ANALISIS DAN PENERAPAN *DOCUMENT ORIENTED DATABASE* NOSQL PADA SISTEM INFORMASI AKADEMIK
(Studi Kasus: PT Ganesha Operation)

TESIS

Disusun sebagai salah satu syarat untuk memperoleh gelar Magister Komputer
dari Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI

Oleh :

MUSBAHNIAR
NPM. 2016210092

PROGRAM STUDI PASCASARJANA
MAGISTER SISTEM INFORMASI
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER LIKMI
BANDUNG

2018

ANALISIS DAN PENERAPAN *DOCUMENT ORIENTED DATABASE* NOSQL PADA SISTEM INFORMASI AKADEMIK (Studi Kasus: PT Ganesha Operation)

TESIS

Disusun sebagai salah satu syarat untuk memperoleh gelar Magister Komputer
dari Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI

Oleh :

MUSBAHNIAR
NPM. 2016210092

Bandung, Oktober 2018
Menyetujui,

Dr. Eng. H. Ana Hadiana
Pembimbing

PROGRAM STUDI PASCASARJANA
MAGISTER SISTEM INFORMASI
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER LIKMI
BANDUNG
2018

Kupersembahkan Tesis ini teristimewa kepada :

Istri Tercinta & Anakku Tersayang

Terima kasih untuk segala Doa dan Dukungannya Kalian adalah Semangatku

ABSTRAK

ANALISIS DAN PENERAPAN *DOCUMENT ORIENTED DATABASE* NOSQL PADA SISTEM INFORMASI AKADEMIK (STUDI KASUS : PT GANESHA OPERATION)

MUSBAHNIAR
NPM : 2016210092

Sebagai bagian dari pencapaian target, pengambilan keputusan bisnis melibatkan pemprosesan dan menganalisis data dalam jumlah yang besar dari suatu perusahaan yang diambil hari ke hari. Mempertimbangkan ukuran dan kerumitan *database* yang digunakan diperusahaan saat ini, itu adalah masalah besar bagi perusahaan untuk merekayasa ulang aplikasi yang menangani data dalam jumlah besar. Objek penelitian adalah PT. Ganesha Operation yang mempunyai gerai lebih dari 750 gerai dengan jumlah siswa yang dikelola pertahunnya lebih dari 250 ribu siswa. Tabel yang digunakan adalah tabel yang dibutuhkan bidang akademik.

Dibandingkan dengan *database relational*, *database* NoSQL lebih cocok untuk penyediaan data yang dinamis, skala horizontal, kinerja, dan manfaat untuk pengembanng. MongoDB merupakan *database* yang menggunakan model penyimpanan data berorientasi dokumen. Pada kenyataanya, peralihan data dari *database relational* ke suatu *database* NoSQL seperti MongoDB merupakan proses rekayasa yang kompleks tergantung kondisi datanya yang akan diproses.

Pada tesis ini mengusulkan metodologi untuk proses memigrasikan data RDBMS ke NoSQL serta mengevaluasi melalui analisis komparatif dari implementasi RDBMS dan NoSQL berdasarkan evalusi kinerja *query*, struktur *query*, dan kelincahan pengembang. Metodologi rekayasa ulang *database* dilakukan dengan analisis entitas yang dibutuhkan dilakukan dengan memanfaatkan struktur dan relasi antar tabel yang ada didalam skema sebagai parameter untuk pembuatan algoritma pembentukan model. Hasil implementasi dan evaluasi dari *database* NoSQL memperlihatkan secara signifikan lebih baik dari *database relational* untuk operasi penyimpanan data terkait INSERT, UPDATE dan DELETE. *Database* NoSQL juga mampu mengolah data dalam jumlah yang besar serta menunjukan hasil luar biasa untuk agregasi dan penyortiran data yang merupakan aspek untuk manajemen data analisis.

Kata Kunci : Database Relational, Database NoSQL, MongoDB, Model Dokumen, Rekayasa Ulang, Migrasi, RDBMS, Ganesha Operation, Akademik.

ABSTRACT

ANALYSIS AND APPLICATION OF DOCUMENT ORIENTED NOSQL DATABASE IN ACADEMIC INFORMATION SYSTEM (CASE STUDY : PT GANESHA OPERATION)

MUSBAHNIAR
NPM: 2016210092

As part of achieving the target, business decision making involves processing and analyzing large amounts of data from a company day by day. Considering the size and complexity of the database used in the company today, it is a big problem for companies to reengineer applications that handle large amounts of data. The object of the research is PT. Ganesha Operation, which has more than 750 outlets with more than 250 thousand students managed each year. The table used is a table needed by the academic field.

Compared to relational databases, the NoSQL database is more suitable for dynamic data provision, horizontal scale, performance, and development benefits. MongoDB is a database that uses document-oriented data storage models. In fact, the transfer of data from a relational database to a NoSQL database such as MongoDB is a complex engineering process depending on the condition of the data to be processed.

In this thesis proposes a methodology for the process of migrating RDBMS data to NoSQL and evaluating through comparative analysis of RDBMS and NoSQL implementations based on evaluation of query performance, query structure, and developer agility. Database reengineering methodology is carried out by entity analysis which is needed to be carried out by utilizing the structure and relations between tables in the scheme as parameters for creating model building algorithms. Implementation and evaluation results from the NoSQL database show significantly better than relational databases for data storage operations related to INSERT, UPDATE and DELETE. The NoSQL database is also capable of processing large amounts of data and showing extraordinary results for aggregation and data sorting which is an aspect for analytical data management.

Keywords: Relational Database, NoSQL Database, MongoDB, Documentation Model, Reverse Engineering, Migration, RDBMS, Ganesha Operation, Academic.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat serta hidayah-Nya serta berkat kerja keras penulis, akhirnya penyusunan Tesis ini dapat selesai pada waktunya.

Dengan tidak mengurangi makna syukur, penyusun ingin mewujudkan rasa terima kasih dengan ketulusan kepada :

1. Bapak Dr. Eng. H. Ana Hadiana selaku dosen pembimbing utama yang telah dengan rela dan penuh keikhlasan membimbing penulis dengan sangat sabar dan bijaksana ditengah tengah kesibukannya.
2. Bapak, ibu , kakak tercinta yang telah memberikan banyak doa dan dorongan dalam penyelesaian tesis maupun selama proses perkuliahan.
3. Bapak Dr. Hery Heryanto, S.Kom., M.Kom. atas masukan dan arahannya pada saat seminar.
4. Istri dan putri tercinta yang selalu memberikan motivasi dan doa dengan tulus untuk penulis supaya dapat menyelesaikan penulisan tesis ini.
5. Direktur dan Manajer Teknologi Informasi PT Pendidikan Ganesha Operation yang telah memberikan kesempatan pada penulis untuk terlibat dalam penyusunan tesis.
6. Seluruh Dosen Program Pascasarjana STMIK LIKMI Bandung yang telah memberikan ilmu dan pengetahuan selama perkuliahan berlangsung, semoga menjadi ilmu yang bermanfaat bagi penulis.
7. Rekan-rekan Pasca Sarjana kelas S 2016 STMIK LIKMI Bandung.
8. Kepada semua pihak yang telah membantu penyelesaian Tesis ini.

Penulis sangat menyadari betul akan keterbatasan kemampuan penulis miliki sehingga tesis ini pasti banyak kekurangan dalam penyusunan sehingga mengharapkan kritik dan sarannya demi kesempurnaannya.

Akhir kata penulis berharap meski banyak kekurangan disana sini semoga tesis ini dapat memberikan sumbangsih pemikiran dan mampu menjadi acuan implementasi Sistem Informasi Akademik bagi PT Pendidikan Ganesha Operation. Semoga bisa bermanfaat bagi penulis khususnya dan pembaca pada umumnya. Amin.

Bandung, Oktober 2018

Penulis

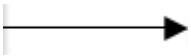
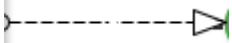
DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR SIMBOL

Notasi	Element	Penjelasan
	Activity-Task	Menyatakan pekerjaan yang harus diselesaikan dalam suatu proses.
	Activity-Subprocess	Menyatakan pekerjaan yang didalamnya dapat dipecah menjadi beberapa proses
	Start Event	Menyatakan sebuah proses dimulai
	Messages Event	Menyatakan sebuah proses dimulai/dipicu setelah adanya sebuah pesan
	Timer Events	Menyatakan sebuah proses dimulai/dipicu oleh sebuah pewaktu
	Rule Events	Menyatakan sebuah proses dimulai/dipicu oleh sebuah aturan (*rule*)
	Intermediate Event	Menyataan akhir sebuah proses dan memicu proses selanjutnya.
	End Event	Menyatakan akhir proses
	Intermediate Error Event	Menyatakan akhir sebuah proses dan memicu proses selanjutnya berdasarkan pesan kesalahan (*rule*)
	Exclusive Data-Base Gateway	Digunakan untuk menggambarkan percabangan alur proses dengan hanya satu proses setelah percabangan yang jalan pada saat yang sama
	Parallel Gateway	Digunakan untuk menggambarkan percabgan alur proses dengan proses percabangan yang berjalan paralel
	Sequence Flow Connector	Menunjukan urutan proses yang akan berlangsung
	Message Flow Connector	Menyatakan alur pesan dari dua entitas dengan salah satu entitas

Notasi	Element	Penjelasan
	Pool	Menggambarkan partisipan dalam proses bisnis

BAB I

PENDAHULUAN

1.1 Latar Belakang

NoSQL atau yang sering dikatakan *Not Only SQL* sering digunakan untuk menyimpan *Big Data*. NoSQL adalah tipe *database* baru yang menjadi lebih populer di kalangan perusahaan besar. NoSQL menyediakan kesederhanaan skalabilitas dan peningkatan kinerja dibanding *database* relational tradisional.

PT Ganesha Operation (GO) adalah lembaga yang berkecimpung dalam jasa pendidikan non formal. Dalam menjalankan bisnis jasa ini, manajemen lembaga memerlukan data untuk pengambilan keputusan. Keputusan yang berkualitas harus didasarkan pada data yang akurat dan terkini. Sehingga data yang baik sangat penting bagi kelangsungan mutu pelayanan di lingkungan PT Ganesha Operation.

