W

;/--n

**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**

**SYNCRONIZATION 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 Komputasi Berbasis Jaringan

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)

# Henning Titi Ciptaningtyas, S.Kom., M.Kom .....................

NIP: 19840708 201012 2 004 (Pembimbing 2)

**SURABAYA**

**MEI, 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 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.*

*Data dari genom sendiri sangatlah besar sehingga dibutuhkan suatu teknik yang dapat mengurangi waktu komputasi. Hal tersebut dapat tercapai dengan cara mengimplementasikan komputasi paralel terhadap algoritma LCS. Oleh karena itu Tugas Akhir ini mengimplementasikan Algoritma LCS yang diparalelkan.*

*Tugas Akhir ini memiliki tujuan untuk mengimplementasikan mekanisme data adapter dalam sinkronisasi basis data RDB dan NoSQL. Berdasarkan hasil uji coba, waktu eksekusi LCS secara paralel terbukti lebih cepat daripada sekuensial dan panjang karakter yang dapat diproses pada komputasi paralel lebih banyak daripada sekuensial.*

***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

*The process of finding the global alignment of two genomes by comparing the sequence is one of the problems frequently encountered in computational biology. One of the ways to solve this problem is to implement the algorithm Longest common subsequence (LCS) in finding the global alignment of both genomes.*

*Data from the genome itself is large so it takes a technique that can reduce the computation time. This can be achieved by implementing parallel computation of the LCS algorithm. Therefore, this undergraduate thesis implements the parallelization of LCS algorithm.*

*The purpose of this undergraduate thesis is to find the similarity of the two genomes and analyze and compare the performance of the sequential of LCS algorithms and the parallelization of LCS algorithm. LCS algorithm implemented in parallel using OpenMPI library. Based on trial results, the execution time LCS in parallel proved faster than sequential and length of characters that can be processed in parallel computing is longer than sequential.*

***Keywords: LCS, OpenMPI, Dynamic Programming, Genom.***

***[Halaman ini sengaja dikosongkan]***

# KATA PENGANTAR

Alhamdulillahirabbil’alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul ***“*RANCANG BANGUN KOMPUTASI PARALEL PADA ALGORITMA *LONGEST COMMON SUBSEQUENCE* UNTUK PERBANDINGAN GENOM MENGGUNAKAN *FRAMEWORK* OPENMPI*”*.** Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Keluarga di Pontianak khususnya Mama, Bapak, dan Ndut yang telah memberikan dukungan moral dan material serta do’a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Keluarga di Surabaya, Tante Vitri, Om Dodo, Mba Dita, Mas Dimas, Ivan dan Iyuk yang telah memberikan kehangatan keluarga serta support untuk penulis selama kuliah di ITS.
4. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
5. Bapak Hudan Studiawan, S.Kom., M.Kom**.** selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
6. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS.
7. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
8. Teman-teman semangat TA Altea, Angga, Bimo, Dian, Fatih, Fifi, Lubna, Luqman, Oshi, Pinas, dan Puni.
9. Teman-teman administrator laboratorium Arsitektur dan Jaringan Komputer(AJK), Mas Sam, Mas Uyung, Mbak Vivi, Kak Harum, Mas Romen, Mas Dimas, Agus, Pur, Thiar, Uul, Wicak, Zaza, Risma, Nindy, Asbun, Daniel, Fatih, Oink, dan Syukron.
10. Teman-teman angkatan 2012 yang yang telah berbagi ilmu, dan memberi motivasi kepada penulis.
11. 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 2016

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# DAFTAR KODE SUMBER

***[Halaman ini sengaja dikosongkan]***

# DAFTAR PSEUDOCODE

***[Halaman ini sengaja dikosongkan]***

# BAB I PENDAHULUAN

## Latar Belakang

*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] 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 mengkueri data tidak menggunakan sintaks yang sama seperti mengambil data pada basisdata SQL karena sintaks *query* yang digunakan berbeda. Dibandingkan merubah kode sumber atau merubah basis data RBD menjadi NoSQL, penulis melakukan riset untuk mengintegrasikan kedua basis data. Aplikasi terkoneksi dengan RDB 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 RDB.

Integrasi basis data dapat mempengaruhi desain sistem yang asli. Dalam sistem, aplikasi memperoleh data dari basis data relasional menggunakan kueri 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 RDB 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.

## Rumusan Masalah

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

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

## Batasan Masalah

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

* 1. Banyak node server untuk basis data yang digunakan adalah berjumlah dua komputer.

1. Mekanisme yang digunakan adalah menggunakan konsep data adapter
2. Basis data yang digunakan adalah MySQL sebagai basis data RDB dan Apache HBase sebagai basis data NoSQL.

## Tujuan

Tujuan Tugas Akhir ini adalah mengimplementasikan mekanisme data adapter dalam sinkronisasi basis data RDB dan NoSQL.

## Manfaat

Dengan dibuatnya Tugas Akhir ini yaitu dapat mensinkronisasikan basis data RDB dengan NoSQL dengan mekanisme data adapter.

## 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.

1. 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.

1. Analisis dan perancangan perangkat lunak

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*.

1. 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.

1. 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.

1. Penyusunan buku Tugas Akhir

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

## Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. 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 Analisis dan Perancangan Perangkat Lunak**

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 perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari perangkat lunak yang telah dibuat sesuai dengan data yang diujikan.

**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.

# 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.

## 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 class, atribut, dan relasi antar basis data. Data adalah fakta. Informasi menggambarkan data. Informasi adalah data dengan konteks yang berarti

## Relational Database

*Relational database*, atau disingkat RDB, menggambarkan suatu kumpulan dari banyak relasi. Konsep RDB 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 untuk.

Sebagian besar database yang digunakan pada aplikasi modern saat ini adalah relasional database. Relational databases 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 “atribut”. 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.

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 field baris pada tabel lainnya. Nilai yang disisipkan ini tabel lain ini disebut dengan *foreign key*

Cara yang digunakan untuk mengakses data pada database relasional adalah dengan menggunakan kueri SQL (*Structured Query Language).* Kueri SQL dapat digunakan untuk membuat, memodifikasi, dan menghapus tabel, serta memilih, insert, dan menghapus data dari tabel yang ada [4].

## NoSQL Database

Database NoSQL, atau *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.

