk merubahnya. Isi konfigurasi variable ini dapat dilihat pada Kode Sumber 4.3.
13. Berkas */usr/local/hadoop/etc/hadoop/core-site.xm*l berisi konfigurasi yang digunakan Hadoop ketika mengawali proses. Buka dan sunting berkas tersebut dengan perintah *vim /usr/local/hadoop/etc/hadoop/core-site.xml*. Isi konfigurasi dapat dilihat pada Kode Sumber 4.4.
14. Salin dan *rename /usr/local/hadoop/etc/hadoop/mapred-site.xml.template* menjadi *mapred-site.xml*. Kemudian tambahkan konfigurasi seperti pada Kode Sumber 4.5.
15. *Berkas /usr/local/hadoop/etc/hadoop/hdfs-site.xml* diatur untuk setiap *host* kluster yang digunakan. Ubah *berkas hdfs-site.xml* menjadi seperti pada Kode Sumber 4.6. Sebelum menyuntingnya, buat dua direktori baru yang akan terdiri *namenode* dan *datanode* Hadoop. Buat direktori dengan menjalankan perintah *mkdir –p /usr/local/hadoop-store/hdfs/namenode* dan *mkdir –p /usr/local/hadoop-store/hdfs/datanode.*
16. Sistem berkas Hadoop perlu di format untuk dapat digunakan. Jalankan perintah *hadoop namenode –format*.
17. Hadoop telah siap digunakan. Masuk ke direktori */usr/local/hadoop/sbin*. Kemudian gunakan *start-all.sh* untuk memulai Hadoop. Tampilan Web UI Hadoop dapat dilihat dengan mengakses *http://localhost:50070* pada *browser*.

```
# HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
```

Kode Sumber 4.2 Konfigurasi pada Berkas .bashrc

```
export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Kode Sumber 4.3 Variable JAVA_HOME pada hadoop-env.sh

```
<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>

 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose scheme and authority determine the  FileSystem implementation.  The uri's scheme  determines the config property (fs.SCHEME.impl)  naming the FileSystem implementation class.  The  uri's authority is used to determine the host, port, etc. for a filesystem.
  </description>
```

```
 </property>
</configuration>

```

Kode Sumber 4.4 Konfigurasi pada Berkas core-site.xml

```
<configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce
job tracker runs at.  If "local", then jobs are run
in-process as a single map  and reduce task.
  </description>
 </property>
 <property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
 </property>
</configuration>

```

Kode Sumber 4.5 Konfigurasi pada Berkas mapred-site.xml

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication. The actual
number of replications can be specified when the
file is created. The default is used if replication
is not specified in create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop_store/hdfs/namenode</v
alue>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_store/hdfs/datanode</v
alue>
```

```
 </property>
</configuration>

```

Kode Sumber 4.6 Konfigurasi pada Berkas hdfs-site.xml

4.2.2.2 Instalasi Apache HBase

Setelah melakukan instalasi sistem berkas Hadoop, dapat dilakukan instalasi Apache HBase. Apache HBase yang digunakna adalah versi 1.2.4. Pada sub bab ini akan dijelaskan mengenai cara pemasangan Apache HBase.

1. Unduh paket Apache Hbase dengan menjalankan perintah *wget https://archive.apache.org/dist/hbase/1.2.4/hbase-1.2.4-bin.tar.gz*.
2. Unzip berkas dengan menjalankan *tar –xvf hbase-1.2.4-bin.tar.gz*.
3. Buat direktori hbase dengan menjalankan perintah *mkdir /usr/lib/hbase*. Pindahkan folder hasi ekstrasi kedalam folder ini.
4. Atur path java yang ada pada server pada berkas hbase-env.sh. Ubah dengan menjalankan perintah *vim /usr/lib/hbase/hbase-1.2.4/conf/hbase-env.sh* dan tambahkan konfigurasi seperti pada
5. Tambahkan juga path HBASE_HOME pada berkas .bashrc. Tambahkan konfigurasi seperti pada dengan menjalan perintah *vim ~/.bashrc*.
6. Selanjutnya adalah mengubah berkas konfigurasi pada HBase pada berkas hbase-site.xml untuk mengatur direktori tempat HBase akan menyimpan data. Menunjukan isi konfigurasi *hbase-site.xml*
7. Untuk memulai HBase masuk ke direktori */usr/lib/hbase/hbase-1.2.4/bin* kemudian jalankan start-hbase.sh.
8. Untuk mengecek apakah HBase telah berjalan, jalankan perintah *hbase shell.*

4.3 Rincian Implementasi Sistem *Data adapter*

Komponen utama pada sistem *data adapter* yaitu DB Adapter dan DB Converter. Sistem *data adapter* dibangun dengan menggunakan bahasa pemrograman Python dan kerengka kerja pemrograman Flask. Selain itu, digunakan beberapa modul tambahan yang dipasang secara terpisah dengan menggunakan pip pada terminal. Pip adalah sebuah perkakas untuk melakukan pemasangan modul-modul paket Python. Pada sub bab ini akan dijelaskan secara rinci mengenai implementasi *data adapter*.

4.3.1 Instalasi Paket

Seperti yang dijelaskan sebelumnya, pemasangan modul-modul Python ini dilakukan dengan menggunakan pip. Cara kerja pip ini adalah mengunduh modul untuk sistem yang dibutuhkan, kemudian dipasang pada server. Semua modul yang dibutuhkan sistem disimpan dalam berkas requirement_interface.txt. Untuk menjalankan pemasangan modul di dalam file tersebut dapat dijalankan dengan menggunakan perintah *pip install –r requirement_interface.txt*. Semua daftar modul di dalam berkas tersebut akan dipasang di server. Modul-modul tambahan yang dipasang yaitu :

1. Flask 0.12.1
2. Appdirs 1.4.3
3. Click 6.7
4. Itsdangerous 0.24
5. Jinja2 2.9.6
6. MarkupSafe 1.0
7. MySQL-python 1.2.5
8. Packaging 16.8
9. Protobuf 3.2.0
10. Pyparsing 2.2.0
11. Six 1.10.0
12. Werkzeug 0.12.1

4.3.2 Implementasi DB Adapter

DB Adapter adalah komponen pada sistem *data adapter* yang bertugas sebagai penghubung antara aplikasi dengan basis data. DB Adapter menerima *query*, mengeksekusi *query* dan mendapatkan hasilnya dari basis data melalui DB Converter. DB Adapter menyediakan antar muka yang dapat diakses oleh aplikasi dimana antarmuka ini dapat menerima permintaan yang dilakukan oleh aplikasi seperti pengajuan *query*, pengambilan data, pembaruan data, dan penghapusan data. Semua permintaan ini diterima oleh DB Adapter dengan menggunakan API yang dibuat dengan menggunakan Flask. Flask yang digunakan adalah versi 0.12.1. Antar muka ini menghasilkan keluaran data dalam bentuk JSON.

Untuk melakukan permintaan atau proses data, DB Adapter menyediakan antar muka yang diakses oleh aplikasi dengan melakukan permintaan ke rute yang disediakan. Dalam pengerjaan Tugas Akhir ini, beberapa rute yang disediakan beserta dengan rincian penjelasannya dapat dilihat pada Tabel 4.2. Rute ini diimplementasikan dalam sebuah berkas Python. Tabel 4.3 berisi daftar sintaks SQL dari masing-masing rute yang dibuat.

Tabel 4.2 Implementasi Rute Pada DB Adapter

No	Rute	Masukan	Luaran	Proses
1	GET /sinkron	-	status, message	Melakukan sinkronisasi basis data MySQL ke HBase
2	POST /insert_routes	data route	status, message	Memasukan data baru ke tabel routes
3	POST /delete_routes_by_id	id_route	status, message	Menghapus satu baris data data pada tabel routes

No	Rute	Masukan	Luaran	Proses
4	POST /update_routes_by_id	id_route	status, message	Melakukan perbaruan data pada tabel routes
5	GET /select_all_routes	-	host, database, rows, flight_routes	Mengambil semua data pada tabel routes
6	GET /select_route_by_id/<id>	id_route	host, database, route	Mengambil satu baris data pada tabel route berdasarkan id
7	POST /insert_airline	data_airline	status, message	Memasukan satu barus data baru ke tabel airline
8	POST /delete_airline_by_id	id_airline	status, message	Menghapus data pada tabel airline berdasarkan id airline
9	POST /update_airline_by_id	data_airline	status, message	Memperbarui data pada tabel airline
10	GET /select_all_airline	-	host, database, rows, airlines	Mengambil semua data pada tabel airline
11	GET /select_airline_by_id/<id>	Id_airline	host, database, airline	Mengambil satu baris data pada tabel airline
12	POST /insert_airport	Data_airport	status, message	Memasukan satu baris data baru ke tabel airport

No	Rute	Masukan	Luaran	Proses
13	POST /delete_airport_by_id	Id_airport	status, message	Menghapus satu barus data pada tabel airport
14	POST /update_airport_by_id	data_airport	status, message	Memperbarui data pada tabel airport
15	GET /select_all_airport	-	host, database, rows, airport	Mengambil semua data pada tabel airport
16	GET /select_airport_by_id/<id>	id_airport	host, database, airport	Mengambil satu baris data pada tabel airport berdasarkan id
17	GET /select_routes_country/<id>	id_route	host, database, route	Mengambil detail dari tabel route dan tabel airline
18	GET /select_all_number_routes_by_airline	-	host, rows, database, routes	Mengambil jumlah rute dari setiap airline
19	GET /select_all_number_routes_more_than/<number>	number	host, database, routes, rows	Mengambil data airline yang memiliki rute lebih dari nomor yang ditentukan
20	GET /select_all_airline_in_routes	-	host, database, airlines, rows	Mengambil semua airline yang ada pada tabel rute
21	GET /select_all_airline_source	-	host, database,	Mengambil semua detail airline dan

No	Rute	Masukan	Luaran	Proses
			airlines, rows	sumber penerbangan

Tabel 4.3 Daftar Sintaks SQL pada Setiap Rute

No	Pengendali	Sintaks SQL
1	/insert_routes	`INSERT INTO routes (id_route, airline, id_airline, src_airport, id_src_airport, dst_airport, id_dst_airport, codeshare, stop_val, equipment, log_date) VALUES (v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11);`
2	/delete_routes_by_id	`DELETE routes WHERE id_route = v0;`
3	/update_routes_by_id	`UPDATE routes SET airline=v1, id_airline=v2 src_airport=v3, id_src_airport=v4, dst_airport=v5, id_dst_airport=v6, codeshare=v7, stop_val=v8, equipment=v9, log_date=v10 WHERE id_route=v11;`
4	/select_all_routes	`SELECT * FROM routes;`
5	/select_route_by_id/<id>	`SELECT * FROM routes WHERE id_route = v1;`
6	/insert_airline	`INSERT INTO airline (id_airline, name, alias, iata, icao, callsign, country, active_stat) VALUES (v1, v2, v3, v4, v5, v6, v7, v8);`
7	/delete_airline_by_id	`DELETE airline WHERE id_airline = v1;`
8	/update_airline_by_id	`UPDATE airline SET name=v1, alias=v2, iata=v3, icao=v4, callsign=v5, country=v6, active_stat=v7 WHERE id_airline=v8;`

No	Pengendali	Sintaks SQL
9	/select_all_airline	`SELECT * FROM airline;`
10	/select_airline_by_id/<id>	`SELECT * FROM airline WHERE id_airline;`
11	/insert_airport	`INSERT INTO airport (airport_id, name_airport, city, country, iata, icao, latitude, longitude, altitude, timezone, dst, tz_db, type_airport, source) VALUES (v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14);`
12	/delete_airport_by_id	`DELETE airport WHERE airport_id = v0;`
13	/update_airport_by_id	`UPDATE airport SET name_airport=v1, city=v2, country=v3, iata=v4, icao=v5, latitude=v6, longitude=v7, altitude=v8, timezone=v9, dst=v10, tz_db=v11, type_airport=v12, source=v13 WHERE airport_id=v14;`
14	/select_all_airport	`SELECT * FROM airport;`
15	/select_airport_by_id/<id>	`SELECT * FROM airport WHERE airport_id=v1;`
16	/select_routes_country/<id_route>	`SELECT a.*,b.country FROM routes2 a JOIN airline2 b on a.id_airline = b.id_airline where a.id_route = v1;`
17	/select_all_number_routes_by_airline	`SELECT a.id_airline,b.name,count(id_route) FROM routes2 a JOIN airline2 b ON a.id_airline = b.id_airline GROUP BY a.id_airline,b.name;`
18	/select_all_number_routes_more_than/<number>	`SELECT id_airline, count(id_route) FROM routes2 GROUP BY id_airline HAVING COUNT(*) > v1;`

No	Pengendali	Sintaks SQL
19	/select_all_airline_in_routes	`SELECT DISTINCT id_airline from routes2;`
20	/select_all_airline_source	`SELECT a.id_route, a.id_airline, b.name, a.src_airport, c.name_airport AS source FROM routes2 a JOIN airline2 b ON a.id_airline=b.id_airline JOIN airport2 c ON a.id_src_airport=c.airport_id;`