Data yang terkini tidak akan terjadi jika server yang digunakan untuk mengelola data tersebut terlalu lambat dalam merespon permintaan (*request*) dari pengguna. Respon server yang lambat menyebabkan pengguna enggan mengakses data tersebut. Akibatnya, data yang diakses bukanlah data terkini dan keputusan yang dibuat bisa menjadi tidak tepat atau tidak berkualitas.

Di PT Ganesha Operation data yang dihasilkan oleh seluruh cabang disimpan dalam server yang sudah terdistribusi. Sampai saat ini, PT Ganesha Operation mempunyai jumlah unit (*outlet*) sebanyak 823 unit dengan jumlah siswa 251.241 untuk semua tingkatan. Pengguna yang mengakses server tidak hanya berasal dari karyawan namun juga siswa dan orangtua. Ada empat puluh satu kantor cabang yang masing-masing terdapat penguna rata-rata empat orang dan lima puluh siswa dan orangtua. Dengan begitu banyaknya pengguna yang mengakses server menyebabkan akses server terasa lambat ketika pengguna mengakses server. Akibatnya pengguna enggan menggunakan aplikasi untuk mengakses data dan memasukan data yang menyebabkan data tidak terbaharui.

Akses server yang bersamaan untuk seluruh pengguna membuat lambatnya respon server disebabkan jumlah data yang semakin besar/banyak untuk satu tahun ajaran

Ganesha Operation memiliki data internal sebesar 25Gb dan data eksternal mencapai 40Gb yang perhari untuk data kuis pada akademik melayani 800.000 *record* dan untuk 1 bulannya mencapai 24 juta *record*. Untuk itu diperlukan *database* yang mampu menangani jumlah data yang besar/banyak dan tetap tersebar. Pada *database* tersebar ada proses fragmentasi yang membagi data menjadi beberapa *database* dan ditempatkan pada server yang terpisah secara fisik namun tetap menjadi satu secara lojik. Isi pada server tersebut bisa saling diselaraskan karena ada proses replikasi pada server-server itu.

Masalah juga timbul jika server pusat mengalami gangguan dan terhenti beroperasi. Maka, seluruh pengguna di Indonesia tidak akan bisa mengakses data yang sangat diperlukan. Kerugian yang timbul oleh gangguan ini sangat besar karena timbulnya potensi pemasukan karena analisis data yang kurang tepat dan kepercayaan dari pengguna.

Dengan *database* tersebar akan terjadi pembagian akses pengguna ke server. Akses ke server oleh pengguna akan semakin ringan dan cepat karena satu server hanya menangani jumlah pengguna yang lebih sedikit, namun begitu ketersediaan data masih ada jaminan karena ada replikasi antar server (Davenport, 2007).

Sebuah solusi penting dari penanganan data dalam jumlah yang besar tersebut adalah manajemen *database* tingkat lanjut, dalam rangka memenuhi peningkatan kebutuhan besar kelompok pengguna dalam hal keandalan dan ketersediaan.

1.2 Rumusan Masalah

Penelitian ini difokuskan untuk memecahkan masalah-masalah yaitu :

1. Bagaimana kebutuhan penyimpanan *database* di lingkungan PT Pendidikan Ganesha Operation yang sangat besar/banyak ?
2. Bagaimana *database* yang sudah dirancang akan digunakan sebagai *database* NoSQL agar dapat menjawab tantangan perkembangan organisasi ?

1.3 Tujuan Penelitian

Penelitian ini mempunyai beberapa tujuan sebagai berikut :

1. Menganalisis sistem informasi akademik mengenai kebutuhan penyimpanan data dan respon permintaan dari pengguna serta pengaruh penerapan sistem *database* NoSQL.
2. Implementasi *database* NoSQL sehingga dapat menangani jumlah data yang besar/banyak dan peningkatan respon permintaan data.

1.4 Ruang Lingkup Penelitian

Ruang lingkup penelitian ini dibatasi sebagai berikut :

1. Perancangan tabel *database* hanya dikhususkan pada data yang berhubungan dengan Akademik.
2. Penelitian difokuskan pada bagaimana proses migrasi data dari *database* relational (RDBMS) menjadi NoSQL *database*.
3. NoSQL *database* yang digunakan adalah MongoDB 3.4 berlisensi *General Public License* (GPL) dan Sistem Operasi yang digunakan pada server adalah Ubuntu 14.04 Trusty Tahr.
4. Penelitian di lakukan di kantor pusat PT Pendidikan Ganesha Operation dan hanya di beberapa komputer sebagai simulasi dengan sistem yang sudah terdistribusi. Koneksi yang digunakan antar server menggunakan *Virual Private Network* (VPN) berbasis OpenVPN versi 2.3.2.

1.5 Sistematika Penulisan

Untuk mempermudah dalam penyusunan tesis ini maka dibuatkan sistematika penulisan sebagai berikut :

BAB I : PENDAHULUAN

Bab ini berisi mengenai latar belakang, rumusan maslaah, tujuan penelitian, ruang lingkup penelitian, dan sistematika penulisan.

BAB II : LANDASAN TEORI

Bab ini berisi tentang teori dasar database relasional, database NoSQL dan teori yang digunakan untuk menganalisis database yang dibutuhkan, pembuatan tabel, fragmentasi, replikasi, dan metodologi untuk pembangunan database NoSQL terdistribusi.

BAB III : OBJEK DAN METODOLOGI PENELITIAN

Bab ini berisi tentang deskripsi lembaga yang digunakan sebagai objek penelitian dan metode yang digunakan untuk penelitian.

BAB IV : ANALISIS DAN DESAIN

Bab ini berisi analisis dan rancangan *database* yang diperlukan di lembaga tempat penelitian. Rancangan *database* berupa *document oriented* dan *collection*.

BAB V : IMPLEMENTASI

Bab ini berisi penjelasan implementasi *database* NoSQL yang sudah dirancang pada bab sebelumnya dan implementasi pembangunan *database* NoSQL terdistribusi.

BAB VI : KESIMPULAN DAN SARAN

Bab terakhir ini berisi kesimpulan dari penelitian yang telah dilakukan dan saran yang diberikan agar penelitian bias mendapatkan hasil lebih sempurna.

BAB II

LANDASAN TEORI

2.1 Data, Informasi, *Database* dan *Database Management System*

Dalam setiap kegiatan, suatu organisasi selalu berhadapan dengan fakta-fakta yang ada. Organisasi sekolah berhadapan dengan fakta identitas siswa, nilai yang diperoleh, fakta kedatangan siswa dll. Fakta-fakta yang belum diolah atau masih mentah ini dinamakan data (Coronel & Morris, 2015). Data bisa juga didefinisikan sebagai berikut :

"Data is a symbol set that is quantified and/or qualified."

(Zins, 2010).

Fakta-fakta nilai siswa adalah karena data dapat dijumlahkan dan dapat dikelompokan menjadi siswa yang bernilai rendah atau tinggi. Fakta-fakta identitas siswa adalah data karena dapat dikelompokan siswa perempuan dan siswa laki-laki atau siswa yang rumahnya dekat dengan kampus atau yang jauh dari kampus.

Data ini bisa disimpan secara manual atau terkomputerisasi. Data identitas siswa, nilai siswa, atau kedatangan siswa bisa disimpan di kertas dan dikumpulkan dalam satu direktori. Petugas di lembaga tersebut bisa mengolah data yang sudah terkumpul untuk menggali makna yang dinamakan informasi (Coronel & Morris, 2015). Dari data tersebut petugas bisa mengetahui seberapa banyak siswa yang mempunyai nilai rendah, menengah, dan tinggi. Pihak lembaga bisa membuat suatu keputusan untuk menindaklanjuti makna yang terkandung dalam informasi tersebut.

Berbagai informasi yang dimiliki bisa dikumpulkan dan diorganisasikan. Kumpulan dari informasi ini mempunyai penjelasan tentang informasi yang disimpan dan kriteria-kriteria dari informasi tersebut atau bisa berisi data dan metadata. Kumpulan objek yang mempunyai data dan metadata ini dinamakan *database* (Powel, 2009). Data dalam *database* siswa adalah nama siswa, dan sebagainya. Metadata dalam *database* siswa adalah penjelasan bahwa nama siswa terdiri dari huruf, tanggal, lalu berisi data berbentuk tanggal dan panjangnya tidak lebih dari sepuluh huruf/angka, dan sebagainya. Pada jaman sekarang *database* sudah bisa dan biasa disimpan di komputer. Penguna bisa memasukan

dan mengelola data-data tersebut dengan bantuan aplikasi *database.* Aplikasi ini bisa merupakan aplikasi tunggal atau sekumpulan aplikasi yang saling terkait. Sekumpulan aplikasi yang digunakan untuk mengelola struktur *database* dan mengendalikan akses *database* ini dinamakan Sistem Pengelolaan *Database* atau *Database Management System* (DBMS) (Coronel & Morris, 2015). Michael V. Manino mendefinisikan DBMS dengan bahasa yang berbeda sebagai berikut :

"A database management system (DBMS) is a collection of components that supports the creation, use, dan maintenance of database."

(Powel, 2009)

Ini berarti dalam DBMS, aplikasi tersebut harus bisa membuat *database* yang meliputi mendefinisikan metadata dari *database* tersebut serta pembuatan data itu sendiri. DBMS juga harus bisa mengakses data tersebut dengan perintah yang ada dan juga pengelolaan pemeliharan *database*.

Sebuah DBMS harus bisa mengelola interaksi antara pengguna dengan *end user* dan DBMS bisa menyediakan kembali dari banyak pengguna. DBMS juga akan mengintegrasikan berbagai kebutuhan *view* dari pengguna ke dalam satu sumber data saja.

Ada banyak keuntungan jika pengelolaan *database* menggunakan DBMS. Keuntungan tersebut antara lain (Coronel & Morris, 2015):

1. Perbaikan berbagi data

   Dengan DBMS data mudah dibagi dari satu pengguna ke penguna yang lain. Jika ada perubahan yang dilakukan oleh satu pengguna, mudah bagi pengguna lain untuk mengetahui dan mengamati perubahan tersebut.