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 dua sampai delapan belas bulan. Dijaman sekarang, tim bekerja dengan secepat-cepatnya, mengiterasi dan mem-push code 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 globbal 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.

## Data Adapater

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

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 kueri untuk menangani kueri NoSQL.

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

Fitur terakhir adalah *Query Approach* atau pendekatan kueri. Pendekatan kueri yang digunakan adalah *direct access (DA mode)*. Pendekatan ini memperbolehkan aplikasi untuk melakukan kueri secara langsung ke basis data tanpa harus menunggu proses transformasi selesai dilakukan.

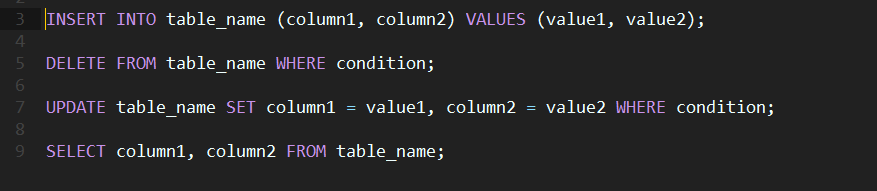
## MySQL

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. [5]

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 kolum, 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. [5]

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). [6] 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 2.1. Tabel 2. 1 adalah contoh gambaran sederhana bentuk tabel pada basis data MySQL.

Pada penilitian 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.



Gambar 2.1 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 |

## Apache HBase

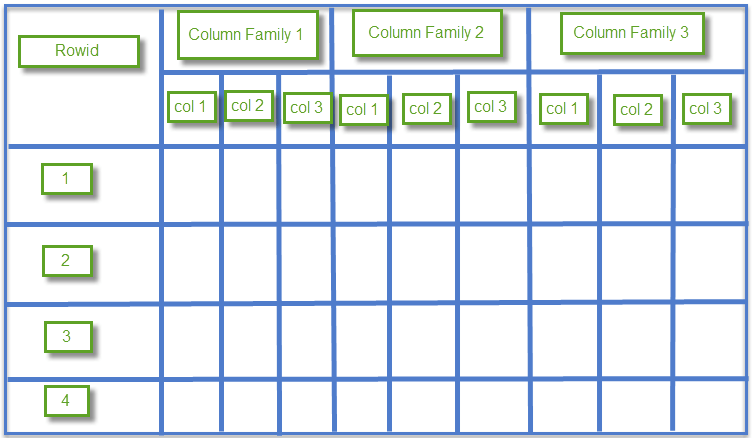
Apache Hbase dibuat pada tahun 2007 di Powerset dan awalnya merupakan bagian dari Hadoop. Sejak saat itu, proyek ini menjadi proyek tingkat atas dibawah Apache Software Foundation. Hbase tersedia dibawah lisensi Apache Software License, versi 2.0. [7]

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.

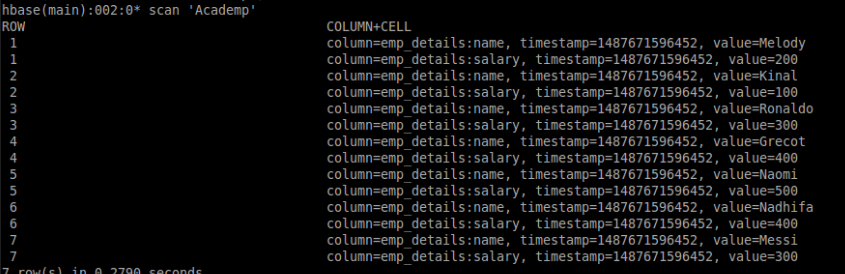
Apache HBase memberikan beberapa fitur seperti secara acak dan *real-time* mengakses data pada Hadoop. HBase dibuat untuk menampung tabel yang berukuran sangat besar, menjadikannya pilihan yag tepat untuk menyimpan *multi-structure* atau data yang jarang. Pengguna dapat melakukan query HBase dalam waktu 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. [8]

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. [9] Skema penyimpan di Hbase dapat dilihat pada Gambar 2.1.

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 2.2



Gambar 2.2 Skema penyimpanan pada Apache Hbase



Gambar 2.3 Contoh tampilan data yang disimpan dalam Hbase jika diakses melalui terminal

## 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. [10] Beberapa perusahaan besar telah menggunakan Apache Phoenix seperti salesforce.com, Bloomberg, Sogou, Hortonworks, Alibaba.com, CertusNet, ebay.com, NGDATA, eHarmony dan PubMatic. [11]

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 kueri 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 kueri 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 kueri data kecil, atau beberapa detik untuk 10 juta baris. [12]

Dalam pengerjaan tugas akhir ini, Apache Phoenix digunakan pada data adapter sebagai data converter untuk merubah data dari MySQL ke HBase. Selain itu, Apache Phoenix juga diunakan sebagai digunakan untuk transformasi data dari HBase yang mana digunakan oleh aplikasi untuk menjalankan sintaks SQL untuk mengambil data melalui data adapter.

## 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). [13]

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. Banyak programmer yang suka dengan bahasa Python karena dapat meningkatkan produktivitas seperti tidak perlu adanya langkah kompilasi, dan siklus edit-tes-debug menjadi sangat cepat. [14]

## 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. [15]

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 2.3. [16]

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.