Untuk melakukan pengolahan data, terdapat *controller* di tiap-tiap di rute yang berfungsi untuk memilih pada basis data mana *query* akan dieksekusi. Jika *query* bersifat merubah data pada basis data, maka eksekusi *query* diarahkan ke basis data SQL, dalam penelitian ini yaitu MySQL. Pseudocode kontroler untuk proses perubahan data ditunjukan pada Pseudocode 4.1. Sedangkan untuk *query* yang memiliki sifat mengambil data, maka eksekusi *query* diarahkan ke basis data NoSQL, dalam hal ini adalah HBase. Pseudocode 4.2 menunjukan proses eksekusi *query* untuk pengambilan data.

```
1  post data from application
2  connect to MySQL DB
3  excecute query
4  return result
```

Pseudocode 4.1 Eksekusi Query untuk Proses Perubahan Data

Penjelasan Pseudocode 4.1 adalah sebagai berikut :

1. Mengambil data *post* dari aplikasi
2. Jika aksi yang dilakukan aplikasi adalah perubahan pada basis data, seperti menambahkan, menghapus dan

menyunting, maka DB Adapter terhubung basis data MySQL.
3. Proses menjalankan sintaks *query*
4. Mengembalikan hasil eksekusi *query*

```
1  get status synchroniztion
2  if syncrhonization process still working
3      connect to MySQL DB
4  else
5      connect to HBase
6  result = excecute query
7  return result
```

Pseudocode 4.2 Eksekusi Query untuk Proses Pengambilan Data

Secara *default*, proses pengambilan data akan diarahkan ke HBase karena untuk proses pembacaan data pada HBase menghabiskan waktu yang lebih singkat dibandingkan dengan basis data relasional. Berikut adalah penjelasan Pseudocode 4.2:

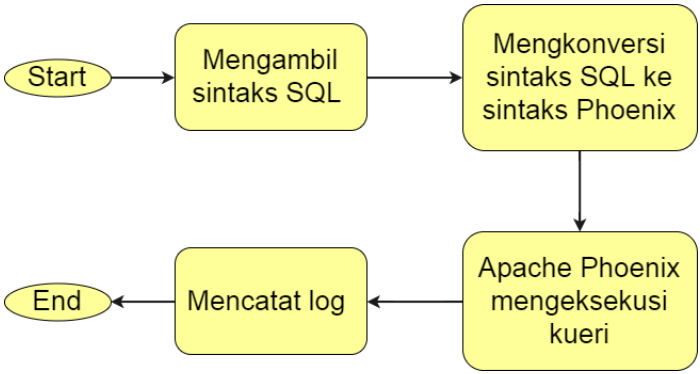
1. Pengeksekusian dimulai dari mengecek status apakah sedang berlangsung proses sinkronisasi atau tidak
2. Cek jika proses sinkronisasi sedang berlangsung, maka proses pengambilan data diarahkan ke basis data MySQL. Jika tidak maka diarahkan ke HBase.
3. Proses menjalankan sintaks *query*.
4. Mengembalikan hasil *query*.

4.3.3 Implementasi DB Converter

DB Converter adalah komponen lain *data adapter* yang berfungsi untuk mencatat log dan status sinkronisasi serta melakukan transformasi data dari basis data SQL ke basis data NoSQL. Komponen yang terpasang pada DB Converter ini diantaranya Flask dan Apache Phoenix.

Flask digunakan untuk membuat antar muka yang berfungsi untuk menerima permintaan proses sinkronisasi dari DB Adapter. Apache Phoenix berfungsi sebagai lapisan layer yang bekerja diatas HBase yang dapat menerjemahkan SQL dan mengeksekusinye ke HBase. Terdapat basis data yang menggunakan MySQL untuk mencatat log dan status setiap kali menjalankan proses sinkronisasi.

Ketika DB Adapter menerima permintaan sinkronisasi, DB Adapter kemudian meneruskan permintaan tersebut dengan meminta DB Converter untuk melakukan sinkronisasi. DB Converter mengambil sintaks SQL dari log *query* pada server MySQL. *Query* ini kemudian dikonversi menjadi sintaks Apache Phoenix. Kemudian Apache Phoenix akan menjalankan *query* tersebut ke Apache HBase. Data di HBase akan berubah sesuai dengan *query* yang dijalankan. Setelah proses transformasi selesai, sistem mencatat log pada basis data log sinkronisasi. Gambar 4.1 menampilkan diagram alir proses sinkronisasi pada DB Converter.



Gambar 4.1 Diagram Alir DB Converter

Proses sinkronisasi basis data SQL ke NoSQL dari MySQL ke Apache HBase ditunjukan pada Pseudocode 4.3. Implementasi dari Pseudocode 4.3 ditunjukan pada Kode Sumber A. 1. Penjelasan secara rinci dari Pseudocode 4.3 adalah sebagai berikut:

1. Proses sinkronisasi diawali dengan mengambil data-data server dari berkas konfigurasi. Data-data ini diambil dari berkas *configuration.ini*.
2. Untuk menandakan jika proses sinkronisasi sedang berjalan, simpan inisialisasi log sinkronisasi pada tabel log sinkronisasi dengan status 0.
3. Cek apakah pada tabel log sinkronisasi terdapat data atau tidak. Jika kosong, maka sinkronisasi belum pernah dilakukan dan sistem akan melakukan proses inisialisasi. Proses inisialisasi adalah membuat tabel baru pada hbase dan menampung semua data pada mysql kedalam sebuah berkas *mysql_table.csv*. Kemudian berkas *mysql_table.csv* akan di import ke basis data Apache HBase.
4. Jika terdapat log, maka proses sinkronisasi sudah pernah dijalankan sebelumnya. Maka langkah selanjutnya adalah melakukan proses *patching* atau penambalan. *Patching* diawali dengan mengambil semua sintaks *query* pada berkas log di server basis data MySQL. Kemudian sintkas-sintaks *query* tersebut akan di konvert menjadi sintaks Apache Phoenix dan di simpan kedalam satu berkas *list_all_query.sql*. Selanjutnya *list_all_query.sql* akan di eksekusi pada basis data Apache HBase.
5. Jika proses telah berakhir, maka *update* status pada tabel log sinkronisasi untuk menandakan jika proses sinkronisasi sedang tidak berlangsung.
6. Catat waktu yang dihabiskan untuk melakukan proses sinkronisasi.

1	**get** mysql_db_conf_data
2	**get** hbase_db_conf_data
3	**get** sync_log_db_conf_data
4	**get** ssh_access_data
5	**get** last_sync_log_data
6	**set** initial_sync_log
7	**set** start_time

```
8   query_count = 0
9   do_patching = false
10
11  if (last_sync_log_data is empty)
12    dump mysql_table to mysql_table.csv
13    execute create_table.sql with apache phoenix
14    if (create_table is success)
15        execute mysql_table.csv to hbase with apache phoenix
16  else
17    ssh to mysql_db_server
18    get log_query_file
19    for line in log_query_file
20        get query_time
21        if (query_time > last_sync_log_data)
22            get query_syntax
23            query_tmp = convert query_syntax to apache phoenix
              syntax
24            append query_tmp to list_all_query.sql
25            do_patching = true
26    if (query_count is 0)
27        update sync_log
28    if (do_patching is true)
29        execute list_all_query.sql with apache phoenix
30        update sync_log
31  duration = time_now – start_time
```

Pseudocode 4.3 Proses Sinkronisasi

Pada proses patching, terdapat proses untuk melakukan perubahan sintaks *query* MySQL menjadi sintaks *query* Apache Phoenix karena terdapat beberapa *query* yang memiliki perbedaan struktur sintkas SQL. Secara garis besar, *query delete* dan *select* pada MySQL memiliki struktur yang sama dengan Apache Phoenix, namun Apache Phoenix menggunakan sintaks *upsert* untuk mengeksekusi perintah *insert* dan *update* yang ada pada

MySQL. Oleh karena itu diperlukan sebuah fungsi yang digunakan untuk melakukan konversi tersebut. Pseudocode 4.4 menunjukan konverter sintaks *query insert*. Untuk konverter sintkas *query update* dapat dilihat pada Pseudocode 4.5.

```
1  get insert_query
2  replace ‘INSERT’ to ‘UPSERT’ in insert_query
```

Pseudocode 4.4 Konverter Sintaks Query Insert MySQL ke Sintaks Query Apache Phoenix

Penjelasan secara rinci Pseudocode 4.4 adalah sebagai berikut :

1. Ambil sintaks *query* MySQL
2. Ganti string ‘*INSERT’* menjadi ‘*UPSERT’*

```
1  get update_str
2  table_name = select string between ‘UPDATE and ‘SET
3
4  list_table = get list primary key from file configuration
5  if  table_name in list_table
6     get table_primary_key
7     get index_table
8  else
9     return table not in list
10
11 kolom_value_list = select string between ‘SET and ‘WHERE
12 for i in kolom_value_list
13    split string i into kolom and value
14    append kolom to kolom_list
15    append value to value_list
16
17 where_clause  = select string after ‘WHERE’
18
```

19	final_string = **merge** string 'UPSERT', table_name, table_primary_key, kolom_list, value_list, and where_clause to apache phoenix update syntax
20	**return** final_string

Pseudocode 4.5 Konverter Sintaks Query Update ke Sintaks Query Apache Phoenix

Untuk perubahan *query* update, dibutuhkan elemen tambahan yaitu *primary key*. *Primary key* didapat dengan mengambilnya pada berkas konfigurasi. Penjelasan secara rinci Pseudocode 4.5 adalah sebagai berikut :

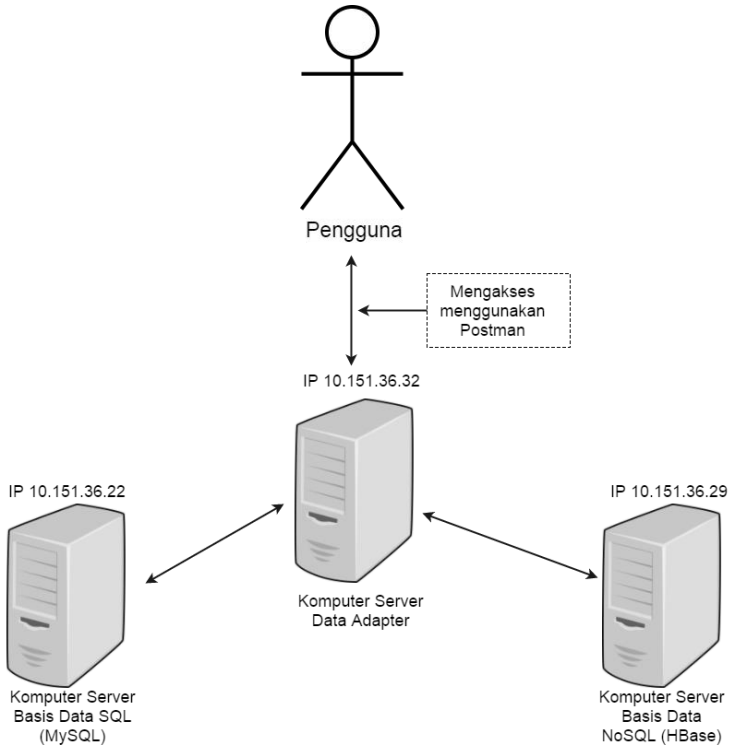
1. Pertama adalah mengambil string update MySQL.
2. Dari struktur update MySQL, nama tabel didapatkan dengan mengambil kata yang ada di antara '*UPDATE'* dan '*SET'*.
3. Selanjutnya adalah mengecek apakah nama tabel tersebut ada di list_table yang ada di basis data. Jika tabel terdapat dalam list maka ambil *primary key* tabel. Jika tidak, maka kembalikan nilai tabel tidak terdapat dalam list.
4. Mengambil nilai kolom dan value yang terletak di antara string '*SET'* dan '*WHERE'*. Dari *string* ini, kumpulkan kolom dan nilai menjadi masing-masing ke variabel *kolom _list* dan *value_list*.
5. Mengambil string yang diikuti setelah *string* '*WHERE'* dan simpan ke variabel *where_clause* untuk mendapatkan klause where atau kondisi dimana sintaks akan dieksekusi.
6. Menggabungkan semua variabel *table_name, primary_key, kolom_list, value_list,* dan *where_clause* menjadi satu baris sintaks *upsert*.

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan uji coba yang dilakukan pada aplikasi yang telah dikerjakan serta analisa dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, skenario uji coba yang meliputi uji kebenaran dan uji kinerja serta analisa setiap pengujian.

5.1 Lingkungan Uji Coba



Gambar 5.1 Desain Arsitektur Uji Coba Sistem

Lingkungan untuk pengujian menggunakan tiga buah server yang terdiri dari server basis data SQL, server basis data NoSQL dan server *data adapter* serta satu buah komputer sebagai komputer penguji. Proses pengujian dilakukan di Laboratorium Arsitektur dan Jaringan Komputer gedung Teknik Informatika ITS. Gambar 5.1 menunjukan desain arsitektur untuk proses uji coba yang digunakan. Spesifikasi perangkat keras dan perangkat lunak server ditunjukan pada Tabel 5.1 untuk server basis data SQL, Tabel 5.2 untuk server basis data NoSQL, Tabel 5.3 untuk spesifikasi server *data adapter*, dan Tabel 5.4 untuk spesifikasi komputer penguji.