2. Peningkatan keamanan data

   Semakin banyak pengguna, resiko keamanan data akan semakin tinggi. DBMS mengatur keamanan tersebut sehingga resiko keamanan bisa diminimalkan.

3. Data integrasi yang lebih baik

DBMS tidak hanya mengelola satu tabel saja (misalkan data siswa), namun bisa mengelola tabel-tabel lain yang berkaitan dengan siswa. Banyak data tersebut bisa dintegrasikan oleh DBMS menjadi satu kesatuan.

4. Memperkecil inkonsistensi data

   Data inkonsistensi ini terjadi jika ada beberapa versi dari data yang sama berada di lokasi yang berbeda. Data inkonsistensi bisa membingungkan pengguna. DBMS bisa memperkecil inkosistensi tersebut dengan desain data di DBMS.

5. Peningkatan akses data

   DBMS bisa meningkatkan kemampuan akses ke *database* dengan adanya pengelolan *query*, *view*, *store procedure*.

6. Perbaikan pengambilan keputusan

   Semakin baik data dikelola, tingkat kekinian data akan semakin mudah sehingga keputusan yang diambil akan menjadi lebih berkualitas.

7. Menaikkan produktivitas pengguna

   Dengan keberadaan data yang mudah dan *tool* yang dimiliki oleh DBMS juga mudah digunakan, kemampuan pengelolaan *database* menjadi lebih cepat. Hal ini bisa meningkatkan produktifitas pengguna.

2.2 *Relational Database Management System*

Sebelumnya berkembang *Relational Database,* telah ada sebelumnya sistem *database* yang sudah dikembangkan yaitu *database* Model Hirarki dan *database* Model *Network. Database* model hirarki menyimpan data dalam bentuk pohon sehingga mudah untuk dikelola namun bersifat kaku. *Database* Model *Network* menyimpan data dalam bentuk *node-node* yang dihubungkan satu sama lain. *Database* model ini lebih fleksibel namun sulit untuk dipelajari. Relational *database* dirancang sebagai penengah untuk keduanya agar mudah dipelajari namun tetap fleksibel (Haryanto, 2007).

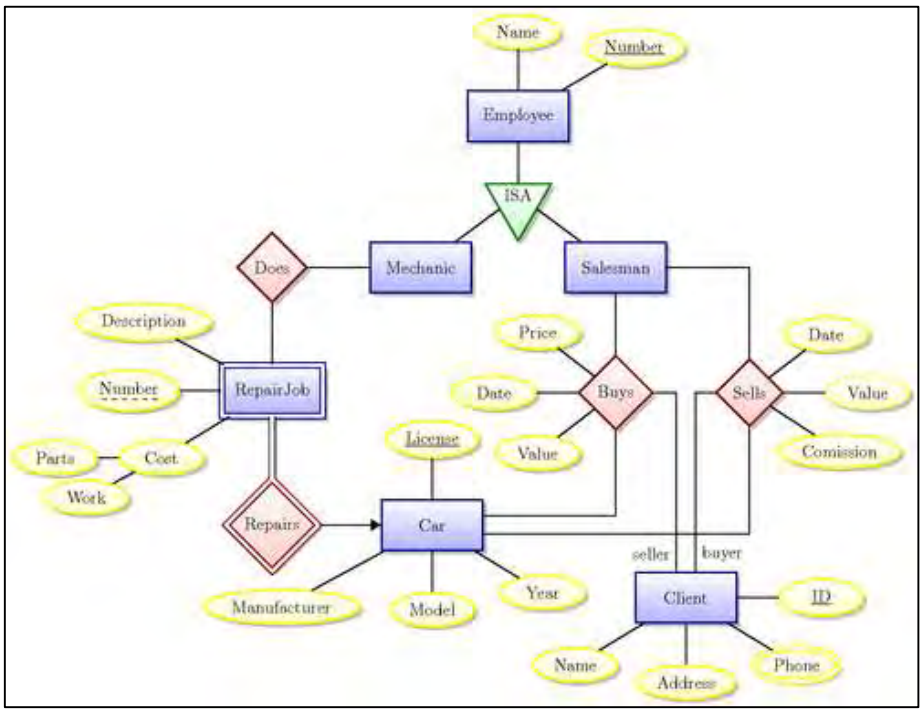
Mengenai relational *database* (Powel, 2009) menjelaskan sebagai berikut :

A relational database is a special type of database using structure called tables. Tables are linked together using what are called relationships. You can build tables with relationships between those, not only to organize your data, but also to allow later retrieval of information from the database.

Maka dalam relational *database* dikenal struktur tabel yang menyimpan suatu tipe informasi. Bisa saja dalam satu *database* disimpan banyak tabel dan masing-masing tabel menyimpan banyak baris yang dinamakan *record*. Tabel-tabel ini saling berkolerasi antar satu dengan yang lainnya sehingga pengguna bisa mengambil informasi baris dikaitkan dengan tabel lain yang berhubungan.

Sekumpulan aplikasi yang digunakan untuk membuat, mengelola, dan mengontrol *database* terhubung ini dinamakan *Relational Database Management System*. RDBMS harus bisa melakukan *update* dan pengembalian data dari tempat penyimpan data (Steven, 2007).

Dalam model *database* relational data disimpan dalam tabel yang berbeda melalui hubungan yang kemudian dapat diakses menggunakan hubungan ini. Hubungan antar entitas secara logis dalam bentuk satu-ke-satu, satu-ke-banyak atau banyak-ke-banyak. Gambar 2.1 menunjukan aktivitas hubungan antar entitas dalam diagram *Entity Relationship*.



Gambar 2.1
Entity Relationship (ER) Diagram
Sumber http://www.texample.net/tikz/examples/entity-relationship-diagram/

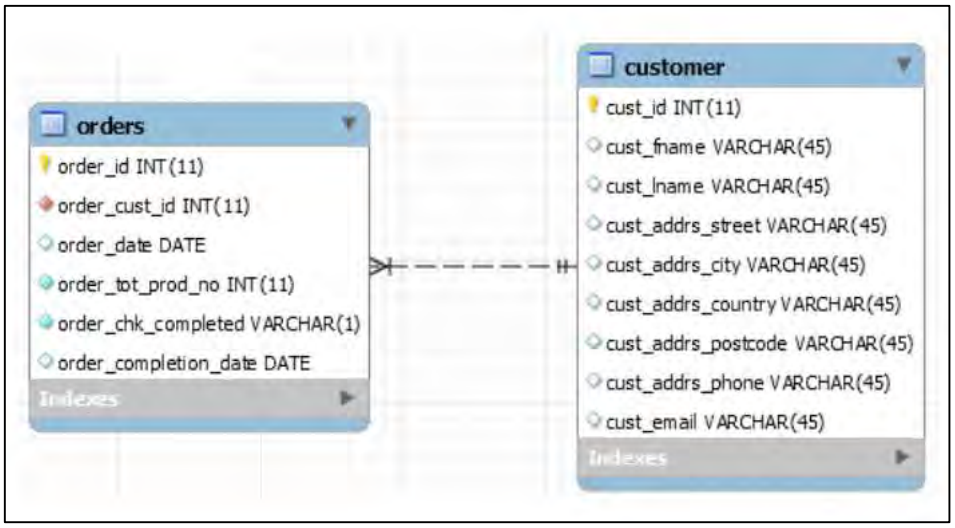
Struktur data dalam model relational didefinisikan oleh beberapa komponen yang dikenal sebagai entitas atau tabel atau tupel, domain, atribut dan relasi. E.F Codd juga mendefinisikan tampilan data relational oleh beberapa properti berikut :

1. Setiap baris dalam tabel mewakili catatan atau tupel.
2. Semua nilai atribut yang dicatat dalam tabel adalah atom (bagian terkecil).
3. Semu baris unik.
4. Urutan baris tidak signifikan.
5. Urutan kolom tidak signifikan.

2.2.1 Konsep Kunci pada RDBMS

Dalam RDBMS konsep kunci sangat penting karena kunci digunakan untuk mengidentifikasi catatan dan membuat hubungan antar tabel. Salah satu sifat penting dari model relational adalah keunikan baris yang juga dibuat menggunakan kunci. Ada tiga jenis kunci utama dalam model RDBMS yaitu sebagai berikut :

1. Kunci Kandidat (*Candidate Key*) adalah kunci yang digunakan untuk mengidentifikasi secara unik catatan apapun dalam tabel tanpa mengacu pada data lain. Kunci kandidat dapat mewakili satu kolom atau kombinasi beberapa kolom. Sebuah tabel dapat memiliki lebih dari satu kunci kandidat.
2. Kunci utama (*Primary Key*) adalah kunci di mana setiap baris menjadi unik, sebuah tabel dapat memiliki beberapa kunci kandidat tetapi hanya satu dari kunci-kunci itu dipilih sebagai kunci utama. Kunci utama memastikan keunikan baris dalam tabel dan mengurangi redudansi data. Ini dapat didefinisikan sebagai kolom tunggal atau kombinasi beberapa kolom. Gambar 2.2 menunjukan penggunaan kunci utama dan kunci kandidat dimana “order_id” dan “cust_id” sebagai kunci utama dari tabel orders dan tabel *customer*



Gambar 2.2
Relationship menggunakan Primary Key dan Foreign Key
(Couchbase, 2015)

3. Kunci asing (*Foreign Key*) mewakili kolom atau kumpulan kolom dalam tabel yang mengacu pada kunci utama dari tabel lain secara unik mengidentifikasi tabel tersebut. Pada Gambar 2.2 “order_cust_id” di tabel orders menunjukan kunci asing yang mengacu pada kunci utama “cust_id” yang didefinisikan pada tabel *customer*

2.2.2 Database ACID

Konsep penting untuk *database* agar memastikan proses transaksi yang aman dan andal dikenal sebagai sifat ACID (Date, 2013). Satu unit kerangka kerja yang dilakukan melalui suatu urutan operasi mengacu pada suatu transaksi. Untuk memenuhi syarat transaksi yang aman, andal, dan konsisten, kerangka kerja harus mengikuti empat properti *Atomicity, Consistency, Isolation, Durability* (ACID) yang dijelaskan sebagai berikut :

1. *Atomicity* adalah transaksi adalah atom (semua atau tidak sama sekali) (Date, 2013). Dalam suatu transaksi baik semua operasinya harus diselesaikan atau tidak ada yang akan diubah.
2. *Consistency* adalah setelah penyelesaian transaksi semua data harus dalam keadaan konsisten yang berarti *database* akan diperbarui hanya dengan data yang valid.