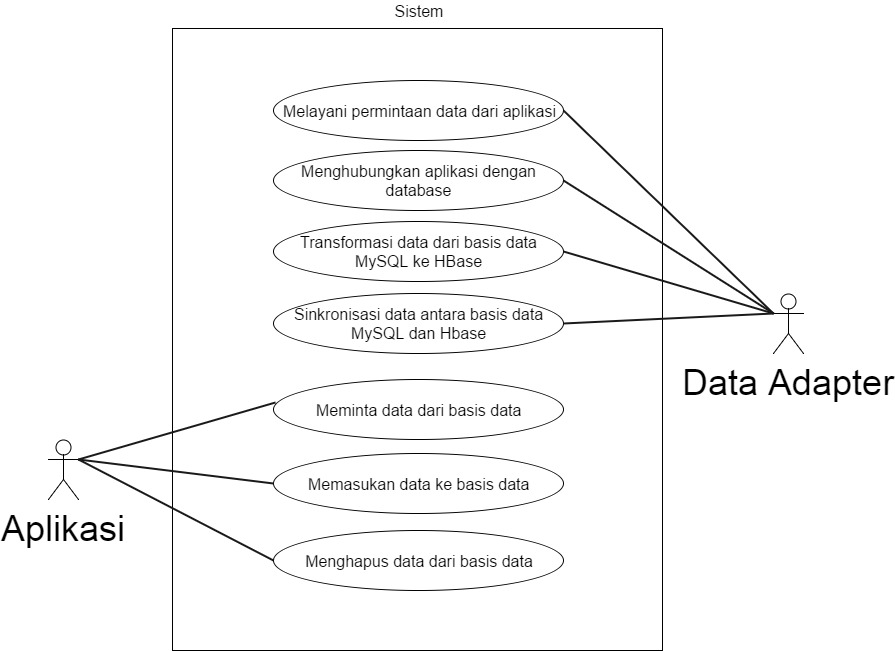
Gambar 2.4 Contoh Penggunaan Flask

# BAB III ANALISIS DAN PERANCANGAN PERANGKAT LUNAK

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.

## Kasus Pengguna

Terdapat dua aktor dalam diagram kasus pengguna yaitu aplikasi dan data adapter. Pada sistem, aplikasi memiliki tiga aktifitas dan data adapter memiliki 4 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

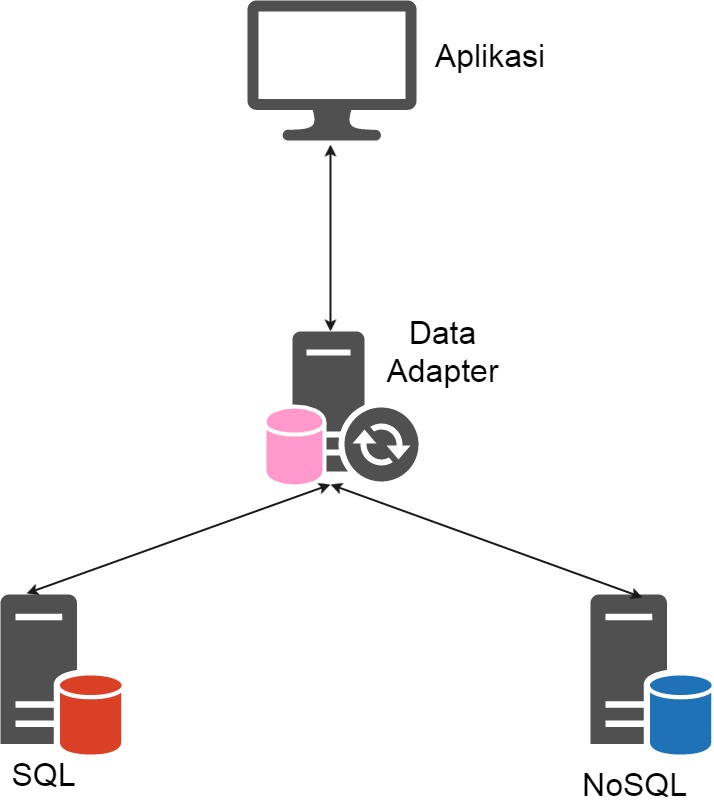
|  |  |  |  |
| --- | --- | --- | --- |
| **Kode** | **Nama Kasus Penggunaan** | **Aktor** | **Deskripsi** |
| UC01 | Melayani permintaan dari aplikasi | Data Adapter | Melayani permintaan data dari aplikasi, kemudian data adapter akan meneruskannya ke basis data |
| UC02 | Menghubungkan aplikasi dengan 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 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. |

## Arsitektur Sistem

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

### 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.

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.

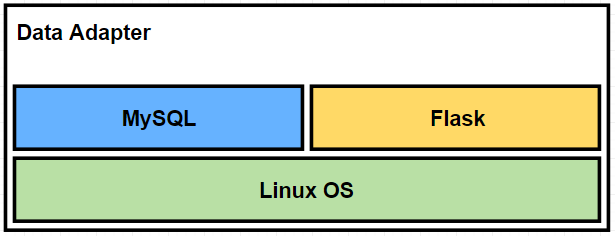
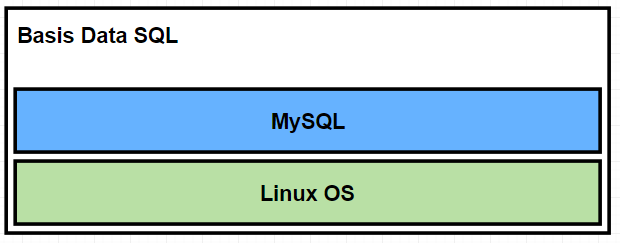
Gambar 3.2 Desain Sistem Data Adapter Secara Umum

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. 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.

### 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

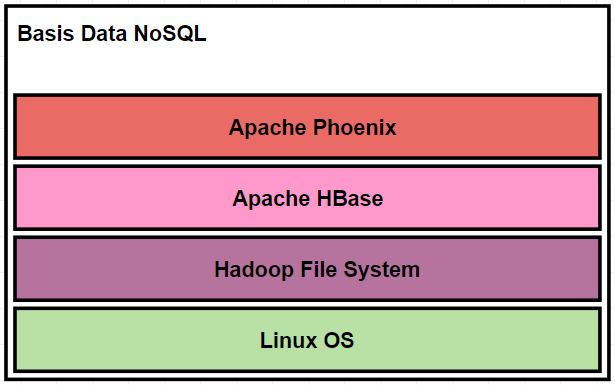
Data adapter terdiri dari DB Adapter dan DB Converter. DB Adapter berfungsi untuk menerima semua permintaan dari aplikasi seperti pengajuan kueri, pengambilan data, pembaruan data, dan penghapusan data. Untuk dapat melakukan hal ini, data adapter menyediakan antar muka dalam bentuk API. API menghasilkan keluaran data denga tipe data JSON. Semua komunikasi data yang dilakukan oleh aplikasi adalah melalui antar muka ini. Flask digunakan untuk membuat fitur antarmuka ini.