Tabel 5.1 Spesifikasi Server Basis Data SQL

Server Basis Data SQL	
Komponen	**Spesifikasi**
Jenis	Komputer Fisik
Nama Perangkat	NARASOMA
Alamat IP	10.151.36.19
Processor	Intel(R) Core(TM) i3-530 CPU @ 2.39GHz (4 CPUs)
Memori	RAM 1,8 GB
Penyimpanan	52,9 GB
Grafis	Intel(R) Ironlake Desktop
Sistem Operasi	Ubuntu Desktop 14.04 LTS
Perangkat Lunak	1. MySQL Server

Tabel 5.2 Spesifikasi Server Basis Data NoSQL

Server Basis Data NoSQL	
Komponen	**Spesifikasi**
Jenis	Komputer Fisik
Nama Perangkat	BHISMA
Alamat IP	10.151.36.29
Processor	Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz (4 CPUs)
Memori	RAM 5,8 GB

Server Basis Data NoSQL	
Komponen	**Spesifikasi**
Penyimpanan	84,0 GB
Grafis	Gallium 0.4 on AMD CAICOS (DRM 2.43.0, LLVM 3.8.0)
Sistem Operasi	Ubuntu Desktop 14.04 LTS
Perangkat Lunak	1. MySQL Server
	2. Apache Hadoop 2.7.3
	3. Apache HBase 1.2.4
	4. Apache Phoenix 4.10.0
	5. Flask 0.12.1
	6. Python 2.7.6

Tabel 5.3 Spesifikasi Server Data adapter

Server *Data adapter*	
Komponen	**Spesifikasi**
Jenis	Komputer Fisik
Nama Perangkat	DUPADI
Alamat IP	10.151.36.32
Processor	Intel(R) Core(TM) 2 Duo-E4600 CPU @ 2.40GHz (2 CPUs)
Memori	RAM 1,9 GB
Penyimpanan	196,7 GB
Grafis	Intel(R) G33
Sistem Operasi	Ubuntu Desktop 16.04 LTS
Perangkat Lunak	1. MySQL Server
	2. Flask 0.12.1
	3. Python 2.7.6

Tabel 5.4 Spesifikasi Komputer Penguji

Server *Data adapter*	
Komponen	**Spesifikasi**
Jenis	Komputer Fisik
Nama Perangkat	USER

Server *Data adapter*	
Komponen	**Spesifikasi**
Processor	Intel(R) Core(TM) i5-3317U CPU @ 1.70GHz (4 CPUs)
Memori	RAM 4 GB
Penyimpanan	500 GB
Grafis	Intel(R) HD Graphics 4000
Sistem Operasi	Windows 8.1 Pro
Perangkat Lunak	1. Postman 4.10.7
	2. Heidi SQL 9.3.0.4984
	3. Microsoft Word 2013

5.2 Dataset Uji Coba

Pada tugas akhir ini, penulis menggunakan *data set* dari situs website *openflight.org*[28] yang berisi data-data tentang penerbangan diseluruh dunia. Data-data tersebut terdiri dari tiga tabel dengan rincian yaitu :

1. Tabel Routes
   a. Baris : 67.663
   b. Kolom :
      1) id_route (integer)
      2) airline (varchar)
      3) id_airline (integer)
      4) src_airport (varchar)
      5) id_src_airport (integer)
      6) dst_airport (varchar)
      7) id_dst_airport (integer)
      8) codeshare (varchar)
      9) stop_val (integer)
      10) equipment (varchar)
      11) log_date (date)
2. Tabel Airline
   a. Baris : 6.161
   b. Kolom :

1) id_airline (integer)
2) name (varchar)
3) alias (varchar)
4) iata (varchar)
5) icao (varchar)
6) callsign (varchar)
7) country (varchar)
8) active_stat (varchar)

3. Tabel Airport
   a. Baris : 7.184
   b. Kolom :
      1) airport_id (integer)
      2) name_airport (varchar)
      3) city (varchar)
      4) country (varchar)
      5) iata (varchar)
      6) icao (varchar)
      7) latitude (varchar)
      8) longitude (varchar)
      9) altitude (varchar)
      10) timezone (varchar)
      11) dst (varchar)
      12) tz_db (varchar)
      13) type_airport (varchar)
      14) source (varchar)

Data set tersebut disimpan dalam berkas dengan format .csv. Jumlah baris dan tabel tersebut tidak sepenuhnya digunakan tetapi dirubah sesuai dengan uji coba yang akan dilakukan.

5.3 Skenario Uji Coba

Skenario uji coba dilakukan dengan beberapa tahap uji coba yaitu uji coba fungsionalitas dan uji coba kapasitas dan peforma. Penjelasan secara rinci skenario uji coba dijelaskan pada sub bab berikut ini.

5.3.1 Skenario Uji Coba Fungsionalitas

Uji coba dilakukan dengan mengakses semua antar muka yang ada dan menguji fungsi sinkronisasi dari *data adapter*. Pengujian dilakukan oleh pengguna dengan menggunakan bantuan aplikasi Postman. Proses pengujian ini berguna untuk menguji apakah operasi-operasi perubahan dasar pada data dapat dilakukan dan memberikan hasil yang diharapkan. Operasi dasar dari *data adapter* dapat dikategorikan kedalam tiga fitur, yaitu sebagai berikut :

- **Inisialisasi Basis Data**
  Inisialisasi basis data akan membuat tabel baru pada basis data NoSQL yaitu HBase. Jika tabel pada HBase telah siap, maka sistem akan mengekspor data pada basis data SQL ke dalam sebuah berkas dengan format .csv. Data ini kemudian dieksekusi ke basis data NoSQL. Pengujian ini akan memastikan proses ini berjalan ketika proses sinkronisasi pada basis data SQL dan NoSQL belum pernah dilakukan sebelumnya, atau dalam kata lain sistem untuk pertama kalinya menjalankan proses sinkronisasi.
- **Transformasi Basis Data**
  Proses transformasi berlangsung ketika terdapat perubahan data pada basis data SQL. Perubahan terjadi dari adanya *query* dari aplikasi seperti *insert*, *update* dan *delete*. Jika sistem sebelumnya sudah pernah melakukan transformasi, maka proses inisialisasi tidak akan berlangsung. Sistem akan mengambil daftar *query* yang di eksekusi pada basis data SQL, kemudian dijalankan ke basis data NoSQL sehingga data pada kedua basis data adalah sama.
- **Antar Muka *Data adapter***
  Antar muka *data adapter* menyediakan rute yang dapat diakses oleh aplikasi untuk melakukan suatu proses kepada basis data. Fitur ini juga bertugas untuk memilih dari basis data mana permintaan aplikasi akan di proses. Pada

pengujian ini diharapkan setiap permintaan dari aplikasi yang berupa perubahan data, yaitu *query insert*, *update* dan *delete*, akan diarahkan ke basis data SQL dan untuk permintaan pengambilan data akan diarahkan ke basis data NoSQL. Tabel 5.5 menunjukan rancangan setiap aksi pengujian dan hasil atau nilai yang diharapkan.

Selain melakukan uji coba untuk mengetahui berhasil atau tidaknya antar muka berjalan, dilakukan pencatatan waktu respon yang diperlukan sistem sampai proses berakhir kedalam tabel. Waktu respon dicatat sebanyak lima kali percobaan, kemudian diambil nilai rata-ratanya.

Tabel 5.5 Aksi dan Hasil Harapan Setiap Fungsi Antar Muka

No	Pengendali	Uji Coba	Hasil Harapan
1	/sinkron	Melakukan sinkronisasi basis data	Status sinkronisasi berhasil, log sinkronisasi tersimpan. Isi data pada basis data SQL dan NoSQL sama
2	/insert_routes	Menambahkan satu baris baru ke tabel routes	Satu baris rute pada tabel routes bertambah
3	/delete_routes_by_id	Menghapus satu baris data pada tabel routes	Satu baris rute pada tabel route terhapus
4	/update_routes_by_id	Menyunting data pada tabel routes	Data pada tabel routes diperbarui nilainya
5	/select_all_routes	Mengambil semua baris data pada tabel routes	Semua data pada tabel routes terambil
6	/select_route_by_id/<id>	Mengambil satu baris data	Satu baris yang dipilih diambil dari tabel routes

No	Pengendali	Uji Coba	Hasil Harapan
		pada tabel routes	
7	/insert_airline	Menambahkan satu baris baru ke tabel airline	Satu baris data berhasil ditambahkan pada tabel airline
8	/delete_airline_by_id	Menghapus satu baris data pada tabel airline	Satu baris data berhasil dihapus dari tabel airline
9	/update_airline_by_id	Menyunting data pada tabel airline	Data pada tabel airline berhasil diperbarui
10	/select_all_airline	Mengambil semua baris data pada tabel airline	Semua data pada tabel airline terambil
11	/select_airline_by_id/<id>	Mengambil satu baris data pada tabenl airline	Satu baris yang dipilih berhasil diambil dari tabel airline
12	/insert_airport	Menambahkan satu baris data baru pada tabel airport	Satu baris data airport baru ditambahkan pada tabel airport
13	/delete_airport_by_id	Menghapus satu baris data pada tabel airport	Satu baris data airport berhasil dihapus dari tabel airport
14	/update_airport_by_id	Menyunting data pada tabel airport	Data pada tabel airport berhasil diperbarui
15	/select_all_airport	Mengambil semua data	Semua data pada tabel airport berhasil diambil

No	Pengendali	Uji Coba	Hasil Harapan
		pada tabel airport	
16	/select_airport_by_id/<id>	Mengambil satu baris data pada tabel airport	Satu baris yang dipilih berhasil diambil dari tabel airport
17	/select_routes_country/<id_route>	Mengambil detail nama airline dan negara pada rute	Satu baris yang dipilih berhasil diambil dari tabel routes dan airline
18	/select_all_number_routes_by_airline	Mengambil jumlah rute yang ditempuh oleh airline	Semua data jumlah rute berhasil diambil dari tabel route dan airline
19	/select_all_number_routes_more_than/<number>	Mengambil airline yang memiliki jumlah rute lebih dari jumlah yang dimaksud	Semua data jumlah rute yang melebihi berhasil diambil dari tabel route
20	/select_all_airline_in_routes	Mengambil data airline yang melakukan penerbangan	Semua data jumlah rute tiap airline berhasil diambil dari tabel route
21	/select_all_airline_source	Mengambil detail airline dan asal penerbangan yang tercatat pada rute	Semua data airline dan asal penerbangan dari gabungan tabel route, airline dan airport

5.3.2 Skenario Uji Coba Kapasitas dan Performa

Pada pengujian ini dilakukan untuk menguji kemampuan tiap-tiap server yang ada pada sistem adapter. Parameter yang menjadi acuan pengukuran adalah dengan menghitung penggunaan CPU, *memory* dan waktu respon selama proses inisialisasi dan transformasi. Hasil uji coba ini disimpan dalam tabel dan ditampilkan dalam bentuk grafik sehingga dapat terlihat dengan jelas perubahan dari setiap skenario yang dijalankan. Uji coba menggunakan arsitektur yang sama seperti yang ditampilkan pada Gambar 5.1.

Komputer server diuji secara bertahap, mulai dari proses inisialisasi maupun transformasi *query*. Kumpulan data yang didapat diolah untuk mendapatkan parameter yang diinginkan. Pengujian dilakukan dengan beberapa skenario sebagai berikut:

- **Perbandingan Jumlah Baris pada Tabel**
  Pengujian ini dilakukan untuk mengetahui berapa jumlah sumber daya yang digunakan dan waktu respon selama proses transformasi data dari basis data MySQL ke HBase saat proses inisialisasi berlangsung dengan perbandingkan jumlah baris. Perbandingan jumlah baris yang digunakan pada tabel untuk setiap percobaan adalah sebagai berikut :
  - Airplane : 6.000, 12.000, 18.000, 24.000, 30.000
  - Airport : 7.000, 14.000, 21.000, 28.000, 35.000
  - Route : 60.000, 120.000, 180.000, 240.000, 300.000
- **Perbandingan Jumlah Tabel pada Basis Data**
  Pengujian ini dilakukan untuk mengetahui berapa jumlah sumber daya yang digunakan dan waktu respon sistem *data adapter* selama proses inisialisasi data dari basis data MySQL ke HBase berlangsung dengan perbandingan perbedaan jumlah tabel. Jumlah baris *default* yang digunakan pada setiap tabel adalah sesuai dengan jumlah baris yang didapat pada *openflight.org* [28]. Perbandingan jumlah tabel yang digunakan untuk setiap percobaan adalah 3, 6, 9, 12, dan 15 tabel.

- **Perbandingan Jumlah *Query Insert, Update* dan *Delete***
  Pengujian ini bertujuan untuk mengetahui seberapa sumberdaya dan waktu yang dihabiskan sistem untuk mengeksekusi tiga operasi *query* yang berbeda yaitu *insert*, *update* dan *delete* dari MySQL ke HBase pada proses tranformasi data. Pada setiap operasi *query*, perbandingan jumlah *query* yang digunakan adalah dimulai dari 1.000, 2.000, 3.000, 4.000 dan 5.000 *query*.

5.4 Hasil Uji Coba dan Evaluasi

Pada subbab ini akan dijelaskan mengenai hasil pengujian yang dilakukan sesuai dengan skenario pengujian yang telah ditentukan. Pengujian yang dilakukan adalah uji coba fungsionalitas dan uji coba kapasitas dan performa.