3. *Isolation* transaksi terisolasi satu sama lain (Date, 2013). Keadaan antara transaksi tidak dapat diakses atau dilihat oleh transaksi lainnya. Isolasi diperlukan untuk menjaga konsistensi antar transaksi.
4. *Durability* setelah penyelesaian transaksi yang berhasil diubah, maka data final perubahan tersebut harus dipastikan tersimpan di dalam sistem bahkan jika mengalami kegagalan sistem.

2.2.3 Data *Query Language*

Database ralational mengunakan bahasa yang terstruktur dengan baik untuk mendapatkan akses ke *database* dan kemudian mengambil informasi yang dikenal sebagai *Structured Query Language (SQL).* Selama tahun 1970an Donal D. Chamberlin dan Raymond F. Boyce mengembangkan bahasa untuk memanipulasi data yang di sebut *Structured English Query Language* untuk mendapatkan akses ke *database* relational terintegrasi (Boyce, 1974). Kemudian SEQL diubah menjadi SQL karena sudah dipatenkan oleh perusahaan lain. Tabel 2.1 memperlihatkan perbedaan pengunaan SQL pada beberapa RDBMS.

Tabel 2.1 Penggunaan SQL pada beberapa RDBMS

RDBMS	***Query Language***
IBM DB2, INFORMIX	SQL
Oracle	*Procedural Language*/ *Structured Query Language* (PL/SQL)
Microsoft SQL Server	Transact SQL
MySQL	SQL
Paradox	SQL
Postgre SQL	SQL

2.3 Prediksi Kecenderungan *Database* di Masa Depan

Relational *database*, pertama kali diperkenalkan oleh seorang ahli komputer E.F. Codd tahun 1970an, dikelola dengan *software* RDBMS (*Relational Database Management System*). relational *database* ini diimplementasikan dalam bentuk tabel-tabel yang saling berelasi satu sama lainnya. Pertengahan tahun 1980an mulai dikenalkan bahasa standar untuk mengelola dan mengakses relational *database*, yaitu

SQL (*Standard Query Language*). Selanjutnya relational *database* dengan *software* RDBMS yang menggunakan bahasa SQL menjadi sangat popular dan banyak digunakan dalam pengelolaan *database* selama puluhan tahun. Saat ini dominasi relational *database* mulai pudar karena munculnya jenis *database* baru yaitu NoSQL. NoSQL didefinisikan sebagai generasi baru *database* yang bersifat : *non-relational*, terdistribusi/tersebar, dan *open source. Database* dengan NoSQL menyediakan mekanisme yang lebih mudah dan sederhana untuk menyimpan dan mengambil data, dibanding dengan relational *database*.

Saat ini *database* sudah memakai logika secara kompleks. Fitur lainnya yang sedang dikembangkan adalah konsep abstrak *database* itu sendiri. Fitur ini memungkinkan *database* untuk ditempatkan di beberapa lokasi dan melakukan proses *query* sebagai unit yang berkelanjutan. Hal ini mungkin terjadi karena juga ditunjang oleh peningkatan kecepatan jaringan. Berikut adalah beberapa point tentang kecenderungan *database* di masa depan :

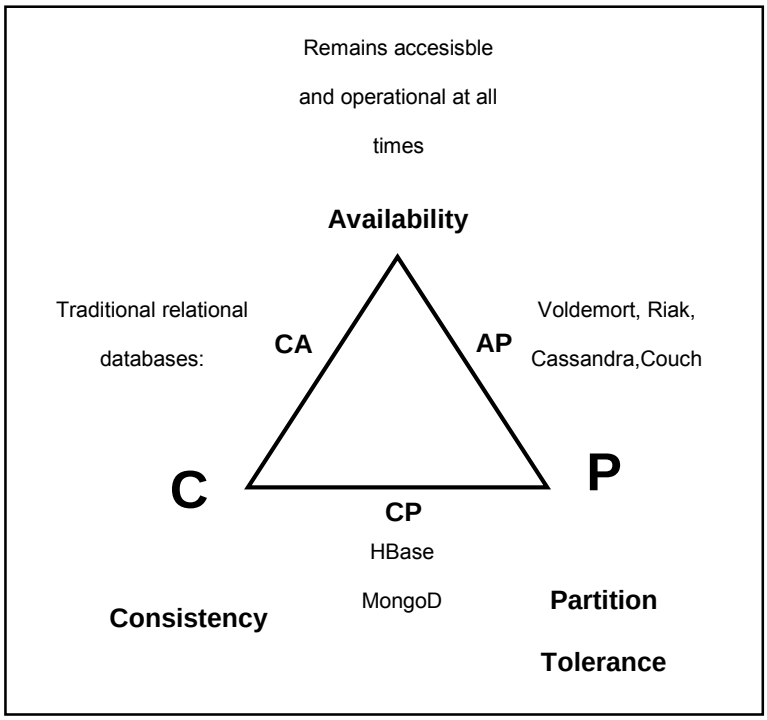
1. *Database* data manajemen berada pada titik balik.
2. Pergantian arsitektur seperti *cloud computing* dan kebutuhan untuk dapat menangani ukuran data dalam jumlah yang sangat besar.
3. *Database* akan digunakan diluar batas yang biasanya. Salah satu contohnya adalah Google's Map Reduce. Koleksi data (bukan *database*) semakin meningkat kepentingannya dalam mendapatkan ilmu pengetahuan. *Cloud, mobile*, dan aplikasi virtual akan menjadi penentu perubahan.

2.4 Konsep Dasar NoSQL

Bagian ini akan menjelaskan fitur yang dimiliki oleh *database* NoSQL terutama dalam hal distribusi data dan *query* dalam beberapa server yang sama. Telah banyak produk *database* NoSQL dengan berbagai keunggulannya masing-masing. Istilah NoSQL pertama kali digunakan pada tahun 1998 untuk relational *database* yang menghilangkan penggunaan SQL (Carlo, 2017). Istilah ini diambil lagi pada tahun 2009 dan digunakan untuk konferensi pendukung *database non-relational.*

Terlepas dari tren *database* pada saat ini, persepsi bahwa NoSQL berusaha untuk menghilangkannya penggunaan *database* relational adalah kesalahpahaman umum. Pendekatan yang lebih tepat adalah penggabungan *database non-relational* dengan penggunaan relational untuk memanfaatkan teknologi NoSQL maupun SQL untuk menyeimbangkan tuntutan kinerja skalabilitas. *Database* NoSQL didasarkan pada teori CAP yaitu pemilihan dua dari tiga aspek yang ada yang harus dipenuhi oleh oleh *database* yaitu *Consisteny*, *Availbility*, dan *Partition-Telerance* (Brewer, 2010), yang diperlihatkan pada Gambar 2.3

1. *Consistency Avaibility* (*CA*) berseberangan dengan *Partition-Tolerance* dan biasanya berhubungan dengan replikasi.
2. *Consistency PartitionTolerance* (*CP*) berseberangan dengan *Avaibility* dalam penyimpanan data.
3. *Avaibility Partition-Tolerance (*AP*)* sistem mencapai kondisi *eventual consistency.*



Gambar 2.3
CAP Theorem
(Brewer, 2010)

2.4.1 Model Data Pada NoSQL *Database*

Di bagian ini akan mengkategorikan berbagai jenis model data NoSQL menurut teknik penyimpanan data. NoSQL *database* dengan *schema-free* dimana dalam melakukan partisi untuk data yang berukuran besar, seperti melakukan *query,* replikasi, dan mendukung adanya konsistensi *NoSQL* mempunyai empat model data *NoSQL* yaitu *column-oriented, document-oriented, object-oriented dan graph-oriented* (Kang, 2010) NoSQL *database* diklasifikasikan ke dalam model data sebagai berikut.

2.4.1.1 *Column Oriented Data Store*

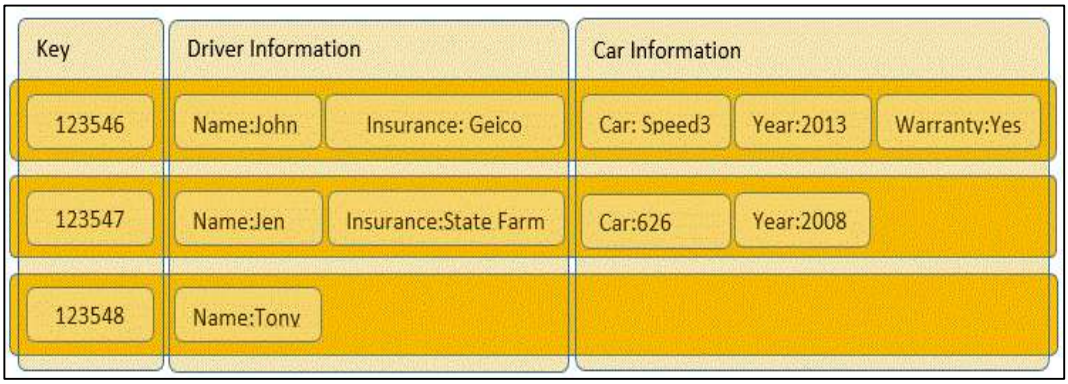
Penerepan *column-oriented* menggunakan distribusi multidimensional *map indexed* dengan sebuah *key*. Baris kunci yang sering digunakan adalah *string* dengan panjang 16-36 bytes. Setiap kolom digabungkan menjadi sebuah *coloumn families*. menerapkan beberapa konsep dalam penerapan *column-oriented*. Suatu kolom unit *atomic* dari informasi yang diekspresikan dengan nama *value*. *Super-column* merupakan gabungan dari kolom dengan nama yang umum dan digunakan untuk pemodelan tipe data yang komplek. Baris secara unik mengidentifikasikan data yang terdapat dalam *column* dan *super-column*. Baris dapat dikenali dengan sebuah kunci. *Column Family* merupakan bagian dari suatu unit abstraksi yang berisi baris kunci yang tergabung dalam *column* dan *super column* yang memiliki struktur data yang tinggi. *Keyspace* merupakan level tertinggi dari unit informasi yang terdapat dalam *column-oriented*. Kumpulan *column families* sebenarnya merupakan sub koordinat dari satu *keyspace*. Pada intinya model data *column-oriented* memungkinkan suatu aplikasi secara bebas untuk mengembangkan bagaimana informasi disusun berdasarkan suatu desain *schema*. Gambar 2.4 mewakili suatu contoh set data dan perbandingan penyimpanan data antara RDBMS dan *column-oriented.*