Sementara itu, 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 di HBase. Proses transformasi data ini dilakukan dengan bantuan Apache Phoenix. Apache Phoenix memungkinkan untuk menerjemahkan kueri SQL, mengkompilasinya menjadi beberapa rangkaian perintah HBase, kemudian mengeksekusinya. Python menjadi bahasa pemrograman yang di implementasikan pada data adapter. Gambar 3.3 menampilkan diagram arsitektur data adapter.

Gambar 3.3 Diagram Arsitektur Server Basis Data SQL

Gambar 3.4 Diagram Arsitektur Data Adapter

### Desain Basis Data SQL

Basis data SQL digunakan untuk menyimpan semua data aplikasi. Dalam perancangan dan pengembangan suatu aplikasi menuntut kemampuan untuk dapat diakses oleh banyak pengguna. Banyak pengembang yang memisahkan server aplikasi dengan server basis data. Pada tugas akhir ini, satu server yang terpisah dari server aplikasi digunakan sebagai basis data SQL. Aplikasi yang digunakan adalah MySQL. 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 3.5 Diagram Arsitektur Basis Data NoSQL

### 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 jumlahnya. Aplikasi yang digunakan sebagai basis data NoSQL adalah Apache HBase. Untuk dapat menggunakan Apache HBase, sebelumnya 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 translator kueri SQL. Desain arsitektur Basis Data NoSQL dapat dilihat pada

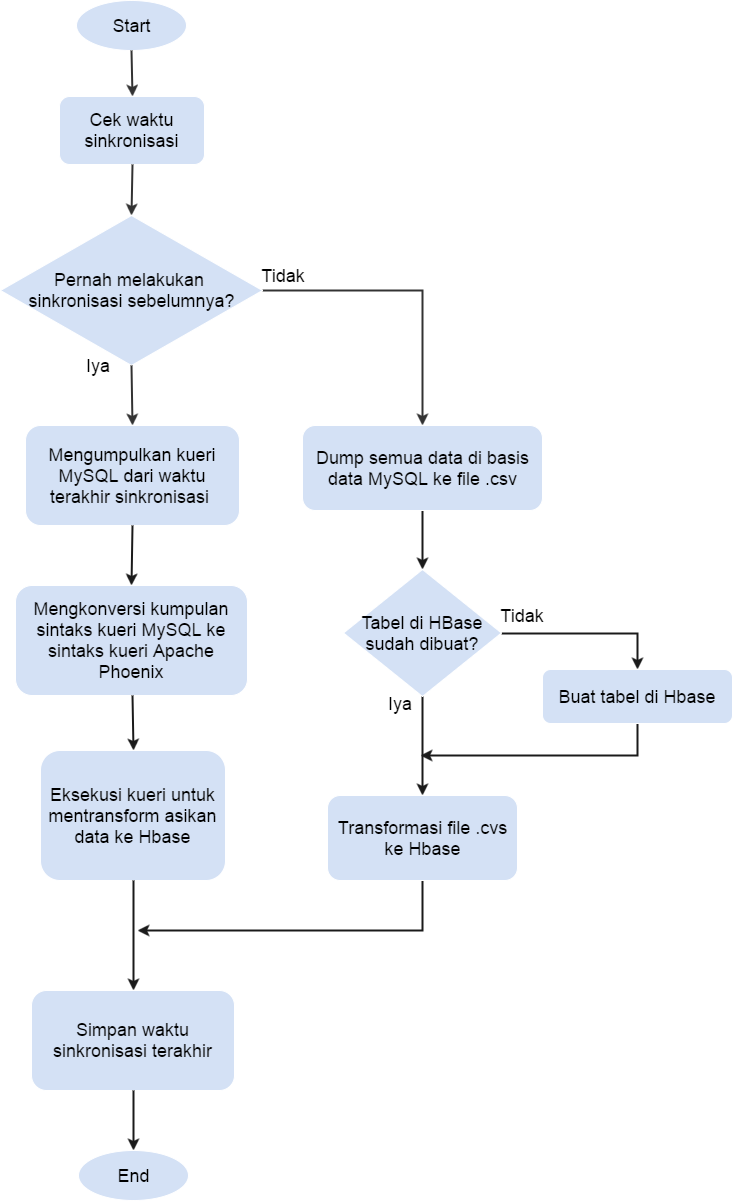
## Proses Transformasi dan Sinkronisasi

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 belum. Jika belum maka proses yang dilakukan adalah proses inisialisasi.

Proses inisialisasi adalah proses menyalin semua data di basis data SQL ke basis data NoSQL. Proses ini diawali dengan menge-dump atau mengekspor semua data pada tabel di MySQL menjadi file .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.

Jika pernah melakukan sinkronisasi sebelumnya, maka inisialisasi tidak akan dilakukan. Proses yang akan dilakukan adalah sistem akan mengumpulkan semua kueri dari log pada basis data MySQL. Kueri yang diambil adalah kueri-kueri yang melakukan perubahan pada basis data yaitu perintah *insert*, *update* dan *delete*. Semua log kemudian disimpan didalam satu file .sql. Dengan bantuan Apache phoenix, semua kueri pada berkas akan ditranslasikan ke HBase satu satu, 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. Diagram alur proses sinkronisasi dapat dilihat pada Gambar 3.6.



Gambar 3.6 Diagram Alur Proses Sinkronisasi

# BAB IV IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

## Lingkungan Implementasi

## Pemasangan g++ dan OpenMPI

# 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.

## Lingkungan Uji Coba

***[Halaman ini sengaja dikosongkan]***

# 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.

## Kesimpulan

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

1. .

## Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. .