5.4.1 Hasil Uji Coba Fungsionalitas

Uji coba fungsionalitas dilakukan oleh pengguna menggunakan bantuan aplikasi Postman. Waktu respon didapat dari perhitungan ketika memulai proses dan mengakhiri proses pada kode program dan dilihat melalui terminal server.

- **Inisialisasi Basis Data**
  Pengujian ini dilakukan dengan melakukan transformasi data untuk pertama kali. Untuk mengujinya, data pada tabel log sinkronisasi di kosongkan dan tabel yang ada di basis data HBase dikosongkan. Kemudian isi data dari kedua basis data dicek apakah sudah sama atau belum Dari pengujian yang dilakukan, proses transformasi berjalan dengan sukses. Log proses selama proses sinkronisasi akan ditampilkan pada terminal. Gambar 5.2 menunjukan hasil sinkronisasi pada terminal.

```
sinkron terakhir:  0
waktu sekarang:  2017-05-22 15:11:15.602675
time stamp:  2017-05-22 15:11:15
sql :  INSERT INTO log_sinkronisasi (waktu, ip_src, ip_dst, status) VALU
17-05-22 15:11:15', '10.151.36.22', '10.151.36.29', '0')
Belum pernah sinkronisasi
sql:  SELECT * FROM routes2
file_output:  file/csv/fetch_all_routes2.csv
Berhasil, tabel  routes2  berhasil di dump ke fetch_all_ routes2 .csv
sql:  SELECT * FROM airline2
file_output:  file/csv/fetch_all_airline2.csv
Berhasil, tabel  airline2  berhasil di dump ke fetch_all_ airline2 .csv
sql:  SELECT * FROM airport2
file_output:  file/csv/fetch_all_airport2.csv
Berhasil, tabel  airport2  berhasil di dump ke fetch_all_ airport2 .csv
Data berhasil di export!
Inisialisasi data awal..
Mentransformasi data ke HBase...
Tabel berhasil dibuat di HBase
Tabel routes2 berhasil diimport ke HBase...
Tabel airline2 berhasil diimport ke HBase...
Tabel airport2 berhasil diimport ke HBase...
Proses inisialisasi selesai..
Durasi sinkronisai : 33.0158581734 seconds
Done
EXIT!

10.151.36.32 - - [22/May/2017 15:11:48] "GET / HTTP/1.1" 200 -
```

Gambar 5.2 Tampilan pada Terminal Ketika Proses Inisialisasi Berlangsung

- **Transformasi Basis Data (*Pathcing*)**
  Pengujian ini dilakukan jika sebelumnya proses inisialisasi sudah pernah dilakukan, karena sistem terlebih dahulu akan mengecek dari tabel log untuk memastikan jika proses sinkronisasi sudah pernah dilakukan sebelumnya. Pengujian dilakukan dengan menjalankan beberapa perintah *query* seperti *insert*, *update* dan *delete*. Kemudian dilakukan pengecekan apakah sistem berhasil membaca *log query* dari basis data SQL, mengkonversi sintaks *query* tersebut menjadi sintaks *query* Apache Phoenix, dan menyimpan semua daftar query pada berkas .csv. Dari berkas .csv ini, Apache Phoenix kemudian mengeksekusinya ke basis data HBase. Setelah proses transformasi ini, isi data pada kedua basis data di cek untuk

memastikan jika kedua basis data memiliki data yang sama. Dari pengujian yang telah dilakukan, proses transformasi berjalan dengan sukses. Gambar 5.3 menampilkan log pada terminal selama proses transformasi berlangsung.

```
0.151.36.32 - - [23/May/2017 10:13:29] "GET / HTTP/1.1" 200 -
LF4J: Class path contains multiple SLF4J bindings.
LF4J: Found binding in [jar:file:/usr/lib/phoenix/apache-phoenix-4.10.0-HBase-1.2-bi
/phoenix-4.10.0-HBase-1.2-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
LF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log
j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
LF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
7/05/23 10:15:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
 your platform... using builtin-java classes where applicable
inkron terakhir:  2017-05-23 10:13:25
aktu sekarang:  2017-05-23 10:15:06.701635
ime stamp:  2017-05-23 10:15:06
ql :  INSERT INTO log_sinkronisasi (waktu, ip_src, ip_dst, status) VALUES ('2017-05-
3 10:15:06', '10.151.36.22', '10.151.36.29', '0')
inkronisasi terakhir:  2017-05-23 10:13:25
engambil data Query dari log ...
ek aksi INSERT, UPDATE, DELETE pada log...
sync] INSERT  airport2
sync] INSERT  airport2
sync] INSERT  airport2
sync] UPDATE  airport2
ama tabel:  airport2
K:  airport_id
sync] DELETE  FROM
sync] DELETE  FROM
inkronisasi, DELETE, INSERT dan UPDATE file on progress sync to HBase...
pdate status sinkronisasi berhasil
ync berhasil dilakukan.
urasi sinkronisai : 4.45966291428 seconds
one
XIT!
```

Gambar 5.3 Tampilan pada Terminal Ketika Proses Transformasi Berlangsung

- **Antar Muka *Data adapter***

  Pengujian dilakukan dengan mengakses rute yang telah disediakan sistem dan disesuaikan dengan hasil harapan yang didapatkan. Daftar antar muka yang diuji berdasarkan alamat rute pengendali ditunjukan pada Tabel 5.6.

Tabel 5.6 Hasil Uji Coba Fungsionalitas

No	Pengendali	Uji Coba	Hasil
1	/sinkron	Melakukan sinkronisasi basis data	Sukses
2	/insert_routes	Menambahkan satu baris baru ke tabel routes	Sukses
3	/delete_routes_by_id	Menghapus satu baris data pada tabel routes	Sukses
4	/update_routes_by_id	Menyunting data pada tabel routes	Sukses
5	/select_all_routes	Mengambil semua baris data pada tabel routes	Sukses
6	/select_route_by_id/<id>	Mengambil satu baris data pada tabel routes	Sukses
7	/insert_airline	Menambahkan satu baris baru ke tabel airline	Sukses
8	/delete_airline_by_id	Menghapus satu baris data pada tabel airline	Sukses
9	/update_airline_by_id	Menyunting data pada tabel airline	Sukses
10	/select_all_airline	Mengambil semua baris data pada tabel airline	Sukses
11	/select_airline_by_id/<id>	Mengambil satu baris data pada tabel airline	Sukses
12	/insert_airport	Menambahkan satu baris data baru pada tabel airport	Sukses
13	/delete_airport_by_id	Menghapus satu baris data pada tabel airport	Sukses
14	/update_airport_by_id	Menyunting data pada tabel airport	Sukses

No	Pengendali	Uji Coba	Hasil
15	/select_all_airport	Mengambil semua data pada tabel airport	Sukses
16	/select_airport_by_id/<id>	Mengambil satu baris data pada tabel airport	Sukses
17	/select_routes_country/<id_route>	Mengambil satu baris data rute beserta detail negara airline	Sukses
18	/select_all_number_routes_by_airline	Mengambil semua data airline dan jumlah rute	Sukses
19	/select_all_number_routes_more_than/<number>	Mengambil semua jumlah penerbangan yang lebih dari pilihan	Sukses
20	/select_all_airline_in_routes	Mengambil semua data airline yang ada pada route	Sukses
21	/select_all_airline_source	Mengambil semua data airline dan asal penerbangan	Sukses

Dari skenario uji coba yang diberikan pada sub bab sebelumnya, hasil uji coba antar muka sistem menunjukan 100% sukses dilaksanakan. Rute yang ada berhasil diakses dan memberikan output sesuai dengan yang diharapkan, dan kondisi data pada kedua basis data setelah di sinkronisasi adalah sama.

Pencatatan waktu respon dibagi menjadi tiga kategori tabel yang berbeda. Kategori pertama adalah mencatat waktu untuk rute yang melakukan *query* perubahan pada basis data. Rute yang termasuk dalam kategori ini adalah semua rute yang menjalankan *query insert*, *update* dan *delete*. *Data adapter* mengarahkan *query* ke basis data MySQL untuk mengeksekusi *query* pada kategori ini. Tabel 5.7 menunjukan waktu respon dari rute yang melakukan

perubahan data. Data yang digunakan adalah data dari *openflight.org* [28] tanpa adanya perubahan jumlah baris.

Tabel 5.7 Hasil Waktu Respons Uji Coba Fungsionalitas untuk Perubahan Data

No	Nama Interfaces	Waktu Respons (ms)					
		1	2	3	4	5	Rata-rata
1	POST /insert_routes	313	59	75	49	58	110,8
2	POST /delete_routes_by_id	259	232	256	257	271	255,0
3	POST /update_routes_by_id	323	241	266	255	257	268,4
4	POST /insert_airline	55	122	58	67	63	73,0
5	POST /delete_airline_by_id	81	100	62	65	71	75,8
6	POST /update_airline_by_id	81	78	81	80	72	78,4
7	POST /insert_airport	88	69	54	33	60	60,8
8	POST /delete_airport_by_id	90	91	87	143	345	151,2
9	POST /update_airport_by_id	85	86	107	134	92	100,8

Kategori yan kedua adalah pengambilan data pada basis data SQL. Pada kategori ini, semua *query* yang berkaitan dengan

pengambilan data, atau *query select*, akan diarahkan ke MySQL. Hal ini terjadi ketika proses sinkronisasi sedang berlangsung. Tabel 5.8 menunjukan catatan waktu respon untuk pengambilan data pada basis data MySQL.

Dan kategori yang terakhir adalah pengambilan basis data pada basis data NoSQL. Proses pengambilan data ini terjadi jika proses sinkronisasi sedang tidak berlangsung. *Query* select akan diarahkan ke HBase dan hasil uji coba dicatat pada Tabel 5.9. Grafik perbandingan waktu respon proses pengambilan data pada MySQL dan HBase ditampilkan pada Gambar 5.4.

Tabel 5.8 Waktu Respon Uji Fungsionalitas Proses Pengambilan Data pada Basis Data MySQL

No. Antarmuka	**Nama Interfaces**	**Waktu Respons (s)**					
		1	**2**	**3**	**4**	**5**	**Rata-rata**
1	GET /select_all_routes	701	708	698	686	728	704,2
2	GET /select_route_by_id/<id>	53	56	57	51	52	53,8
3	GET /select_all_airline	89	53	56	54	67	63,8
4	GET /select_airline_by_id/<id>	18	28	30	12	22	22,0
5	GET /select_all_airport	126	129	169	129	139	138,4
6	GET /select_airport_by_id/<id>	20	22	15	17	52	25,2

No. Antar muka	Nama Interfaces	Waktu Respons (s)					
		1	2	3	4	5	Rata-rata
7	GET /select_routes_country/<id>	345	343	336	331	323	335,6
8	GET /select_all_number_routes_by_airline	21340	21338	21308	21308	21340	21326,8
9	GET /select_all_number_routes_more_than/<number>	46	49	50	66	55	53,2
10	GET /select_all_airline_in_routes	49	45	49	64	54	52,2
11	GET /select_all_airline_source	TLE	TLE	TLE	TLE	TLE	TLE

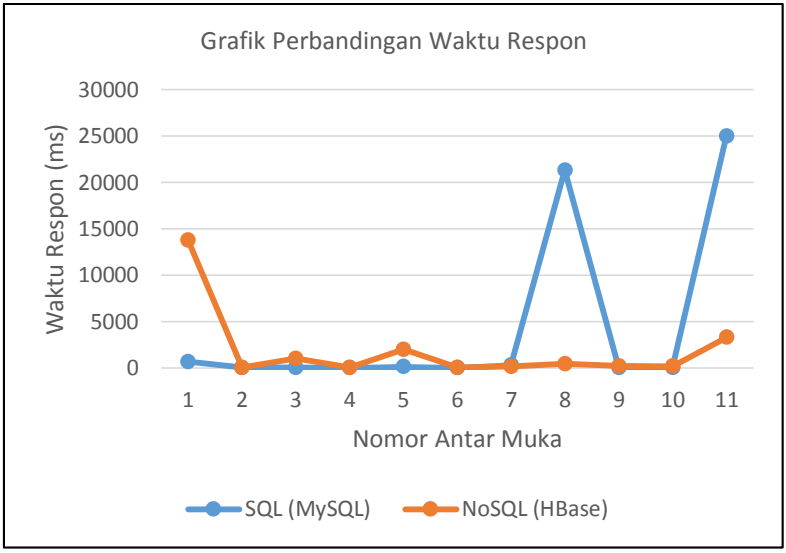
Keterangan :

- *TLE : Time Limit Exceeded*

Tabel 5.9 Waktu Respon Uji Fungsionalitas Proses Pengambilan Data pada Basis Data HBase

No. Antar muka	Nama Interfaces	Waktu Respons (ms)					
		1	2	3	4	5	Rata-rata
1	GET /select_all_routes	13873	3784	13711	13762	13772	13780,4

No. Antar muka	Nama Interfaces	Waktu Respons (ms)					
		1	2	3	4	5	Rata-rata
2	GET /select_route_by_id/<id>	52	46	52	50	50	50,0
3	GET /select_all_airline	1148	1006	1004	998	998	1030,8
4	GET /select_airline_by_id/<id>	84	94	47	50	45	64,0
5	GET /select_all_airport	2031	1990	2058	2024	2018	2024,2
6	GET /select_airport_by_id/<id>	99	72	60	57	49	67,4
7	GET /select_routes_country/<id>	228	207	180	158	181	190,8
8	GET /select_all_number_routes_by_airline	480	455	451	449	444	455,8
9	GET /select_all_number_routes_more_than	208	237	216	229	222	222,4
10	GET /select_all_airline_in_routes	211	207	216	212	215	212,2
11	GET /select_all_airline_source	3372	3367	3222	3302	3321	3316,8



Gambar 5.4 Grafik Perbandingan Waktu Respon Rata-rata Basis Data SQL (MySQL) dengan NoSQL (HBase)

Dari hasil uji coba yang dilakukan, terdapat beberapa *query* dengan perbedaan waktu respon yang signifikan seperti pada nomor antar muka 1, 8, dan 11. Pada nomor antar muka 11, waktu respon untuk pengambilan data pada MySQL melebihi dari batas waktu yang ditentukan atau TLE (*time limit exceeded*). Waktu respon untuk pengambilan data ke basis data SQL lebih cepat pada nomor antar muka 1, 3, 4, 5, 6, 9, dan 10. Waktu respon untuk pengambilan data pada NoSQL lebih cepat pada nomor antar muka 2, 7, 8, dan 11.