Data To Be Stored in	
RDBMS	**Column-Oriented Data Store**
SM1,Anuj,Sharma,45,10000000 MM2,Anand,,34,5000000 T3,Vikas,Gupta,39,7500000 E4,Dinesh,Verma,32,2000000	SM1,MM2,T3,E4 Anuj,Anand,Vikas,Dinesh Sharma,,Gupta,Verma, 45,34,39,32 10000000,5000000,7500000,2000000

Gambar 2.4
Penyimpanan data antara RDBMS dan *column-oriented.*
Sumber (Valentine, 2013)

Gambar 2.4 menunjukan bahwa masing-masing baris dalam RDBMS menyediakan satu set data lengkap termasuk tentang ID Karyawan, nama depan, nama belakang, usia dan gaji. Sedangkan set data yang sama diatur dalam kolom yang berbeda dari penyimpanan data *column oriented.* Seperti yang ditunjukan pada Gambar 2.5, beberapa atribut dapat dikelompokan dalam satu kolom dalam struktur data berorientasi yang juga disebut *column family.* Menurut (Moniruzzaman & Hossain, 2013) yang utama penggunaan penyimpanan data *column oriented* meliputi :

1. Penyimpanan data terdistribusi
2. Pemprosesan data skala besar yang berorientasi pada *bacth* yang mencakup pemilahan, penguraian, konversi, dll.



Gambar 2.5
Contoh Strutkur dari *Column Oriented Data Store*
Sumber (Valentine, 2013)

2.4.1.2 *Document Oriented Data Store*

Penyimpanan *document oriented data store* dirancang untuk mengelola dan menyimpan data dalam bentuk dokumen yang mencakup memasukkan, mengambil dan memanipulasi data semi-terstruktur (Vaisg, 2013). Agar memudahkan pekerjaan pengembang, itu beberapa dokumen berbeda yang ditampung dalam penyimpanan data dokumen bersifat independent dan bebas dari skema yang ditentukan. Contoh berikut yang diambil dari (Vaisg, 2013) menunjukkan dua dokumen berbeda yang disimpan dalam penyimpanan data dokumen untuk mendapatkan gambaran tentang dokumen penyimpanan data pada Gambar 2.6

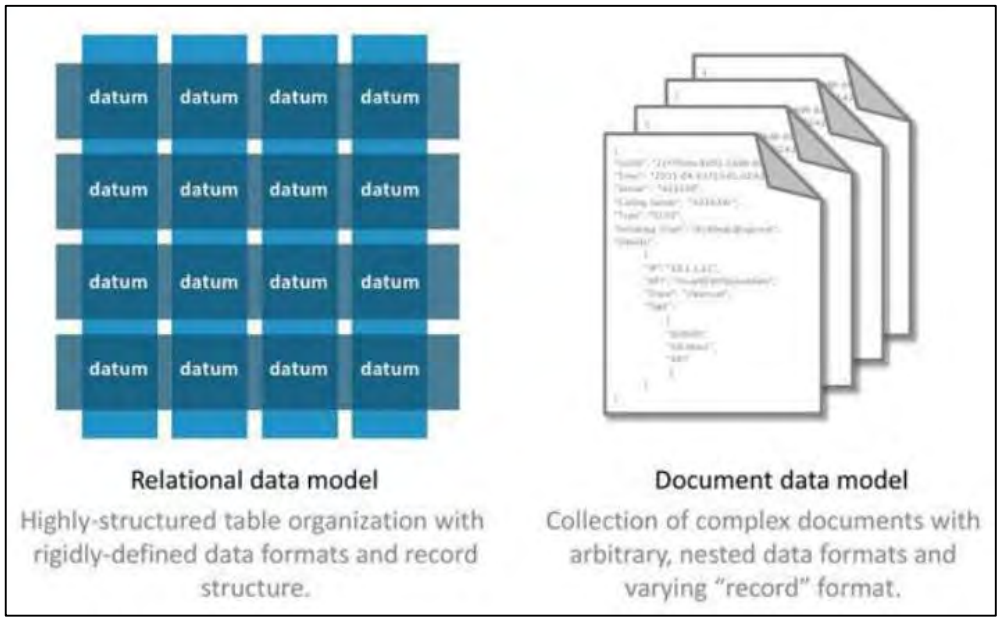
Document 1	Document 2

{
"EmployeeID": "SM1",
"FirstName" : "Islam",
"LastName" : "Shamima",
"Age" : 40,
"Salary" : 10000000
}
	{
	"EmployeeID": "MM2",
	"FirstName" : "Amar",
	"LastName" : "Prem",
	"Age" : 34,
	"Salary" : 5000000,
	"Address" : {
	"Street" : "123, Park Street",
	"City" : "Toronto",
	"Province" : "Ontario"
	},
	"Projects" : [
	"nosql-migration",
	"top-secret-007"
	]
	}

Gambar 2.6
Contoh Struktur Penyimpanan Data Dokumen
Sumber (Vaisg, 2013)

Dalam *document oriented data store*, format XML, JSON, BSON (Binary JSON) digunakan untuk menyimpan data di setiap dokumen. Gambar 2.6 menunjukkan dua dokumen berformat JSON di mana ‘Dokumen 1’ adalah dokumen terstruktur sederhana dan ‘Dokumen 2’ bersarang dengan sub-dokumen lain ‘Alamat’. ‘Dokumen 2’ juga berisi koleksi yang ditampilkan sebagai ‘Proyek’. Tetapi tidak satupun mewakili ID dokumen yang diperlukan dengan URL untuk mendapatkan akses ke *database* dokumen. Dalam pendataan berorientasi dokumen, sistem yang dihasilkan atau identifier yang ditentukan

oleh pengembang digunakan yang secara unik dialokasikan untuk masing-masing dokumen untuk mengidentifikasikan. Gambar 2.7 menunjukkan bagaimana empat *record* dari model data relational disimpan sebagai empat dokumen terpisah dalam penyimpanan data berorientasi dokumen.



Gambar 2.7
Record dari relational model disimpan di data dokumen
Sumber (Couchbase, 2015)

Menurut (Moniruzzaman & Hossain, 2013) model data dokumen terutama berguna untuk aplikasi berbasis web sebagai bagian dari pengelolaan dan pemrosesan data berskala besar yang didistribusikan dalam suatu jaringan termasuk dokumen teks, pesan email dan dokumen XML. MongoDB, CouchDB, Jackrabbit, Lotus Notes, Apache Cassandra, Terrastore, BaseX adalah contoh populer dari penyimpanan data berorientasi dokumen (Vaisg, 2013).

2.5 JSON (JavaScript Object Notation)

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari bahasa pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format

teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

JSON terbuat dari dua struktur :

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek, rekaman, struktur, kamus, tabel hash, daftar berkunci, atau associative array.
2. Daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik, vektor, daftar, atau urutan.

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

JSON menggunakan bentuk sebagai berikut:

1. Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).
2. Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

2.6 Evaluasi NoSQL *Database*

Menurut (Moniruzzaman & Hossain, 2013) daftar karakteristik NoSQL *database* dari empat kelompok besar yang telah dievaluasi. Pemilihan karakteristik *database* NoSQL di dasarkan pada bagaimana *database* NoSQL menyimpan data. Tabel 2.2 menunjukkan evaluasi dari beberapa penyimpanan data NoSQL yang didasarkan pada desain dan fitur, integritas data, pengindeksan, distribusi dan sistem yang ada pada setiap fitur *database* NoSQL.

Tabel 2.2 Evaluasi penyimpanan data NoSQL dari empat kelompok besar

Attribute	NoSQL Database					
	Database Model	Document Store		Wide Column Store		
	Fitur	MongoDB	CouchDB	DynamoDB	HBase	Cassandra
Desain	Data Storage	Volatile, Memory File Systems	Volatile, Memory File Systems	SSD	HDFS	
	Query Language	Volatile, Memory File Systems	Javascript protocol	API Calls	API Calls, REST, XML	API Calls
	Protocol	Custom, Binary (BSON)	HTTP, REST		HTTP, REST	Custom Binary
	Conditional Entry Update	Ya	Ya	Ya	Ya	Tidak
	MapReduce	Ya	Ya	Ya	Ya	Ya
	Unicode	Ya	Ya	Ya	Ya	Ya
	TTL for Entries	Ya	Ya	Tidak	Ya	Ya
	Compression	Ya	Ya	-	Ya	Ya
Inegritas data	Integrity Model	BASE	MVCC	ACID	Log	BASE
	Atomicity	Conditional	Ya	Ya	Ya	Ya
	Consistency	Ya	Ya	Ya	Ya	Ya
	Isolation	Tidak	Ya	Ya	Tidak	Ya
	Durability	Ya	Ya	Ya	Ya	Ya
	Transaction	Tidak	Tidak	Tidak	Tidak	Tidak
Index	Secondary Index	Ya	Ya	Tidak	Ya	Ya
	Composite Keys	Ya	Ya	Ya	Ya	Ya
	Full Text Seacrh	Tidak	Tidak	Tidak	Tidak	Tidak
Distributed	Horizontal	Ya	Ya	Ya	Ya	Ya
	Replication	Ya	Ya	Ya	Ya	Ya
	Replication Mode	Master-Slave	Master Slave	-	Master Slave	Master Slave
	Sharding	Ya	Ya	Ya	Ya	Ya

2.6.1 OLAP

Online Analytical Processing (OLAP) adalah kategori analisis data yang memfasilitasi respons cepat terhadap kueri multi dimensi (Codd, Codd, & Salley, 2015). Sebagai bagian dari kelompok *business intelligence* (BI) yang lebih luas, pendekatan ini digunakan untuk pelaporan bisnis termasuk area penjualan, pemasaran, dan terutama dalam pengambilan keputusan bisnis yang mencakup penganggaran dan peramalan. OLAP memungkinkan melakukan operasi analitik. Konsolidasi mengacu pada *roll-up*

informasi sebagai bagian dari penggabungan data untuk menganalisanya dalam cara multi-dimensi (Codd, Codd, & Salley, 2015). Kemudian dengan mengambil lebih dalam kemungkinan luas data agregat dapat diakses sesuai dengan jalur konsolidasi. Dan akhirnya pengguna bisa mendapatkan kumpulan data spesifik mereka dengan bantuan fitur pengiris dari OLAP.