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | N. F. Almeida Jr, E. N. Caceres, C. E. R. Alves dan S. W. Song, “Comparison of Genomes using High-Performance Parallel Computing,” dalam *Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing*, 2003. |
| [2] | A. Levitin, “Dynamic Programming,” dalam *Introduction to The Design and Analysis of Algorithms 3rd Edition*, United States of America, Pearson Education, Inc., 2012, p. 311. |
| [3] | D. Abraham dan N. S. Raj, “Approximate String Matching Algorithm for Phishing Detection,” dalam *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, New Delhi, 2014. |
| [4] | M. Elhadi dan A. Al-Tobi, “Duplicate Detection in Documents and WebPages using Improved Longest Common Subsequence and Documents Syntactical Structures,” dalam *Fourth International Conference on Computer Sciences and Convergence Information Technology*, Seoul, 2009. |
| [5] | A. Grama, A. Gupta, G. Karypis dan V. Kumar, Introduction to Parallel Computing, Second Edition, Addison Wesley, 2003, p. 534. |
| [6] | C. E. R. Alves, E. N. Cáceres, F. Dehne dan S. W. Song, “A Parallel Wavefront Algorithm for Efficient,” dalam *International Conference on Computational Science and Applications 2003*, Canada, 2003. |
| [7] | “Genetics Home Reference,” National Library of Medicine (US), 14 June 2016. [Online]. Available: https://ghr.nlm.nih.gov/primer/hgp/genome. [Diakses 16 June 2016]. |
| [8] | “What is DNA?,” National Library Medicine (US), 14 June 2016. [Online]. Available: http://ghr.nlm.nih.gov/handbook/basics/dna. [Diakses 15 June 2016]. |
| [9] | “The Science Behind the Human Genome Project: Understanding the Basics,” U.S. Department of Energy Human Genome Project, [Online]. Available: http://web.ornl.gov/sci/techresources/Human\_Genome/project/info.shtml. [Diakses 15 June 2015]. |
| [10] | “What is a gene?,” National Library of Medicine (US), 14 June 2016. [Online]. Available: https://ghr.nlm.nih.gov/primer/basics/gene. [Diakses 15 June 2016]. |
| [11] | “What is Heredity?,” Genetic Science Learning Center, 22 June 2014. [Online]. Available: http://learn.genetics.utah.edu/content/inheritance/intro/. [Diakses 15 June 2016]. |
| [12] | “NCBI National Center for Biotechnology Information,” [Online]. Available: http://www.ncbi.nlm.nih.gov/. [Diakses 21 June 2015]. |
| [13] | “Open MPI: Open Source High Performance Computing,” [Online]. Available: http://www.open-mpi.org/. [Diakses 21 June 2015]. |
| [14] | “MPI: A Message-Passing Interface Standard,” University of Tennessee, Knoxville, 2015. |
| [15] | GeeksforGeeks, “Printing Longest Common Subsequence,” [Online]. Available: http://www.geeksforgeeks.org/printing-longest-common-subsequence/. [Diakses 11 May 2016]. |

**LAMPIRAN A**

# KODE SUMBER

Pada lampiran berikut dijelaskan mengenai kode sumber yang digunakan. Kode sumber menggunakan Bahasa Pemrograman C++. Berikut kode sumber yang dicantumkan:

1. Kode Sumber 1 Fungsi Perhitungan LCS Paralel merupakan kode sumber dari prosedur perhitungan *LCS* paralel.
2. Kode Sumber 2 Fungsi Mencetak LCS merupakan kode sumber dari prosedur mencetak karakter *LCS* dari banyak proses.
3. Kode Sumber 3 Pemilihan Indeks Perhitungan merupakan potongan kode sumber bagian dari Kode Sumber 1 untuk memilih indeks dalam perhitungan *LCS* agar tidak terjadi perulangan perhitungan.
4. Kode Sumber 4 Pemilihan Indeks Kolom Untuk Menerima Data merupakan potongan kode sumber bagian dari Kode Sumber 1 untuk memilih kolom dalam penerimaan data ketika *proses* > 0.
5. Kode Sumber 5 Pengambilkan Karakter LCS merupakan potongan kode sumber bagian dari Kode Sumber 2 dalam mengambil karakter *LCS*.
6. Kode Sumber 6 Perhitungan Nilai Indeks LCS merupakan potongan kode sumber bagian dari Kode Sumber 1 dalam menghitung nilai indeks *LCS*.