5.4.2 Hasil Uji Coba Kapasitas dan Performa

Seperti yang dijelaskan pada sub bab sebelumnya, terdapat lima skenario yang digunakan untuk melakukan uji coba kapasitas dan performa. Untuk mendapatkan hasil yang akurat, pengujian

dilakukan sebanyak sepuluh kali pada setiap nomor percobaan, kemudian dihitung nilai rata-ratanya dan dicatat dalam tabel. Hasil pengujian disimpan kedalam tabel Hasil uji coba yang telah dilakukan dijelaskan secara rinci pada sub bab berikut:

5.4.2.1 Perbandingan Jumlah Baris dalam Tabel

Tabel 5.10 Hasil Uji Kapasitas dan Performa pada Komputer Data adapter (DB Converter) dengan Perbandingan Jumlah Baris

No	Jumlah Baris	Rata-rata Memori (MB)	Rata-rata Processor (%)	Waktu Transformasi (s)
1	AL : 6.000 AP : 7.000 RT : 60.000	2.293,4	35,727	31,251
2	AL : 12.000 AP : 14.000 RT : 120.000	2.371,5	36,744	37,638
3	AL : 18.000 AP : 21.000 RT : 180.000	2.426,0	37,482	43,293
4	AL : 24.000 AP : 28.000 RT : 240.000	2.483,9	39,248	49,709
5	AL : 30.000 AP : 35.000 RT : 300.000	2.579,4	38,850	54,652

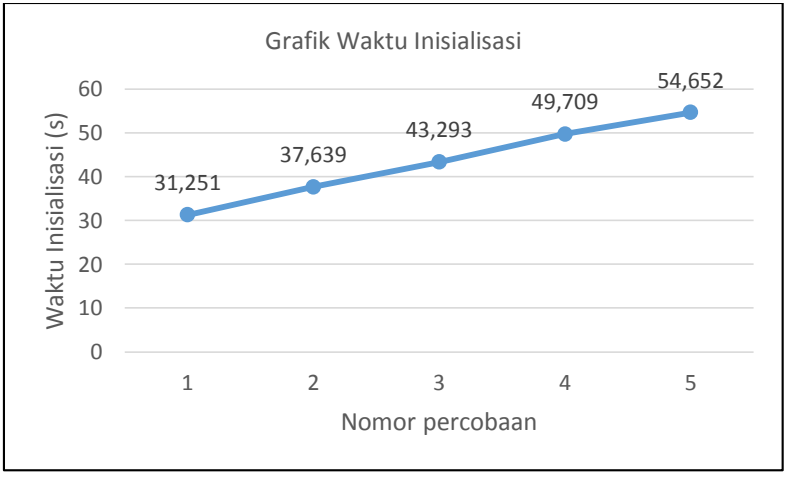
Keterangan :

- *AL : Tabel Airline*
- *AP : Tabel Airport*
- *RT : Tabel Route*

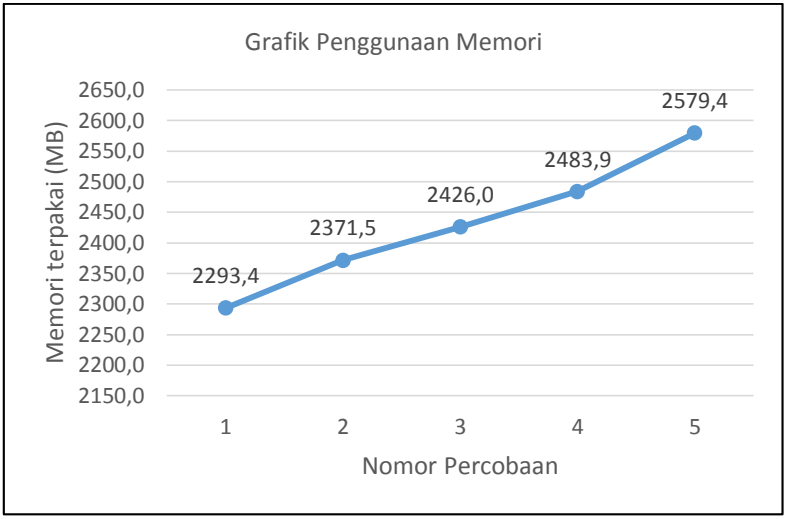
Pengujian ini dilakukan dengan menggunakan kumpulan data yang memiliki jumlah baris yang berbeda. Pengujian diawali dengan baris dengan jumlah 6.000 pada tabel Airplane, 7.000 pada tabel Airport dan 60.000 pada tabel Route. Jumlah baris bertambah

sesuai dengan skenario. Waktu respon dicatat mulai dari awal proses sampai dengan proses inisialisasi telah berakhir. Tabel 5.10 menunjukan hasil sumber daya yang digunakan dan waktu respon saat proses inisialisasi data dari MySQL ke HBase.

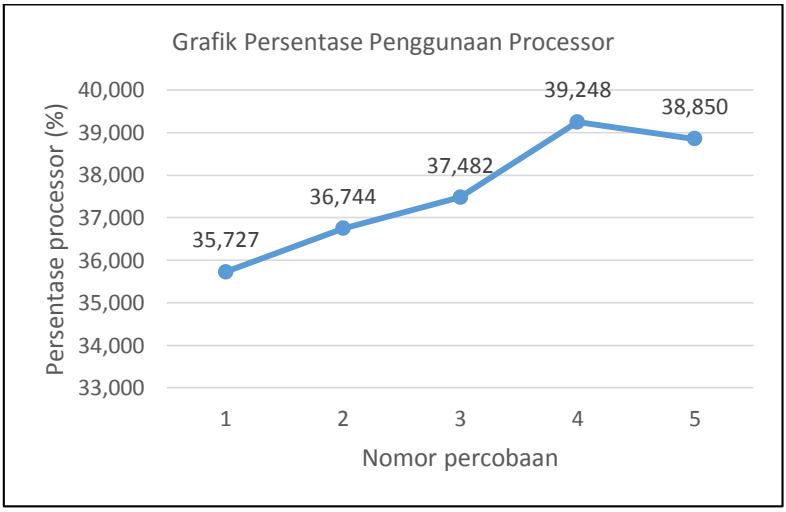
Dari hasil pengujian yang didapatkan, waktu respon yang dibutuhkan untuk melakukan inisialisasi meningkat secara konsisten, mulai dari 31,251 detik untuk uji coba pertama dan naik bertahap sampai mencapai 54,652 detik pada skenario terakhir. Grafik peningkatan waktu respon ini dapat dilihat pada Gambar 5.5. Perubahan penggunaan memori RAM menunjukan angka perubahan yang tidak terlalu signifikan dengan persentase terendah adalah 2.293,4 MB dan nilai tertinggi sebesar 2.579,4 MB. Grafik penggunaan memori dapat dilihat pada Gambar 5.6. Untuk persentase *processor* yang digunakan, nilai persentase ditunjukan pada Gambar 5.7. Persentase processor yang digunakan mengalami peningkatan secara linear dari angka terendah adalah 35,727% sampai dengan angka tertinggi 39,248%, namun mengalami penurunan pada pengujian ke-lima menjadi 38,850%.



Gambar 5.5 Grafik Waktu Inisialisasi Berdasarkan Perbandingan Jumlah Baris



Gambar 5.6 Grafik Penggunaan Memori Berdasarkan Perbandingan Jumlah Baris



Gambar 5.7 Grafik Persentase Penggunaan Processor Berdasarkan Perbandingan Jumlah Baris

5.4.2.2 Perbandingan Jumlah Tabel pada Basis Data

Pengujian dilakukan dengan menggunakan kumpulan data yang memiliki jumlah tabel berbeda, sesuai dengan skenario pada sub bab sebelumnya. Penambahan jumlah tabel dilakukan dengan jumlah baris yang sama. Waktu respon dicatat mulai dari awal proses sampai dengan proses inisialisasi telah berakhir. Tabel 5.11 menunjukan hasil sumber daya yang digunakan dan waktu respon saat proses inisialisasi data dari MySQL ke HBase.

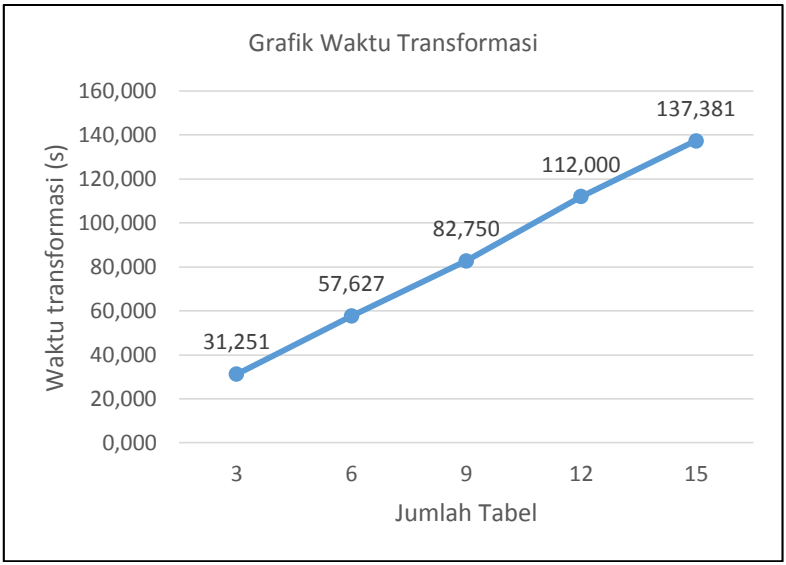
Tabel 5.11 Hasil Uji Kapasitas dan Performa pada Komputer Data adapter (DB Converter) dengan Perbandingan Jumlah Tabel

No	Jumlah Tabel	Rata-rata Memori (MB)	Rata-rata Processor (%)	Waktu Transformasi (s)
1	3	2.293,4	35,727	31,251
2	6	2.359,5	36,156	57,627
3	9	2.534,6	37,133	82,750
4	12	2.771,6	38,867	112,000
5	15	2.884,3	37,956	137,381

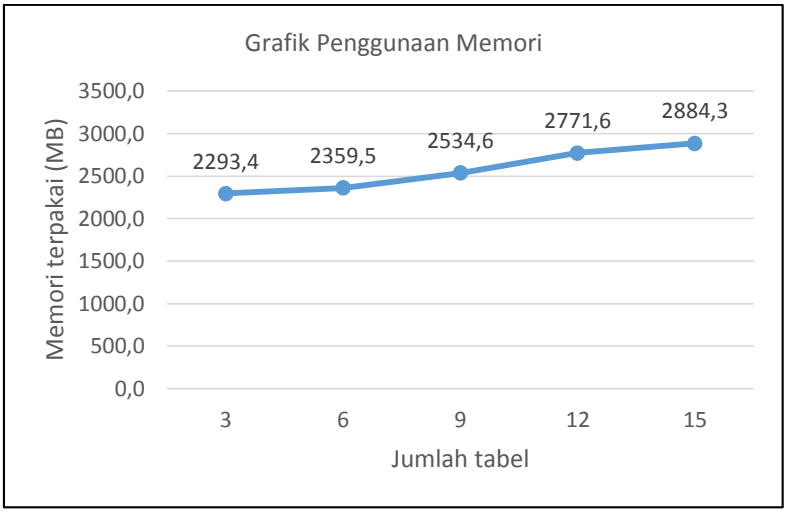
Dari hasil pengujian, waktu respon yang dibutuhkan untuk melakukan inisialisasi adalah semakin meningkat untuk penambahan jumlah tabel, namun memiliki selisih yang lebih besar jika dibandingan pengujian dengan perbedaan jumlah baris. Waktu transformasi tercepat adalah 31,251 detik pada jumlah tabel tiga dan waktu transformasi paling lama adalah 137,381 detik pada jumlah tabel 15. Grafik peningkatan waktu respon ini dapat dilihat pada Gambar 5.8.