2.6.2 Perbandingan RDBMS dan NoSQL *Database*

Poin-poin berikut telah dirangkum dan memberikan perbandingan analisis pada *database* relational dan NoSQL d*atabase* :

1. Keandalan transaksi : RDBMS mendukung properti ACID untuk menyediakan keandalan transaksi sedangkan *database* NoSQL tidak dapat diandalkan seperti RDBMS karena sifatnya properti BASE yang lebih lemah dibandingkan dengan ACID.
2. Model Data : relational *database* didasarkan pada model relational dimana tabel-tabel itu berisi set baris mewakili relasinya. Di sisi lain, *database* NoSQL mengambil banyak teknik pemodelan seperti penyimpanan nilai kunci, penyimpanan data dokumen, penyimpanan data kolom dan grafik model data.
3. Skalabilitas : aplikasi web berbasis internet memerlukan skalabilitas horizontal seperti tersebar di beberapa server di lingkungan terdistribusi. Dukungan penyimpanan data NoSQL skalabilitas horizontal dan itu merupakan tantangan besar untuk model relational.
4. *Cloud* : *database* relational tidak dapat menangani skema data yang kurang terstruktur yang dapat dilakukan hanya bekerja dengan skema yang terdefinisi dengan baik. Tetapi ini adalah salah satu persyaratan untuk penanganan database *cloud*. Namun, NoSQL *database* cocok untuk solusi *cloud computing*.
5. *Big Data* : karena masalah dengan skalabilitas dan distribusi data dalam lingkungan *cluster*, itu bukan tugas yang mudah untuk *database* relational untuk menangani data besar. Di *database* NoSQL sisi lain yang dirancang untuk menangani data yang besar dan didistribusikan di lingkungan terdistribusi.

6. Kompleksitas : kompleksitas hari ke hari dalam *database* relational meningkat karena adanya persyaratan yang berubah cepat secara terus menerus. Jika data untuk persyaratan yang diubah tidak tidak sesuai dengan skema RDBMS yang ada, maka itu akan membuat situasi yang rumit dalam hal mengubah skema dan kode pemrograman terkait. Di sisi lain tidak ada efek yang signifikan pada *database* NoSQL karena dapat menyimpan data tidak terstruktur, semi-terstruktur atau data terstruktur.
7. *Crash Recovery*: *Recovery* manager memastikan pemulihan *crash* untuk data RDBMS. Di atas pemulihan kerusakan tergantung pada replikasi data untuk *database* NoSQL. MongoDB menggunakan *file journal* sebagai mekanisme pemulihan.
8. *Security*: Mekanisme yang sangat aman diadopsi oleh RDBMS untuk mengamankan datanya. *Database* NoSQL dirancang untuk menyimpan dan menangani data besar, dan selanjutnya memberikan kinerja yang lebih tinggi dengan biaya keamanan.

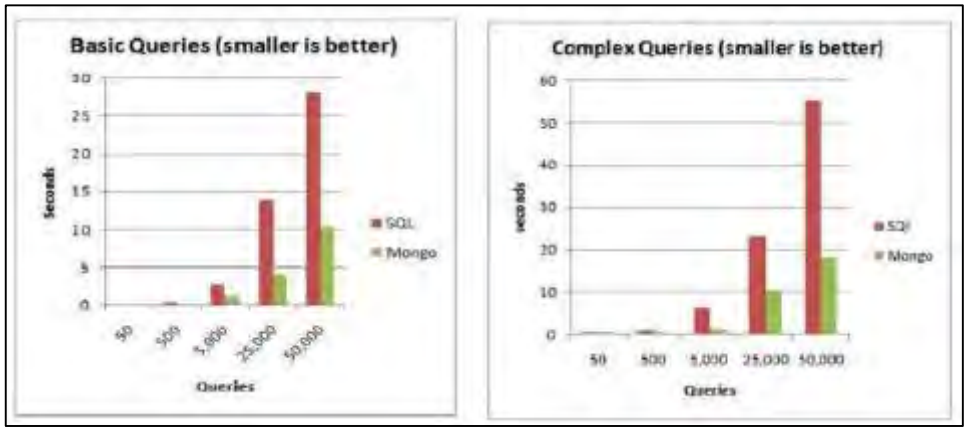
2.6.3 Perbandingan Kinerja Berdasarkan Penelitian

Untuk pengujian yang dilakukan menggunakan HeidiSQL 8.3 untuk MySQL, Robomongo 0.8.3 untuk MongoDB. Dalam (Wei ping, 2011), dilakukan pengujian perbandingan dengan NoSQL *database* (MongoDB) dengan MySQL dengan bantuan PHP. Penelitian sebelumnya telah melakukan pengujian dengan menggunakan beberapa *database* mentah untuk perbandingan kecepatan *query* dengan kinerja. Grafik yang diberikan menunjukkan hasil pengujian. Dalam pengujian kinerja, telah memasukkan 100 hingga 50.000 baris tipe yang sama dengan bantuan *looping* dalam pemrograman php agar dengan mudah dimasukkan. Waktu *query* MongoDB dan MySQL dicatat seperti yang ditunjukkan dalam grafik pada Gambar 2.8 dan Gambar 2.9

Number of Parallel Clients 5 Time in seconds

	Total Rows	Rows / client	SQL Time	Mongo Time	Sql Ops/sec	Mongo Ops/sec
Basic Query	50	10	0.1	0.08	500	625
with index	500	100	0.38	0.1	1,316	5,000
	5,000	1,000	2.8	1.2	1,786	4,167
	25,000	5,000	14	4	1,786	6,250
	50,000	10,000	28	10.4	1,786	4,808

Gambar 2.8
Perbandingan Kecepatan Query
(Kumar, Rajawat, & Joshi, 2015)



Gambar 2.9
Perbandingan Waktu Query Standar dan Kompleks
(Kumar, Rajawat, & Joshi, 2015)

Sistem manajemen relational d*atabase* tidak akan hilang untuk aplikasi d*atabase* besar karena memiliki kekurangan pada aplikasi seperti memakan waktu dan kecepatan eksekusi dan skala kueri. Tetapi kebutuhan penyimpanan untuk aplikasi generasi baru sangat berbeda dari aplikasi sebelumnya. Memilih NoSQL *database* bukan MySQL karena dua faktor, kemudahan penggunaan dan kinerja waktu. Dapat disimpulkan bahwa jika aplikasi web yang bersifat intensif data dan menyimpan banyak data, banyak kueri data, dan secara umum hidup *database*, maka akan lebih baik melakukannya secara efisien atau memiliki sumber daya (yaitu uang) untuk dibakar. Itu layak dilakukan untuk semua organisasi untuk mengembangkan aplikasi.

Menggunakan bahasa *query* SQL dasar untuk mendapatkan hasil dari *database* dan semua aturan konversi format akan dilakukan oleh metadata karena pengambilan data dari

database dalam bentuk format JSON. Sistem ini diusulkan karena MongoDB baru muncul, sedangkan bahasa SQL standar telah bertahun-tahun dan, oleh karena itu jika kita menggabungkan keduanya kita dapat menggunakan fitur dari kedua *database*. Meskipun, NoSQL *database* memiliki keuntungan dari ekspansi horizontal, tetapi untuk permintaan SQL yang kompleks, tidak dapat mendukungnya dengan sangat baik. Untuk *query* berdasarkan *key/value* dan persyaratan penyimpanan data yang sangat besar, NoSQL *database* adalah pilihan yang sangat berharga untuk semua pengembang dan organisasi lain yang mengembangkan aplikasi besar.

2.6.4 *Horizontal Scaling*

Horizontal scaling memungkinkan *database* dijalankan pada beberapa server untuk meningkatkan kemampuan perangkat penyimpanan dan meningkatkan efisiensi waktu. Hal ini memerlukan kemampuan dinamis pemartisian data dalam serangkaian *node* (seperti *storage hosts*) dalam suatu *cluster server*. Kemampuan untuk meningkatkan kemampuan dengan menambahkan beberapa komputer sangatlah penting dilakukan untuk data yang jumlahnya banyak, karena *vertical scaling* dilakukan dengan meningkatkan kemampuan spesifikasi *single server* (misalnya penambahan prosesor, memori, dan peralatan penyimpanan) terbatas untuk dilakukan dan lebih mahal (Leavitt, 2010). *Horizontal scaling* berarti memungkinkan dilakukannya penambahan server dalam satu jaringan dan user tidak sadar jika ada *hardware* yang diganti dari sisi server (*transparent*). Ada beberapa teknik partisi yang digunakan dalam *database* untuk melakukan horizontal *scaling*, salah satunya adalah *consistent hashing* yang digunakan dalam Cassandra dan Amazon Dynamo. Kunci dari menerapkan *consistent hashing* adalah membuat suatu lingkaran atau “ring”. Setiap *node* dalam sistem yang ditandai dengan *random value* dalam suatu *space* yang merepresentasikan posisi dari ring. Suatu kunci item ditandai dengan sebuah *node* untuk memperoleh posisinya pada ring, kemudian berpindah menuju *node* selanjutya yang sudah ditandai. *Node* memerankan peranan penting sebagai koordinator untuk kunci yang akan digunakan dalam *route request*. Kemudian, setiap *node* menjadi tanggung jawab dari daerah yang ada di ring diantara *node* dan *node* dalam ring sebelumnya.