Kode Sumber 1 Fungsi Perhitungan LCS Paralel

|  |
| --- |
| 1. **for** (k = 1; k <= numberOfProcess / alpa; k++) { 2. **if** (process == 0) { 3. **int** rowStart = ((alpa \* (k - 1) \* lengthOfX) / numberOfProcess); 4. **int** rowEndPrevProcess = ((alpa \* (k - 2) \* lengthOfX) / numberOfProcess); 5. **int** rowEnd = (alpa \* k \* lengthOfX) / numberOfProcess; 7. **if** (rowStart != 0) rowStart = rowStart + 1; 9. **if** (k != 1 && rowEnd == 0 && rowStart == 0) { 10. rowStart = rowStart + 1; 11. } 12. **if** (k != 1 && rowEndPrevProcess == 0 && rowStart == 0) { 13. rowStart = rowStart + 1; 14. } 16. **int** coloumnStart = (process) \* lengthOfY / numberOfProcess; 17. **if** (coloumnStart != 0) coloumnStart = coloumnStart + 1; 19. **int** coloumnEnd = (process + 1) \* lengthOfY / numberOfProcess; 20. **for** (rowCalculate = rowStart; rowCalculate <= rowEnd; rowCalculate++) { 21. numberOf*LCS*Character[rowCalculate].resize(sizeForY); 22. **for** (coloumnCalculate = coloumnStart; coloumnCalculate <= coloumnEnd; coloumnCalculate++) { 24. **if** (rowCalculate == 0 || coloumnCalculate == 0) { 25. numberOf*LCS*Character[rowCalculate][coloumnCalculate] = 0; 26. } **else** **if** (X[rowCalculate - 1] == Y[coloumnCalculate - 1]) { 27. numberOf*LCS*Character[rowCalculate][coloumnCalculate] =  numberOf*LCS*Character[rowCalculate - 1][coloumnCalculate - 1] + 1; 28. } **else** { 29. numberOf*LCS*Character[rowCalculate][coloumnCalculate] =  max(numberOf*LCS*Character[rowCalculate -1][coloumnCalculate],  numberOf*LCS*Character[rowCalculate][coloumnCalculate - 1]); 30. } 31. } 33. } 34. dest = process + 1; 35. **int** rowSendProcess0, coloumnSendProcess0; 36. **for** (rowSendProcess0 = rowStart; rowSendProcess0 <= rowEnd; rowSendProcess0++) { 37. coloumnSendProcess0 = (dest \* lengthOfY / numberOfProcess); 39. MPI\_Send( & numberOf*LCS*Character[rowSendProcess0][coloumnSendProcess0], 1, MPI\_INT, dest,  0, MPI\_COMM\_WORLD); 40. } 42. } **else** **if** (process > 0) { 43. **int** finish = 0; 44. **int** source = process - 1; 45. dest = process + 1; 46. **int** coloumnReceive, rowReceive, coloumnSend, rowSend; 47. **int** finishReceived = 0; 49. **int** firstColoumn = ((process) \* lengthOfY / numberOfProcess); 50. **int** rowStart = ((alpa \* (k - 1) \* lengthOfX) / numberOfProcess); 51. **int** rowEndPrevProcess = ((alpa \* (k - 2) \* lengthOfX) / numberOfProcess); 53. **int** coloumnStart = (process) \* lengthOfY / numberOfProcess; 54. **int** coloumnEnd = (process + 1) \* lengthOfY / numberOfProcess; 56. **int** rowEnd = (alpa \* k \* lengthOfX) / numberOfProcess; 57. **int** coloumnPreviousProcess = (process - 1) \* lengthOfY / numberOfProcess; 58. **if** (rowStart != 0) { 59. rowStart = rowStart + 1; 60. } 61. **if** (k != 1 && rowEnd == 0 && rowStart == 0) rowStart = rowStart + 1; 62. **if** (k != 1 && rowStart == 0 && rowEndPrevProcess == 0) rowStart = rowStart + 1; 64. **if** (coloumnStart != 0) { 65. coloumnStart = coloumnStart + 1; 66. } 68. **int** colStartForProcess1; 69. colStartForProcess1 = (1 \* lengthOfY / numberOfProcess); 70. **if** ((process \* lengthOfY / numberOfProcess) != colStartForProcess1) { 71. coloumnReceive = firstColoumn - coloumnPreviousProcess; 72. } **else** { 73. coloumnReceive = coloumnStart - 1; 74. } 76. **for** (rowReceive = rowStart; rowReceive <= rowEnd; rowReceive++) { 77. numberOf*LCS*Character[rowReceive].resize(sizeForY); 78. MPI\_Recv( & numberOf*LCS*Character[rowReceive][coloumnReceive], 1, MPI\_INT, source, 0,  MPI\_COMM\_WORLD, & status); 79. numberOf*LCS*Character[rowReceive][coloumnReceive - coloumnReceive] =  numberOf*LCS*Character[rowReceive][coloumnReceive]; 80. finishReceived = 1; 81. } 82. **if** (finishReceived == 1) { 83. **for** (rowCalculate = rowStart; rowCalculate <= rowEnd; rowCalculate++) { 84. numberOf*LCS*Character[rowCalculate].resize(sizeForY); 85. **for** (coloumnCalculate = coloumnStart; coloumnCalculate <= coloumnEnd;  coloumnCalculate++) { 86. **if** (rowCalculate == 0 || coloumnCalculate == 0) { 87. numberOf*LCS*Character[rowCalculate][coloumnCalculate - firstColoumn] = 0; 88. } **else** **if** (X[rowCalculate - 1] == Y[coloumnCalculate - 1]) { 89. numberOf*LCS*Character[rowCalculate][coloumnCalculate - firstColoumn] =  numberOf*LCS*Character[rowCalculate - 1][coloumnCalculate - 1 - firstColoumn] + 1; 90. } **else** { 91. numberOf*LCS*Character[rowCalculate][coloumnCalculate - firstColoumn] =  max(numberOf*LCS*Character[rowCalculate - 1][coloumnCalculate - firstColoumn],  numberOf*LCS*Character[rowCalculate][coloumnCalculate - 1 - firstColoumn]); 92. } 93. rowIndex = rowCalculate; 94. coloumnIndex = coloumnCalculate - firstColoumn; 95. } 97. }; 98. coloumnSend = coloumnEnd - firstColoumn; 99. **if** (process != numberOfProcess - 1) { 100. **for** (rowSend = rowStart; rowSend <= rowEnd; rowSend++) { 101. **http://www.fileformat.info/info/unicode/font/arial_unicode_ms/u21AA.png**                    MPI\_Send( & numberOf*LCS*Character[rowSend][coloumnSend], 1, MPI\_INT, dest, 0,  MPI\_COMM\_WORLD); 102. } 103. } 104. } 105. } 106. finishedCalculating = 1; |