Dari penggunaan sumber daya memori RAM, nilai menunjukan peningkatan untuk setiap penambahan jumlah tabel mulai dari 2.293,4 MB sampai dengan 2.884,3 MB. Gambar 5.9 menampilkan grafik penggunaan memori untuk pada uji coba perbandingan jumlah tabel. Sementara itu untuk penggunaan

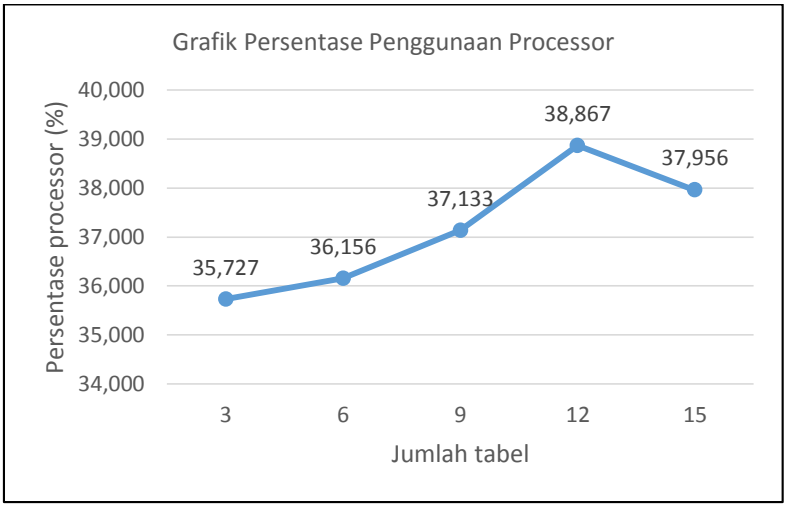
processor, angka yang dihasilkan tidak menunjukan kecenderungan. Rata-rata persentase penggunaan processor yang digunakan berada di antara 42,295% sampai dengan 55,210%. Gambar 5.10 menampilkan grafik persentase penggunaan processor pada uji coba ini.



Gambar 5.8 Grafik Waktu Transformasi Berdasarkan Perbandingan Jumlah Tabel



Gambar 5.9 Grafik Penggunaan Memori Berdasarkan Perbandingan Jumlah Tabel



Gambar 5.10 Grafik Persentase Penggunaan Processor Berdasarkan Perbandingan Jumlah Tabel

5.4.2.3 Perbandingan Jumlah *Query* Insert, Update dan Delete

Pengujian yang pertama pada sub bab ini adalah dengan mengeksekusi sejumlah *query insert* pada sistem. Selanjutnya, sistem akan melakukan proses *patching* dengan mentransformasikan semua *query* yang dieksekusi pada basis data SQL ke basis data NoSQL. Untuk transformasi sejumlah *query* insert, hasil uji coba ditunjukan pada Tabel 5.12.

Tabel 5.12 Hasil Uji Kapasitas dan Performa pada Komputer Data adapter (DB Converter) dengan Perbandingan Jumlah Query Insert

No	Jumlah *Query Insert*	Rata-rata Memori (MB)	Rata-rata Processor (%)	Waktu Transformasi (s)
1	1.000	1.736,6	30,450	8,518
2	2.000	1.911,4	28,438	14,248
3	3.000	1.985,8	26,664	22,244
4	4.000	2.064,5	24,601	30,540
5	5.000	2.188,4	22,576	41,334

Hal yang sama dilakukan untuk proses transformasi sejumlah *query update*. Hasil uji coba transformasi sejumlah *query* update dapat dilihat pada Tabel 5.13.

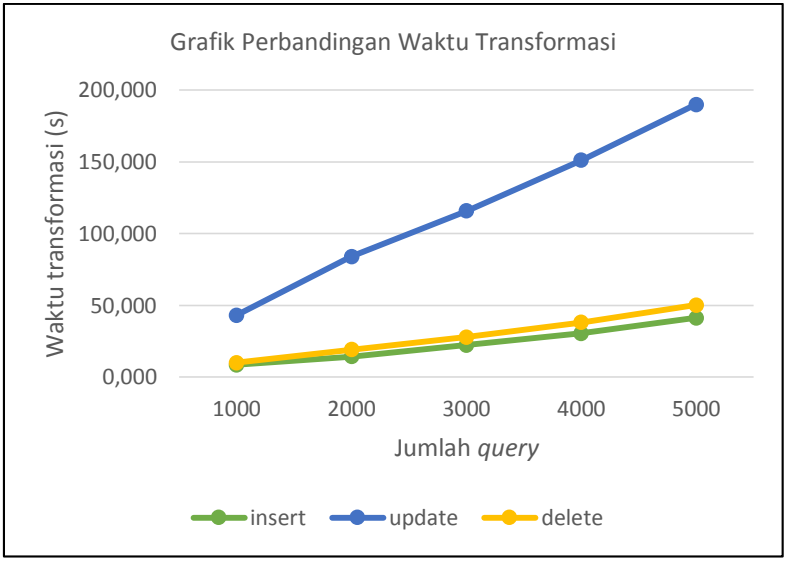
Tabel 5.13 Hasil Uji Kapasitas dan Performa pada Komputer Data adapter (DB Converter) dengan Perbandingan Jumlah Query Update

No	Jumlah *Query Update*	Rata-rata Memori (MB)	Rata-rata Processor (%)	Waktu Transformasi (s)
1	1.000	1.961,8	20,239	43,130
2	2.000	2.151,7	16,198	84,050
3	3.000	2.216,3	15,556	115,662
4	4.000	2.374,1	14,863	151,028
5	5.000	2.431,8	14,186	189,764

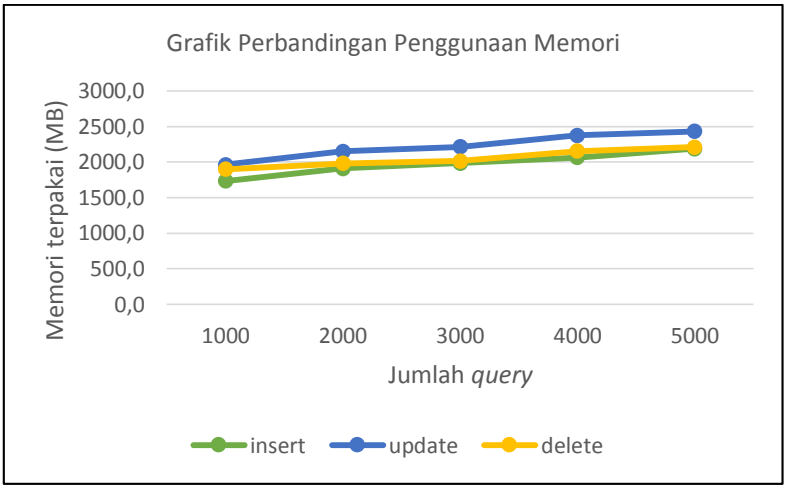
Operasi terakhir yang diuji adalah melakukan transformasi untuk sejumlah *query delete*. Hasil uji coba kapasitas dan performa untuk perbandingan jumlah *query delete* dapat dilihat pada Tabel 5.14.

Tabel 5.14 Hasil Uji Kapasitas dan Performa pada Komputer Data adapter (DB Converter) dengan Perbandingan Jumlah Query Delete

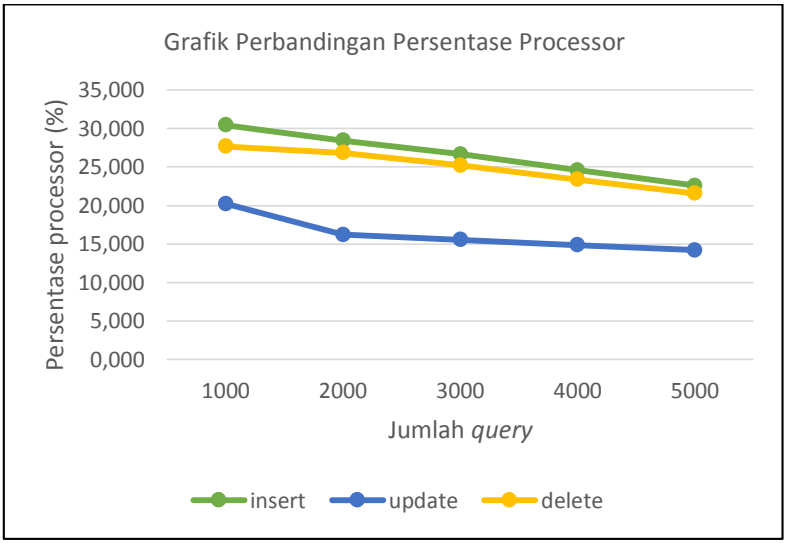
No	Jumlah *Query Delete*	Rata-rata Memori (MB)	Rata-rata Processor (%)	Waktu Transformasi (s)
1	1.000	1.899,0	27,685	9,996
2	2.000	1.983,4	26,864	19,106
3	3.000	2.016,7	25,216	27,826
4	4.000	2.154,4	23,382	38,169
5	5.000	2.212,8	21,567	50,251



Gambar 5.11 Grafik Perbandingan Waktu Transformasi pada Transformasi Query Insert, Update dan Delete



Gambar 5.12 Grafik Perbandingan Penggunaan Memori pada Transformasi Query Insert, Update dan Delete



Gambar 5.13 Grafik Perbandingan Persentase Processor pada Transformasi Query Insert, Update dan Delete

Dari hasil percobaan, waktu yang dibutuhkan untuk proses transformasi pada operasi *insert* dan *delete* memiliki nilai yang hampir sama. Sedangkan untuk proses transformasi operasi *update* memiliki waktu yang lebih lama dan meningkat secara linear. Hal ini dapat dilihat pada Gambar 5.11 dimana waktu yang dibutuhkan untuk transformasi *update* adalah 43,130 detik untuk 1.000 *query* dan meningkat sampai dengan 189,764 detik untuk jumlah *query* 5.000. Sementara untuk operasi *insert*, waktu yang dibutuhkan lebih singkat dibandingkan dengan operasi *update* yaitu dari 8,518 pada 1.000 *query* dan meningkat secara bertahap sampai dengan 41,334 detik untuk 5.000 *query*. Hasil yang hampir sama ditunjukan oleh operasi *delete*, yaitu 9,996 detik pada 1.000 *query* dan meningkat secara bertahap sampai dengan 50,251 detik untuk 5.000 *query*.

Sumber daya memori yang digunakan untuk operasi *insert*, *update* dan *delete* memiliki nilai penggunaan memori yang hampir sama yaitu berada diantara angka 1.736,6 MB sampai 2.431,8 MB. Gambar 5.12 menunjukan perbandingan penggunaan memori ketiga operasi adalah meningkat secara linear.

Sementara itu hasil berbeda ditunjukan untuk persentase penggunaan *processor* atau CPU. Ketiga operasi menunjukan pengurangan nilai persentase processor untuk setiap peningkatan jumlah *query* yang ditransformasikan. Operasi *query update* memiliki persentase yang paling rendah jika dibandingkan dengan operasi *insert* dan *delete*, yaitu berada dibawah angka 25% yaitu persentase tertinggi pada angka 20,239% dan terendah 14,186%. Sementara untuk operasi *insert* dan *delete*, persentase processor yang digunakan berada dibawah angka 35% dan diatas 20%. Untuk *query insert*, penggunaan processor tertinggi adalah 30,450% dan terendah 22,576%. Operasi *delete* memakan persentase processor tertinggi pada 27,685% dan yang terendah sebanyak 21,567%. Gambar 5.13 menunjukan grafik persentase penggunaan processor untuk ketiga operasi.

5.4.3 Evaluasi

Berdasarkan hasil uji coba, sistem dapat berjalan sesuai dengan skenario pada sub bab 5.3. Dari pengujian fungsionalitas, semua fungsi dapat berjalan sesuai dengan harapan. Namun, untuk proses pengambilan data pada MySQL, terdapat satu antar muka dengan waktu respon yang melebih batas. MySQL tidak cukup tangguh untuk menangani *query* yang membutuhkan pengambilan data ke lebih dari satu tabel pada tabel yang memiliki jumlah data yang banyak. Disisi lain, HBase mampu menangani semua permintaan pemrosesan *query* dari setiap antar muka yang diujicobakan.

Dari sisi kapasitas dan performa, untuk semua skenario yang diujikan, jumlah persentase sumber daya memori RAM yang digunakan menunjukan peningkatan secara linear, tidak ada perubahan yang melonjak naik atau menurun secara signifikan, berkisar pada angka 1.736,6 MB sampai dengan 2.431,8 MB. Sementara sumber daya *processor* yang digunakan menunjukan penurunan persentase untuk setiap peningkatan jumlah *query* yang ditransformasi. Penurunan ini juga ditunjukan pada hasil uji coba untuk perbandingan jumlah baris dan perbandingan jumlah tabel pada saat proses inisialisasi. Namun perbedaanya, penurunan terjadi pada pengujian yang ke-empat. Hal ini karena Apache Phoenix hanya membutuhkan persentase processor yang tinggi ketika proses diawal atau saat Apache Phoenix mulai melakukan eksekusi *query*. Sisanya, selama proses sinkronisasi berlangsung, nilai persentase penggunaan cendrung lebih rendah dan bergerak pada angka yang konsisten. Apache Phoenix membutuhkan lebih banyak sumber daya memori RAM untuk melakukan proses transformasi dibandingkan dengan penggunaan persentase processor.