2.7 *Sharding*

Sharding adalah istilah yang digunakan untuk menjelaskan praktek penggunaan beberapa server pada sebuah *database* yang sama dan melakukan konfigurasi untuk dapat menyimpan data secara terpisah di server-server yang berbeda. Proses ini akan meningkatkan kinerja *database* karena tiap server akan menangani data yang berbeda, jika satu server *database* menjadi sangat besar maka mempengaruhi waktu eksekusi *query*. Pada Gambar 2.10 terlihat tiga komponen pada *sharding*.

Ada kemungkinan suatu tabel dipakai seluruhnya oleh semua *client* di setiap *node*. Ini berarti setiap ada perubahan di sebuah *node*, *node* yang lain juga harus mengikuti perubahan itu. Untuk itu diperlukan metode penyalinan data yang dinamakan replikasi.

Replikasi sendiri dijelaskan sebagai berikut :

> "Database replication is the process of creating and maintaining multiple instances of the same database and the process of sharing data or database design changes between databases in different locations without having to copy the entire database". (Mazilu, 2010)

Dengan replikasi seperti yang diuraikan oleh Mazilu maka replikasi mempunyai kegiatan menciptakan, merawat, dan berbagi *database*. Menciptakan *database* terjadi jika seorang *client* membuat *database* di sebuah *node* maka di *node* yang lain akan diciptakan *database* yang sama. Perubahan yang terjadi karena proses perawatan seperti mengubah struktur, *update*, *delete*, dan lain-lain harus diikuti berbagi perubahan tersebut ke *node* yang lain.

Proses replikasi tidak hanya bertujuan berbagi data saja, namun ada tujuan-tujuan lainnya antara lain :

1. Meningkatkan ketersediaan data

   Jika di sebuah tempat akan diakses beberapa data dan performansi *server* di tempat tersebut melambat, maka data yang sama bisa ditemukan di *server* yang lain. Begitu juga jika ada masalah komunikasi dengan sebuah *server*, maka data yang sama bisa ditemukan di *server* yang lain.

2. *Query* yang lebih cepat.

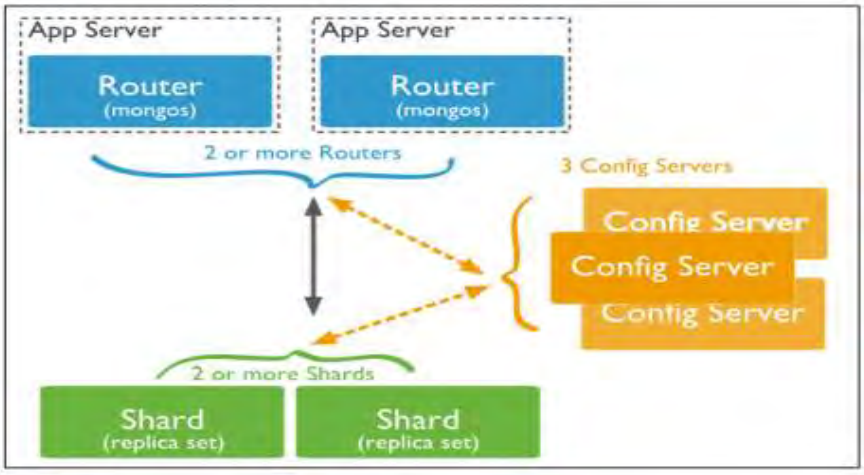
   Query akan lebih cepat karena data yang sebenarnya akan diakses di *server* lain terdapat juga di *server* lokal.

3. *Node decoupling*

   Setiap transaksi akan dikirim tanpa ada koordinasi dengan server yang lain. Sehingga jika ada sebuah *node* yang mati, transaksi tidak bisa disalin ke *node* tersebut. Namun ketika *node* tersebut hidup kembali, kopi transaksi akan dilakukan.

4. Mengurangi lalu lintas jaringan pada jam sibuk

   Seringkali pemutahiran data terjadi pada jam sibuk ketika dibutuhkan respon *query* yang cepat. Dengan replikasi bisa ditunda replikasi tersebut dan dilakukan ketika diantara *query* tersebut ada waktu lenggang, replikasi terjadi.



Gambar 2.10
Sharding pada MongoDB mengunakan *sharded cluster*
(Anomin, MongoDB NoSQL Document Database, 2017)

Pada *sharding* terdapat tiga kompenen utama yaitu :

1. *Shards* digunakan untuk menyimpan data. *Shards* menyediakan ketersediaan dan konsistensi data yang tinggi dalam lingkungan produksi, setiap pecahan adalah satu set replikasi terpisah.
2. Server konfigurasi menyimpan metadata *cluster*. Data ini mengandung sebuah pemetaan kumpulan data klaster ke pecahan. Router kueri menggunakan metadata ini

untuk menargetkan operasi ke pecahan tertentu. Di lingkungan produksi, gugus *sharded* memiliki tepat 3 konfigurasi server.

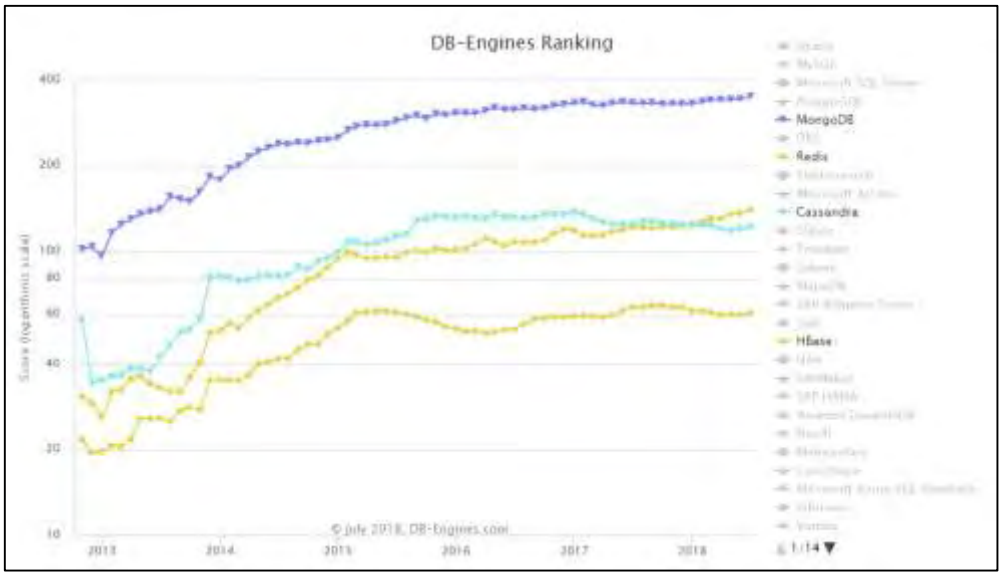
3. *Query routers* pada dasarnya adalah *instance mongo*, antarmuka dengan klien aplikasi dan operasi langsung ke pecahan yang sesuai. Router kueri memproses dan menargetkan operasi ke pecahan dan kemudian mengembalikan hasil ke klien. *Cluster sharded* dapat berisi lebih dari satu *router query* untuk membagi beban permintaan klien. Klien mengirim permintaan ke satu router permintaan. Umumnya, satu *cluster sharded* memiliki banyak *router query*.

2.8 Pemilihan NoSQL *Database*

Menurut Google Trends, MongoDB masuk dalam pencarian yang lebih sering muncul daripada beberapa *database* NoSQL lainnya seperti CouchDB, Cassandra, Redis dan HBase. Berdasarkan pencarian yang ada pada Google Trens pada tahun 2018 didapatkan nilai 350.325 untuk MongoDB, 139.912 untuk Redis, 21.057 untuk Cassandra, dan 60.77 untuk Hbase.

Tren pencarian ini mencerminkan bagaimana MongoDB semakin populer dari hari ke hari. Mempertimbangkan karakteristik yang mencakup kesederhanaan, kelincahan, dan fitur yang ramah untuk pengembang yang tersedia pada MongoDB itu akan menjadi pilihan yang baik untuk memenuhi penerapan dan tujuan dari tesis ini.

Gambar 2.11 memperlihatkan grafik pencarian yang sering muncul menurut Google Trends.



Gambar 2.11
DB Engines Ranking
Sumber https://db-engines.com/en/ranking_trend

2.8.1 MongoDB

Selain menjadi *top ranking* dalam pencarian Google Trends, MongoDB ditulis dalam C++ yang dapat melakukan hal-hal dengan cepat dan dokumen berbasis sumber terbuka JSON *database* MongoDB adalah NoSQL *database* yang populer di antara pilihan opsi NoSQL yang ada. Tersedia dalam banyak platform dan memanfaatkan standar yang mendukung sebagian besar bahasa populer seperti C#, Python, Ruby, PHP, JavaScript dan dapat berjalan pada sistem operasi baik di Windows, Mac atau Linux. Fitur MongoDB termasuk berbasis JSON dokumen untuk menyimpan data, fleksibilitas, replikasi yang mengarah pada ketersediaan tinggi, dukungan pengindeksan, *auto sharding* untuk skalabilitas horizontal, permintaan data dan *MapReduce*.

MongoDB mendukung distribusi data melalui beberapa mesin yang disebut *Sharding* dan yang juga merupakan bagian dari skala data. Masing-masing mesin tempat data berada didistribusikan dan dapat direplikasi untuk menghindari kehilangan data. Pemrosesan kueri dilakukan oleh MongoDB adalah cara yang sangat sederhana termasuk memilih indeks, menemukan dokumen dan akhirnya mengirim output sebagai dokumen BSON (Binary JSON) ke soket.