Kode Sumber 2 Fungsi Mencetak LCS

|  |
| --- |
| 1. **if** (finishedCalculating == 1) { 2. **if** (process == numberOfProcess - 1) { 3. **int** destBacktrack, coloumnStartBacktrack, coloumnEndBacktrack; 4. point valueForBacktrack; 5. **int** firstColoumn = ((process)\*lengthOfY / numberOfProcess); 7. destBacktrack = process - 1; 9. coloumnStartBacktrack =   (((process)\*lengthOfY / numberOfProcess) + 1) - firstColoumn; 10. coloumnEndBacktrack =   ((process + 1) \* lengthOfY / numberOfProcess) - firstColoumn; 11. **int** index = numberOf*LCS*Character[rowIndex][coloumnIndex]; 12. **int** initialIndex*LCS* = index - 1; 13. **int** lengthOf*LCS* = index; 15. vector<**char**> character*LCS*; 16. character*LCS*.resize(index); 18. **int** rowBacktrack = rowIndex, coloumnBacktrack = coloumnIndex; 19. **int** originalColoumnIndexY = coloumnBacktrack + firstColoumn; 20. **while** (coloumnBacktrack >= coloumnStartBacktrack  && coloumnBacktrack <= coloumnEndBacktrack) { 21. **if** (X[rowBacktrack - 1] == Y[originalColoumnIndexY - 1]) { 22. character*LCS*[index - 1] = X[rowBacktrack - 1]; 23. originalColoumnIndexY--; 24. rowBacktrack--; 25. coloumnBacktrack--; 26. index--; 27. } 28. **else** **if** (numberOf*LCS*Character[rowBacktrack - 1][coloumnBacktrack] > numberOf*LCS*Character[rowBacktrack][coloumnBacktrack - 1]) { 29. rowBacktrack--; 30. } 31. **else** { 32. coloumnBacktrack--; 33. originalColoumnIndexY--; 34. } 35. } 36. valueForBacktrack.rowForBacktrack = rowBacktrack; 37. valueForBacktrack.coloumnForBacktrack = coloumnBacktrack; 38. valueForBacktrack.index*LCS*ForBacktrack = index; 39. valueForBacktrack.originalColoumnBacktrack = originalColoumnIndexY; 40. **int** endIndex*LCS* = index; 42. MPI\_Send(&valueForBacktrack, 4, MPI\_INT, destBacktrack, 3, MPI\_COMM\_WORLD); 44. MPI\_Send(&initialIndex*LCS*, 1, MPI\_INT, 0, 4, MPI\_COMM\_WORLD); 45. MPI\_Send(&endIndex*LCS*, 1, MPI\_INT, 0, 4, MPI\_COMM\_WORLD); 47. MPI\_Send(&lengthOf*LCS*, 1, MPI\_INT, 0, 5, MPI\_COMM\_WORLD); 48. **for** (**int** sendCharacter = endIndex*LCS*; sendCharacter <= initialIndex*LCS*;  sendCharacter++) { 49. MPI\_Send(&character*LCS*[sendCharacter], 1, MPI\_CHAR, 0, 4, MPI\_COMM\_WORLD); 50. } 51. } 52. **if** (process < numberOfProcess - 1) { 54. **int** sourceBacktrackProcess, destBacktrackProcess; 55. **int** coloumnStartBacktrack, coloumnEndBacktrack; 56. **int** firstColoumn = ((process)\*lengthOfY / numberOfProcess); 57. point valueForBacktrack; 59. sourceBacktrackProcess = process + 1; 60. destBacktrackProcess = process - 1; 62. MPI\_Recv(&valueForBacktrack, 4, MPI\_INT, sourceBacktrackProcess, 3,  MPI\_COMM\_WORLD, &status); 63. **if** (process != 0) { 64. coloumnStartBacktrack =   (((process)\*lengthOfY / numberOfProcess) + 1) - firstColoumn; 65. coloumnEndBacktrack =   ((process + 1) \* lengthOfY / numberOfProcess) - firstColoumn; 66. **int** rowBacktrack, coloumnBacktrack, indexOf*LCS*; 67. **int** initialIndex*LCS*, endIndex*LCS*, originalColoumnIndexY; 68. rowBacktrack = valueForBacktrack.rowForBacktrack; 69. coloumnBacktrack = coloumnEndBacktrack; 70. indexOf*LCS* = valueForBacktrack.index*LCS*ForBacktrack; 71. originalColoumnIndexY = valueForBacktrack.originalColoumnBacktrack; 73. vector<**char**> character*LCS*; 74. character*LCS*.resize(indexOf*LCS*); 75. initialIndex*LCS* = indexOf*LCS* - 1; 76. **int** backtrack = 0; 77. **while** (coloumnBacktrack >= coloumnStartBacktrack &&  coloumnBacktrack <= coloumnEndBacktrack) { 78. **if** (indexOf*LCS* <= 0) 79. **break**; 80. **if** (X[rowBacktrack - 1] == Y[originalColoumnIndexY - 1]) { 81. character*LCS*[indexOf*LCS* - 1] = X[rowBacktrack - 1]; 82. rowBacktrack--; 83. coloumnBacktrack--; 84. indexOf*LCS*--; 85. originalColoumnIndexY--; 86. } 87. **else** **if** (numberOf*LCS*Character[rowBacktrack - 1][coloumnBacktrack] >  numberOf*LCS*Character[rowBacktrack][coloumnBacktrack - 1]) { 88. rowBacktrack--; 89. } 90. **else** { 91. coloumnBacktrack--; 92. originalColoumnIndexY--; 93. } 94. valueForBacktrack.rowForBacktrack = rowBacktrack; 95. valueForBacktrack.coloumnForBacktrack = coloumnBacktrack; 96. valueForBacktrack.index*LCS*ForBacktrack = indexOf*LCS*; 97. valueForBacktrack.originalColoumnBacktrack = originalColoumnIndexY; 98. endIndex*LCS* = indexOf*LCS*; 99. backtrack = 1; 100. } 101. MPI\_Send(&valueForBacktrack, 4, MPI\_INT, destBacktrackProcess, 3,  MPI\_COMM\_WORLD); 103. **char** flag[] = { '\0' }; 104. **int** flag\_notcount = 1; 105. **if** (backtrack == 1) { 106. MPI\_Send(&initialIndex*LCS*, 1, MPI\_INT, 0, 4, MPI\_COMM\_WORLD); 107. MPI\_Send(&endIndex*LCS*, 1, MPI\_INT, 0, 4, MPI\_COMM\_WORLD); 108. **for** (**int** sendCharacter = endIndex*LCS*; sendCharacter <= initialIndex*LCS*;  sendCharacter++) { 109. MPI\_Send(&character*LCS*[sendCharacter], 1, MPI\_CHAR, 0, 4,  MPI\_COMM\_WORLD); 110. } 111. } 112. **if** (backtrack == 0) { 113. initialIndex*LCS* = -1; 114. endIndex*LCS* = -1; 115. MPI\_Send(&initialIndex*LCS*, 1, MPI\_INT, 0, 4, MPI\_COMM\_WORLD); 116. MPI\_Send(&endIndex*LCS*, 1, MPI\_INT, 0, 4, MPI\_COMM\_WORLD); 117. } 118. } 119. **if** (process == 0) { 120. **int** initialIndex*LCS*, endIndex*LCS*, lengthOf*LCS*; 121. **int** coloumnBacktrack, rowBacktrack; 123. MPI\_Recv(&lengthOf*LCS*, 1, MPI\_INT, numberOfProcess - 1,  5, MPI\_COMM\_WORLD, &status); 124. vector<**char**> characterOf*LCS*; 125. characterOf*LCS*.resize(lengthOf*LCS*); 127. characterOf*LCS*[lengthOf*LCS*] = '\0'; 128. **int** finishReceived*LCS* = 0, finishBacktrackAndReceived = 0,  finishBacktrackProcess0 = 0; 129. **int** firstColoumn = ((process)\*lengthOfY / numberOfProcess); 130. coloumnStartBacktrack =   (((process)\*lengthOfY / numberOfProcess)) - firstColoumn; 131. coloumnEndBacktrack =   ((process + 1) \* lengthOfY / numberOfProcess) - firstColoumn; 133. **int** index*LCS*, originalColoumnIndexY; 135. rowBacktrack = valueForBacktrack.rowForBacktrack; 136. coloumnBacktrack = coloumnEndBacktrack; 137. index*LCS* = valueForBacktrack.index*LCS*ForBacktrack; 138. originalColoumnIndexY = valueForBacktrack.originalColoumnBacktrack; 139. **while** (coloumnBacktrack >= coloumnStartBacktrack &&  coloumnBacktrack <= coloumnEndBacktrack) { 140. **if** (rowBacktrack - 1 < 0 || coloumnBacktrack - 1 < 0 ||  originalColoumnIndexY - 1 < 0 || index*LCS* <= 0) 141. **break**; 142. **else** **if** (X[rowBacktrack - 1] == Y[originalColoumnIndexY - 1]) { 143. characterOf*LCS*[index*LCS* - 1] = X[rowBacktrack - 1]; 144. rowBacktrack--; 145. coloumnBacktrack--; 146. index*LCS*--; 147. originalColoumnIndexY--; 148. } 149. **else** **if** (numberOf*LCS*Character[rowBacktrack - 1][coloumnBacktrack] >  numberOf*LCS*Character[rowBacktrack][coloumnBacktrack - 1]) { 150. rowBacktrack--; 151. } 152. **else** { 153. coloumnBacktrack--; 154. originalColoumnIndexY--; 155. } 156. } 157. finishBacktrackProcess0 = 1; 159. **if** (finishBacktrackProcess0 == 1) { 160. **for** (**int** fromProcess = numberOfProcess - 1;  fromProcess >= 1; fromProcess--) { 161. MPI\_Recv(&initialIndex*LCS*, 1, MPI\_INT, fromProcess, 4,  MPI\_COMM\_WORLD, &status); 162. MPI\_Recv(&endIndex*LCS*, 1, MPI\_INT, fromProcess, 4,  MPI\_COMM\_WORLD,  &status); 163. **if** (initialIndex*LCS* == -1 && endIndex*LCS* == -1) { 164. } 165. **else** { 166. **for** (**int** received*LCS* = endIndex*LCS*;  received*LCS* <= initialIndex*LCS*; received*LCS*++) { 167. MPI\_Recv(&characterOf*LCS*[received*LCS*], 1, MPI\_INT,  fromProcess, 4, MPI\_COMM\_WORLD, &status); 168. } 169. } 171. finishReceived*LCS* = 1; 172. } 173. } 175. finishBacktrackAndReceived = 1; 176. } 177. } 178. } |