Rata-rata persentase processor yang digunakan untuk proses inisialisasi adalah 37,389%. Proses transformasi *query* mengabiskan rata-rata sumber daya processor lebih sedikit dibandingkan proses inisialisasi, yaitu 26,546% untuk *query insert*,

16,208% untuk *query update* dan 24,943% untuk *query delete*. Proses transformasi *query update* memiliki angka persentase penggunaan processor paling kecil, namun berkebalikan dengan waktu transfomasinya yang paling lama diantara operasi yang lain. Rata-rata waktu transformasi yang dibutuhkan untuk setiap operasi adalah 23,4 detik untuk *query insert*, 116,7 detik untuk *query update* dan 29,0 detik untuk *query delete*.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan pada bab sebelumnya, yaitu Bab Uji Coba dan Evaluasi. Bab ini juga digunakan sebagai jawaban dari rumusan masalah yang dikemukakan pada Bab Pendahuluan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. *Data adapter* dapat digunakan sebagai sistem untuk mensinkronisasi antara basis data SQL dan basis data NoSQL. Dari hasil pengujian, proses sinkronisasi berhasil bekerja 100% sukses dan kedua basis data memiliki data yang sama. Dengan menggunakan pendekatan *query direct access*, aplikasi tetap dapat mengakses basis data saat proses sinkronisasi sedang berlangsung.
2. Berdasarkan hasil pengujian, untuk setiap peningkatan jumlah data atau *query*, waktu yang dibutuhkan untuk melakukan inisialisasi adalah linear, baik pada proses inisialisasi maupun transformasi. Jumlah penggunaan memori RAM tidak mengalami peningkatan yang signifikan berada pada rentang 1.736,6 MB sampai dengan 2.431,8 MB. Hal ini menunjukan jika sistem *data adapter* membutuhkan kapasitas RAM yang cukup besar. Untuk persentase penggunaan processor, sistem mengambil sumber daya pada rentang 14,186% sampai dengan 30,450%.
3. Basis data NoSQL, yaitu Apache HBase, memiliki kemampuan yang lebih baik jika dibandingkan dengan basis data relasional, yaitu MySQL, dalam hal melakukan *query*

analisis untuk pengambilan data beberapa tabel dengan jumlah baris yang banyak. Hal ini ditunjukan dengan waktu respon yang dibutuhkan untuk mengambil data pada antar muka nomor 8 dan 11 pada Gambar 5.4 menunjukan waktu respon HBase lebih cepat dibandingkan dengan MySQL, bahkan pada percobaan nomor 11, basis data MySQL tidak memberikan respon sampau melewati batas waktu. Hal ini menunjukan jika basis data NoSQL layak digunakan untuk menangani permasalahan *big data*.

6.2 Saran

Saran yang diberikan untuk pengembangan lebih lanjut terhadap sistem ini adalah sebagai berikut:

1. Mekanisme *data adapter* perlu dikembangkan lagi dengan melakukan perbandingan dengan menggunakan tipe basis data NoSQL yang lainnya.
2. Perlu dilakukan penelitian lebih lanjut untuk sinkronisasi basis data dengan jumlah node yang lebih banyak, serta bagaimana sistem mampu menangani kegagalan yang bisa terjadi pada server basis data atau server *data adapter*.

DAFTAR PUSTAKA

[1] G. Audin, "No Jitter," UBM, 2017. [Online]. Available: http://www.nojitter.com/post/240170228/the-network-impact-of-big-data. [Diakses 03 April 2017].

[2] Oracle, "Oracle," Oracle, [Online]. Available: https://www.oracle.com/mysql/index.html. [Diakses 13 April 2017].

[3] COUCHBASE, "COUCHBASE," COUCHBASE , 2017. [Online]. Available: https://www.couchbase.com/nosql-resources/why-nosql. [Diakses 13 April 2017].

[4] R. S. Samiddha Mukherjee, "Big Data – Concepts, Applications, Challenges," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 2, p. 66, 2016.

[5] S. Matteson, "TechRepublic," CBS Interactive., 2017. [Online]. Available: http://www.techrepublic.com/blog/big-data-analytics/big-data-basic-concepts-and-benefits-explained/. [Diakses 2017 Mei 10].

[6] J. Ellingwood, "Digital Ocean," DigitalOcean™ Inc., 28 September 2016. [Online]. Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-big-data-concepts-and-terminology. [Diakses 10 Mei 2017].

[7] SAS Institute Inc., "SAS US," SAS Institute Inc., 2017. [Online]. Available: https://www.sas.com/en_us/insights/big-data/what-is-big-data.html. [Diakses 12 Mei 2017].

[8] P. K. Pankaj Sareen, "NOSQL DATABASE AND ITS COMPARISON WITH SQL DATABASE," *International Journal of Computer Science & Communication Networks*, vol. 5, pp. 293-298.

[9] G. C. Everest, Database Management: Objectives, System Functions, and Administration, New York: McGraw-Hill Companies, 1986.

[10] P. ". D. D. Christensson, "Techterms.com," Sharpened Production , 16 July 2016. [Online]. Available: https://techterms.com/definition/relational_database.. [Diakses 28 March 2017].

[11] M. Rouse, "TechTarget.com," TechTarget, 10 2011. [Online]. Available: http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL. [Diakses 20 10 2016].

[12] mongoDB, "mongoDB," MongoDB, Inc., [Online]. Available: https://www.mongodb.com/nosql-explained. [Diakses 20 10 2016].

[13] B. M. Sasaki, "neo4j," Neo Technology, Inc., 2017. [Online]. Available: https://neo4j.com/blog/why-nosql-databases/. [Diakses 12 Mei 2017].

[14] J. Z. C.-H. L. S.-C. C. C.-H. H. C. M.-F. J. Y.-C. C. Ying-Ti Liao, "*Data adapter* for *query*ing and transformation between SQL and NoSQL database," *Elsevier,* vol. 65, pp. 111-121, 2016.

[15] MySQL, "MySQL," Oracle Corporation, [Online]. Available: https://www.mysql.com/about/. [Diakses 20 10 2016].

[16] P. Lutus, "arachnoid.com," 2012. [Online]. Available: https://arachnoid.com/MySQL/. [Diakses 28 Marh 2017].

[17] INTELLIPAAT.COM, "INTELLIPAAT.COM," 2017. [Online]. Available: https://intellipaat.com/tutorial/hbase-tutorial/introduction/. [Diakses 20 04 2017].

[18] H. Inc., "HORTONWORKS," Hortonworks Inc., 2016. [Online]. Available: http://hortonworks.com/apache/hbase/. [Diakses 20 11 2016].

[19] "Guru99," 2017. [Online]. Available: http://www.guru99.com/hbase-architecture-data-flow-usecases.html. [Diakses 03 March 2017].

[20] Apache HBase Team, "Apache HBase Reference Guide," The Apache Software Foundation, 11 April 2017. [Online]. Available: http://hbase.apache.org/book.html#_read_hbase_shell_commands_from_a_command_file. [Diakses 20 May 2017].

[21] "HORTONWORKS," Hortonworks Inc, [Online]. Available: https://hortonworks.com/hadoop-tutorial/bi-apache-phoenix-odbc/. [Diakses 6 April 2017].

[22] Apache Software Foundation, "Apache Phoenix," Apache Software Foundation, [Online]. Available: https://phoenix.apache.org/who_is_using.html. [Diakses 6 April 2017].

[23] S. Srungarapu, "Cloudera Engineering Blog," Cloudera, Inc, 6 Mei 2015. [Online]. Available: https://blog.cloudera.com/blog/2015/05/apache-phoenix-joins-cloudera-labs/. [Diakses 6 Maret 2017].

[24] T. Point, "Tutorialspoint.com," Tutorials Point (I) Pvt. Ltd., 2017. [Online]. Available: https://www.tutorialspoint.com/python/python_overview.htm. [Diakses 14 April 2017].

[25] Python, "Python.org," Python Software Foundation, 2017. [Online]. Available: https://www.python.org/. [Diakses 14 April 2017].

[26] K. Das, "Pym," pymbook, 2015. [Online]. Available: http://pymbook.readthedocs.io/en/latest/flask.html. [Diakses 2 Mei 2017].

[27] A. Ronacher, "Flask," Armin Ronacher and contributors, 2015. [Online]. Available: http://flask.pocoo.org/. [Diakses 2 Mei 2017].

[28] Contentshare, “OpenFlights.org,” Contentshare, 2017. [Online]. Available: https://openflights.org/data.html. [Diakses 20 04 2017].

LAMPIRAN A
KODE SUMBER

```
1  import paramiko
2  import string
3  import webbrowser
4  import time, os, re
5  from datetime import datetime
6  import MySQLdb
7  import csv, sys
8  import subprocess
9  import conv_phoenix
10 import c_db
11 from ConfigParser import SafeConfigParser
12
13 sync_time_init=0
14 allow_q = ["insert", "delete", "update"]
15
16 def getLastSync(data):
17      host = data['host']
18      username = data['username']
19      password = data['password']
20      db_name = data['db_name']
21
22      db = MySQLdb.connect(host=host, user=username, passwd=password, db=db_name)
```

```
        cursor = db.cursor()
23
24      sql = "SELECT * FROM log_sinkronisasi ORDER BY waktu DESC LIMIT 1"
25      try :
26          cursor.execute(sql)
27          row = cursor.fetchone()
28          get_date = row[1]
29          return get_date
30      except :
31          print "Error, unable to fetch Last Sync data"
32          return 0
33      db.close()
34
35  def dumpMysqlTableToCsv(db_data, tabel_name):
36      data = db_data
37      host = data['host']
38      username = data['username']
39      password = data['password']
40      db_name = data['db_name']
41      # dump dari MySQL ke csv
42      db = MySQLdb.connect(host=host, user=username, passwd=password, db=db_name)
43      cursor = db.cursor()
44      sql = "SELECT * FROM {0}".format(tabel_name)
45      file_output = "file/csv/fetch_all_{0}.csv".format(tabel_name)
46
47      print "sql: ", sql
48      print "file_output: ", file_output
```

```
49
50      try :
51          cursor.execute(sql)
52          result = cursor.fetchall()
53          c = csv.writer(open(file_output, "wb"))
54          for row in result:
55              c.writerow(row)
56          return 1
57      except :
58          print "Error, unable to fetch data from MySQL (",host,")"
59          return 0
60      finally:
61          db.close()
62
63  def initiateDBtoHBase(data_db):
64      get_table = c_db.getTableToSync()
65      list_table = get_table['table'].replace(" ", "").split(',')
66      for table in list_table:
67          d = dumpMysqlTableToCsv(data_db, table)
68          if (d) :
69              print "Berhasil, tabel ", table," berhasil di dump ke
    fetch_all_",table,".csv"
70      return list_table
71
72
73  def insertNewSync(sync_db, mysql_db, hbase_db):
74      # DB Log sinkronisasi
```

```
75      host = sync_db['host']
76      username = sync_db['username']
77      password = sync_db['password']
78      db_name = sync_db['db_name']
79
80      # DB MySQL
81      host_mysql = mysql_db['host']
82
83      # DB HBase
84      host_hbase = hbase_db['host']
85
86      ts = time.time()
87      timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
88      print 'time stamp: ', timestamp
89      db = MySQLdb.connect(host=host, user=username, passwd=password, db=db_name)
90      cursor = db.cursor()
91
92      sql = "INSERT INTO log_sinkronisasi (waktu, ip_src, ip_dst, status) VALUES
    ('{0}', '{1}', '{2}', '{3}')".format(timestamp,host_mysql,host_hbase,0)
93      print 'sql : ', sql
94
95      try :
96          cursor.execute(sql)
97          db.commit()
98          if cursor.lastrowid :
99              return cursor.lastrowid
100         else :
```

```
101                 return 0
102         except :
103             db.rollback()
104             print "Error, unable to record current sync time!"
105             return 0
106         finally :
107             db.close
108 
109 def updateStatusSinkronisasi(sync_db, id_log, status):
110         # DB Log sinkronisasi
111         host = sync_db['host']
112         username = sync_db['username']
113         password = sync_db['password']
114         db_name = sync_db['db_name']
115         db = MySQLdb.connect(host=host, user=username, passwd=password, db=db_name)
116         cursor = db.cursor()
            sql = "UPDATE log_sinkronisasi SET status='{0}' WHERE id_log={1}".format(status,
117 id_log)
118 
119         try :
120             cursor.execute(sql)
121             db.commit()
122             return 1
123         except :
124             db.rollback()
125             print "Error, unable to update log status!"
126             return 0
```

```
127      finally :
128          db.close
129
130  def isDateFormat(input):
131      try:
132          time.strptime(input, '%y%m%d')
133          return True
134      except ValueError:
135          return False
136
137  def isTimeFormat(input):
138      try:
139          time.strptime(input, '%H:%M:%S')
140          return True
141      except ValueError:
142          return False
143
144  def convertDateTimeFormat(input):
145      result = datetime.strptime(input, '%y%m%d %H:%M:%S')
146      return result
147
148  def remoteServerMysqlDb():
149      data = c_db.getSshAccess()
150      host = data['host']
151      username = data['username']
152      password = data['password']
153
```

```
154       ssh = paramiko.SSHClient()
155       ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
156       ssh.connect(host, username=username, password=password)
157       return ssh
158
159  def mainGetFile() :
160       ftp = ssh.open_sftp()
161       ftp.get('/var/log/mysql/log_query.log', 'file/get_log_query.log')
162       ftp.close()
163       # webbrowser.open_new(os.getcwd()+'/get_log_query.log')
164       print ("FILE GOT!")
165
166  def initiateTransformMysqlToHbase(mysql_db, hbase_db, sync_db):
167       print "Belum pernah sinkronisasi"
168       init = initiateDBtoHBase(mysql_db)
169
170       if (init) :
171            print "Data berhasil di export!"
172            print "Inisialisasi data awal.."
173            print "Mentransformasi data ke HBase..."
174            create_tabel = "psql.py file/flight_create_2.sql"
175
176            try :
177                 result_create_tabel = subprocess.check_output([create_tabel],
     shell=True)
178                 if (result_create_tabel):
179                      print "Tabel berhasil dibuat di HBase"
```

```
180                     for table in init:
181                          hbase_table = table.upper()
182                          phoenix_cmd = "psql.py -t {0}
    file/csv/fetch_all_{1}.csv".format(hbase_table, table)
183
184                          try :
185                               result_phoenix = subprocess.check_output([phoenix_cmd],
    shell=True)
186                               if (result_phoenix):
187                                    print "Tabel {0} berhasil diimport ke
    HBase...".format(table)
188                               else :
189                                    print "Tabel {0} gagal diimport ke
    HBase..".format(table)
190                          except Exception as e:
191                               print(e)
192                 else:
193                      print "Gagal membuat tabel di HBase.. "
194
195               except Exception as e:
196                    print(e)
197
198          update_status_sync = updateStatusSinkronisasi(sync_db, sync_time_init, 1)
199          if (update_status_sync) :
200               print 'Proses inisialisasi selesai..'
201
202     if __name__ == '__main__':
```

```
203     data_mysql = c_db.getConfMysqlDb()
204     data_hbase = c_db.getConfHbaseDb()
205     data_sync_db = c_db.getConfSyncLogDb()
206     data_ssh = c_db.getSshAccess()
207
208     ssh = paramiko.SSHClient()
209     ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
210     ssh.connect(data_ssh['host'], username=data_ssh['username'],
    password=data_ssh['password'])
211
212     try :
213         act = int(sys.argv[1])
214         if (act==1):
215             c = 1
216         elif (act==2):
217             c = 2
218         else :
219             c = -1
220             print "Parameter salah ..."
221             print ">> 1: get file log"
222             print ">> 2: Syncronize MySQL to HBase"
223     except :
224         print "Sintaks salah!"
225         c = -1
226
227     if c==1 :
228         mainGetFile()
```

```
229     if c==2 :
230         start_time = time.time()
231         last_sync = getLastSync(data_sync_db)
232         present = datetime.now()
233         print 'sinkron terakhir: ',last_sync
234         print 'waktu sekarang: ', present
235         sync_time_init = insertNewSync(data_sync_db, data_mysql, data_hbase)
236 
237         if (not last_sync) :
238             initiateTransformMysqlToHbase(data_mysql, data_hbase, data_sync_db)
239 
240         elif (last_sync) :
241             print "Sinkronisasi terakhir: ", last_sync
242             do_patching = False
243 
244             # Mengambil data DELETE dari remote server, kemudian di simpan di file
    local
245             print "Mengambil data Query dari log ..."
246             ftp = ssh.open_sftp()
247             file = ftp.file('/var/log/mysql/log_query.log', 'r')
248             f_list_query = open('file/list_all_query.sql', 'w')
249             date_tmp = ''
250 
251             count_query = 0
252             print "Cek aksi INSERT, UPDATE, DELETE pada log..."
253             for line in file :
254                 query_line = line
```

```
255                     get_waktu = query_line.split()
256                     tgl = str(get_waktu[0])
257                     if (len(get_waktu)>1):
258                          wkt = str(get_waktu[1])
259                     if (isDateFormat(tgl)) :
260                          if (isTimeFormat(wkt)) :
261                               date_time_join = tgl + ' ' + wkt
262                               date_tmp = convertDateTimeFormat(date_time_join)
263                     if (date_tmp > last_sync) :
264                          if 'Query' in line :
265                               for a_q in allow_q :
266                                    q = re.split(r'\t+', query_line)
267                                    if (a_q in q[2]) or (a_q.upper() in q[2]) :
268                                         query = q[2].replace("\n", "")
269                                         split_query = query.split()
270
271                                         count_query+=1
272                                         # Cek jika sintaks adalah UPDATE
273                                         if (split_query[0].upper()=="INSERT"):
274                                              print "[sync] INSERT ", split_query[2]
275                                              new_query = query.replace("INSERT",
    "UPSERT").replace("insert", "UPSERT") + ";\n";
276                                         elif (split_query[0].upper()=="UPDATE"):
277                                              print "[sync] UPDATE ", split_query[1]
278                                              new_query =
    conv_phoenix.update_to_upsert(query)
279                                              if (new_query==0) :
```

```
280                                                  print "[err] Table not int List! "
281                                         elif (split_query[0].upper()=="DELETE") :
282                                              print "[sync] DELETE ", split_query[1]
283                                              new_query = query.replace("INSERT",
    "UPSERT").replace("insert", "UPSERT") + ";\n";
284
285                                         if (new_query!=0):
286                                              f_list_query.write(new_query)
287                                         do_patching = True
288                    if (count_query ==0) :
289                         print "Tidak ada data yang transformasikan!"
290                         updateStatusSinkronisasi(data_sync_db, sync_time_init, 1)
291
292                    f_list_query.close()
293                    ftp.close()
294
295                    # Menjalankan DELETE, INSERT, dan UPDATE
296                    if (do_patching):
297                         print "Sinkronisasi, DELETE, INSERT dan UPDATE file on progress
    sync to HBase..."
298                         phoenix_cmd = "psql.py file/list_all_query.sql"
299                         res_phoenix =''
300
301                         try :
302                              res_phoenix = subprocess.check_output([phoenix_cmd],
    shell=True)
303                         except Exception as e:
```

```
304                         print(e)
305
306                     if (res_phoenix) :
307                         insert_time = updateStatusSinkronisasi(data_sync_db,
    sync_time_init, 1)
308                         if (insert_time) :
309                             print 'Update status sinkronisasi berhasil'
310                         print 'Sync berhasil dilakukan.'
311
312             duration = time.time() - start_time
313
314             print "Durasi sinkronisai : %s seconds" % (duration)
315             print "Done"
316         if c==3 :
317             print "Key terlarang!"
318             getconf = c_db.getConfMysqlDb()
319             print getconf
320         else :
321             print "EXIT!";
322         ssh.close()
```

Kode Sumber A. 1 Sinkronisasi

```
1  import re
2  from ConfigParser import SafeConfigParser
3
4  def getPrimaryKey():
```

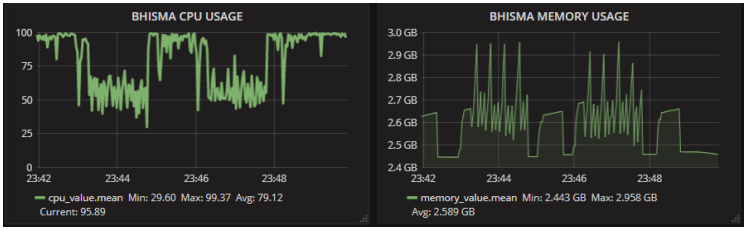
```
5       parser = SafeConfigParser()
6       parser.read('configuration.ini')
7       conf_list = {
8            'table' : parser.get('mysql_table', 'table'),
9            'primary_key' : parser.get('mysql_table', 'primary_key')
10      }
11      return conf_list
12
13  def update_to_upsert(val) :
14      update_str = val
15      update_split = update_str.split()
16      tabel = update_split[1]
17      kolom = [None]*50
18      value = [None]*50
19
20      # Get nama table
21      table_name = re.findall(r'UPDATE(.*?)SET', update_str)[0].replace(" ", "")
22      # Mencari Primary Key Table
23      get_list_tabel = getPrimaryKey()
24      list_table = get_list_tabel['table'].split(',')
25      list_pk = get_list_tabel['primary_key'].split(',')
26      if (table_name in list_table):
27           pos = list_table.index(table_name)
28           pk = list_pk[pos]
29           print "PK: ", pk
30      else:
31           return 0
```

```
32
33      # get kolom and value betwen SET and WHERE
34      fields = re.findall(r'SET(.*?)WHERE', update_str)
35      kol_val = fields[0].split(',')
36      list_kolom = "({0},".format(pk)
37      list_value = " SELECT {0},".format(pk)
38      for i in range(len(kol_val)):
39           tmp = kol_val[i]
40           tmp_split = tmp.split('=')
41           kolom[i] = tmp_split[0]
42           value[i] = tmp_split[1]
43           if (i==(len(kol_val)-1)):
44                list_kolom += kolom[i] + ")"
45                list_value += value[i]
46           else :
47                list_kolom += kolom[i] + ","
48                list_value += value[i] + ","
49
50      last_pos = len(kol_val)
51      split_where = update_str.split('WHERE')[1]
52
53      final_string = "UPSERT INTO {0} {1} {2} FROM {0} WHERE {3};\n".format(table_name,
list_kolom, list_value, split_where)
54      return final_string
```

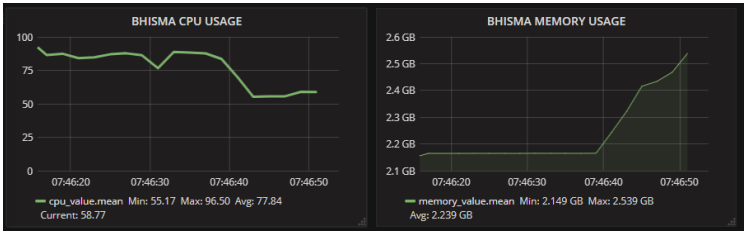
Kode Sumber A. 2 Converter Sintaks MySQL ke Sintaks Phoenix

LAMPIRAN B
GAMBAR PENDUKUNG

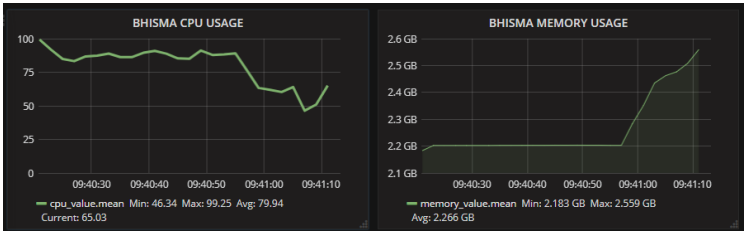
Gambar B. 1 Contoh Tangkapan Layar Uji Coba Performa Perbandingan Jumlah Baris



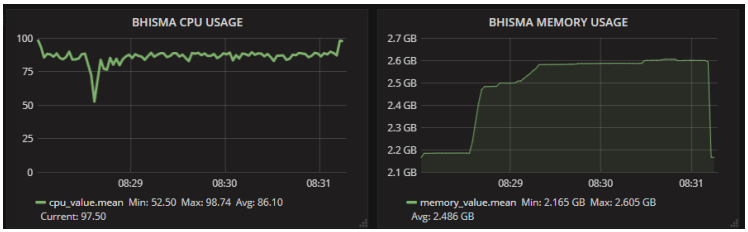
Gambar B. 2 Contoh Tangkapan Layar Uji Coba Performa Perbandingan Jumlah Tabel



Gambar B. 3 Contoh Tangkapan Layar Uji Coba Performa Transformasi Query Insert



Gambar B. 4 Contoh Tangkapan Layar Uji Coba Performa Transformasi Query Delete



Gambar B. 5 Contoh Tangkapan Layar Uji Coba Performa Transformasi Query Update

BIODATA PENULIS

I Gusti Ngurah Adi Wicaksana, lahir pada 29 September 1995 di Denpasar. Penulis menempuh pendidikan mulai dari SDN 1 Ubud (2001-2007), SMPN 1 Ubud (2007-2010), SMAN 3 Denpasar (2010-2013), dan S1 Teknik Informatika ITS (2013-2017). Memiliki beberapa hobi antara lain mendengarkan dan bermain musik serta bermain futsal. Pernah menjadi asisten dosen di mata kulian Sistem Digital, Kecerdasan Buatan, Kecerdasan Komputasional dan Keamanan Informasi dan Jaringan Komputer serta menjadi asisten praktikum pada mata kuliah Sistem Operasi dan Jaringan Komputer. Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain Staff Departemen Media Informasi Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ke-2 dan Staff Ahli Departemen Media Informasi Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ke-3. Penulis juga aktif dalam kegiatan kepanitian Schematics 2014 dengan menjadi staff divisi Web dan 3D (Desain, Dokumentasi dan Dekorasi) dan koordinator 3D (Desain, Dokumentasi dan Dekorasi) pada Shcematics 2015. Penulis juga merupakan administrator di Laboratorium Arsitektur dan Jaringan Komputer.

Kritik dan saran sangat diharapkan guna peningkatan kualitas dan penulisan selanjutnya. Untuk itu, silahkan kirim kritik dan saran ke : adiwicaksana29@gmail.com.