Kode Sumber 3 Pemilihan Indeks Perhitungan

|  |
| --- |
| 1. **int** rowStart = ((alpa \* (k - 1) \* lengthOfX) / numberOfProcess); 2. **int** rowEndPrevProcess = ((alpa \* (k - 2) \* lengthOfX) / numberOfProcess); 3. **int** rowEnd = (alpa \* k \* lengthOfX) / numberOfProcess; 5. **if** (rowStart != 0) 6. rowStart = rowStart + 1; 8. **if** (k != 1 && rowEnd == 0 && rowStart == 0) { 9. rowStart = rowStart + 1; 10. } 11. **if** (k != 1 && rowEndPrevProcess == 0 && rowStart == 0) { 12. rowStart = rowStart + 1; 13. } 15. **int** coloumnStart = (process)\*lengthOfY / numberOfProcess; 16. **if** (coloumnStart != 0) 17. coloumnStart = coloumnStart + 1; 19. **int** coloumnEnd = (process + 1) \* lengthOfY / numberOfProcess; |

Kode Sumber 4 Pemilihan Indeks Kolom Untuk Menerima Data

|  |
| --- |
| 1. **int** colStartForProcess1; 2. colStartForProcess1 = (1 \* lengthOfY / numberOfProcess); 3. **if** ((process \* lengthOfY / numberOfProcess) != colStartForProcess1) { 4. coloumnReceive = firstColoumn - coloumnPreviousProcess; 5. } 6. **else** { 7. coloumnReceive = coloumnStart - 1; 8. } |

Kode Sumber 5 Pengambilkan Karakter LCS

|  |
| --- |
| 1. **if** (X[rowBacktrack - 1] == Y[originalColoumnIndexY - 1]) { 2. character*LCS*[indexOf*LCS* - 1] = X[rowBacktrack - 1]; 3. rowBacktrack--; 4. coloumnBacktrack--; 5. indexOf*LCS*--; 6. originalColoumnIndexY--; 7. } 8. **else** **if** (numberOf*LCS*Character[rowBacktrack - 1][coloumnBacktrack] >  numberOf*LCS*Character[rowBacktrack][coloumnBacktrack - 1]) { 9. rowBacktrack--; 10. } 11. **else** { 12. coloumnBacktrack--; 13. originalColoumnIndexY--; 14. } |

Kode Sumber 6 Perhitungan Nilai Indeks LCS

|  |
| --- |
| 1. **if** (rowCalculate == 0 || coloumnCalculate == 0) { 2. numberOf*LCS*Character[rowCalculate][coloumnCalculate - firstColoumn] = 0; 3. } 4. **else** **if** (X[rowCalculate - 1] == Y[coloumnCalculate - 1]) { 5. numberOf*LCS*Character[rowCalculate][coloumnCalculate - firstColoumn] =  numberOf*LCS*Character[rowCalculate - 1][coloumnCalculate - 1 - firstColoumn] + 1; 6. } 7. **else** { 8. numberOf*LCS*Character[rowCalculate][coloumnCalculate - firstColoumn] =  max(numberOf*LCS*Character[rowCalculate - 1][coloumnCalculate - firstColoumn],  numberOf*LCS*Character[rowCalculate][coloumnCalculate - 1 - firstColoumn]); 9. } |

***[Halaman ini sengaja dikosongkan]***

BIODATA PENULIS

I Gusti Ngurah Adi Wicaksana, lahir pada 29 September 1995 di Denpasar. Penulis merupakan seorang mahasiswa yang tengah menempuh studi di Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember. Memiliki beberapa hobi antara lain mendengarkan dan bermain musik serta bermain futsal. Pernah menjadi asisten dosen di mata kulian Sistem Digital dan Kecerdasan Buatan, serta 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.

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