Fitur-fitur menarik yang mencakup model data, kemudahan dan kecepatan permintaan data membuatnya lebih populer untuk pengembang. Cara menerapkannya menggunakan memori file yang dipetakan. Ini menggunakan sebanyak mungkin memori untuk mengoptimalkan kinerja. MongoDB mendukung distribusi data melalui beberapa mesin yang disebut *Sharding* yang juga merupakan bagian dari penskalaan data. Dan masing-masing mesin di mana data didistribusikan dapat direplikasi untuk menghindari hilangnya data.

2.8.2 Perbandingan Sintak MySQL dan MongoDB

Untuk manipulasi data, *database* MySQL menggunakan bahasa SQL yang menyediakan fungsionalitas seperti INSERT, UPDATE, DELETE, dan SELECT. Di samping itu MongoDB menggunakan fungsi yang tersedia dalam *Application Programming Interfaces* (API JavaScript) untuk manipulasi datanya. Bagian ini mewakili beberapa perbedaan sintaks antara MySQL dan MongoDB untuk operasi yang sama. Tabel 2.3 mencakup beberapa perintah *query* yang digunakan oleh MySQL dan MongoDB untuk operasi yang sama.

Tabel 2.3 Perbedaan Sintak SQL pada MySQL dan MongoDB

Operations	**MySQL *Syntax***	**MongoDB *Syntax***
Membuat table/collection	CREATE TABLE ‘customer’ (‘cust_id int(11) NOT NULL, ‘first_name’ Varchar(45) DEFAULT NULL,’last_name’ Varchar(45) DEFAULT NULL);	Collection dibuat pada saat pertama kali memasukan data
Menghapus table/collection	DROP TABLE customer	db.customer.drop();
Menambah data	INSERT INTO customer (cust_id,first_name,last_name) VALUES (1,’John’,’Alvares’);	db.customer.save({‘cust_id’:1, ‘first_name’:’John’, ‘last_name’:’Alvares’})
Mengedit data	UPDATE customer SET first_name=’Jhonn’ WHERE cust_id=1	db.customer.update({‘first_name’;’Jhonn’},{$set:{‘cust_id’:1}}})
Menghapus data	DELETE FROM customer WHERE cust_id=1	db.customer.remove({‘cust_id’:1})
Mengambil data	SELECT * FROM customer WHERE first_name=’Jhon’	Db.customer.find({‘first_name’:’Jhon’})
Mengurutkan data	SELECT * FROM customer ORDER BY first_name	db.customer.find().sort({‘first_name’:1})

MongoDB bekerja berdasarkan konsep koleksi dan dokumen. Tabel 2.4 memperlihatkan hubungan istilah antara RDBMS dengan mongoDB

Tabel 2.4 Hubungan Istilah RDBMS dengan MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Table Join	Embedded Document
Primary Key	Primary Key (Default key_id provided by MongoDB itself
Database Server dan Client	
MySQLD	Mongod
MySQL	Mongo

2.9 Metode Pengembangan *Database Non-Relational*

Pendekatan strategi pada kondisi tertentu suatu perusahaan/organisasi perlu melakukan pergantian sistem sehingga diperlukan sebuah cara untuk menyesuaikan *database* pada sistem lama agar dapat dipakai di sistem baru. Ada dua pendekatan strategi untuk mengembangkan *database*. Strategi tersebut adalah *top down* dan *bottom up* (Oszum & Patrik, 2011) .

Pendekatan *top-down* dijelaskan berikut ini :

> Top-down approaches stress an initial focus on knowledge of higher-level contructs, such as identification of populations and collections of things and etity types membership rules, and relationship between such populations. Adoption of a top-down approach will generally start with a set of high-level requirements, such as a narrative. These requirements start a process of identifying the types og things needed to represent data with as well as the attributes og those things, which may become attributes in tables (Kung & Adrian, 2012)

Pendekatan *top-down* digunakan untuk menganalisa mengenai proses bisnis yang ada diobjek penelitian. Agar *database* pada sistem lama dapat dipakai di sistem baru dilakukan dengan rekayasa ulang (*reverse engineering*) *database*. Dengan rekayasa ini diharapkan *database* lama dapat digunakan kembali pada sistem baru tanpa harus membangun data dari awal.

Rekayasa ulang (*reverse engineering*) adalah proses penganalisisan sistem dengan maksud mengindentifikasikan komponen-komponen sistem dan keterkaitan antar

mereka dan mempresentasikan sistem tersebut dalam bentuk yang mempunyai tingkat abstraksi yang lebih tinggi (Sonhaji, 2010).

Kelengkapan proses rekayasa ulang (*reverse engineering*) mengacu pada tingkat detail yang diberikan pada suatu tingkat abstraksi. Kelengkapan meningkat berbanding lurus dengan jumlah yang dilakukan. Rakayasa ulang (*reverse engineering*) *database* terdiri tiga tahap yaitu translasi skema *database*, konversi data, dan translasi program *database* (Ahmadi, 2012).

Dalam tesis ini akan digunakan pendekatan *top-down* untuk analisis proses bisnis dan rekayasa ulang *database* untuk membangun *database non-relational*.

2.9.1 Analisis Kebutuhan

Tahap ini merupakan tahap yang sangat kritis dan sangat menentukan keberhasilan desain *database*. Tahapan ini berawal dari kebutuhan organisasi akan *database*. Kebutuhan organisasi bisa diperoleh dari beberapa metode (Talab, 2013) pencarian fakta antara lain : wawancara secara personal dengan karyawan, kusioner, observasi, dan dokumen. Secara umum langkah untuk mengetahui kebutuhan data sebuah organisasi adalah :

1. Mengumpulkan formulir yang ada.
2. Mengkaji kebijakan dan sistem yang berlaku.
3. Wawancara dengan pengguna tentang data-data yang ada.
4. Menentukan proses-proses yang dikerjakan sehari-hari.

Dengan langkah di atas diharapkan tujuan berikut tercapai :

1. Mengumpulkan data yang ada digunakan oleh organisasi.
2. Identifikasi relasi antar data.
3. Identifikasi kebutuhan data di masa mendatang.
4. Menentukan bagaimana data digunakan dan dibangkitkan.

Kebutuhan organisasi tersebut kemudian dianalisis untuk mengetahui area data yang akan dibuat. Area yang akan dianalisis bergantung dari tujuan pembuatan *database*.

Tahapan ini akan menghasilkan keluaran berupa sekumpulan detail kebutuhan data yang disimpan dalam sebuah kamus data. Keluaran detail kebutuhan data tidak hanya berupa data yang dibutuhkan untuk dibuat secara lojik namun juga kebutuhan untuk penyebaran data tersebut. Keluaran ini lah yang kemudian digunakan untuk mendesain data secara konseptual.

2.9.2 Analisis Bisnis Proses

Untuk menyusun *database* secara *top-down* maka perlu dianalisa proses bisnis yang terjadi di sebuah perusahaan/lembaga. Proses bisnis sendiri didefinisikan

> "a business process is a series of steps designed to produce a product or service. Most process(…) are cross-functional, spanning the white space between the boxes on the organitation chat. Some process result in a product of service that is received by an produce product that are invisible to the external customer but essential to the effective management of the business. We all these support processes."
> (Rummier, 2010)

Sesuai dengan definisi di atas, proses bisnis menggambarkan tahapan-tahapan yang dilakukan oleh sebuah organisasi atau unit organisasi untuk mengubah sesuatu menjadi sebuah produk atau layanan.

Untuk menggambarkan proses bisnis dalam bentuk diagram alir (*flow chart*) bisa menggunakan perkakas (*tools*) berupa notasi-notasi yang telah disepakati yaitu *Business Process Modeling Notation* (BPMN). BPMN diperkenalkan pada tahun 2004 sebagai BPMN versi 1.0 yang sekarang telah bekembang menjadi versi 2.0 pada tahun 2008.

2.9.3 Rekayasa Ulang *Database*

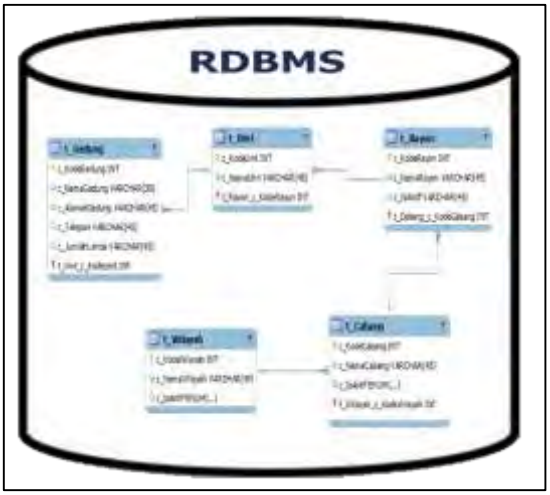
2.9.3.1 Translasi Skema *Database*

Translasi skema merupakan proses awal dalam melakukan rekayasa ulang *database*. Translasi skema memiliki dua pendekatan yaitu :

1. *Direct Translation*: pendekatan ini secara langsung merubah skema relasional ke skema non-relasional. Pendektan ini rawan terjadi kehilangan informasi karena *direct translation* cenderung tidak dapat melakukan *recovery* atau mengindentifisikasi semua skema awal.

2. *Indirect Translation*: pendekatan ini secara bertahap dimana skema baru didapat dengan memetakan kembali skema yang lama sehingga kemungkinan terjadinya hilang informasi dapat diminimalisir. Metode ini memerlukan translasi kembali skema lama ke model skema baru, dalam hal ini model yang digunakan adalah model skema *entity relationship* (ER).

Translasi skema menggunakan model ER dari suatu skema *database* yang ada pada MySQL mempertimbangkan struktur data dan teknik penyimpanan, *database* NoSQL berbeda dengan RDBMS. Model relational sangat terstruktur dan data tersebut dinormalkan menjadi tabel yang berbeda sesuai dengan hubungan antar tabel sedangkan penyimpanan data NoSQL adalah semi terstruktur atau tidak terstruktur dan menyimpan data dengan cara de-normalisasi. Data yang yang akan dimigrasikan akan memanfaatkan informasi yang terdapat pada "*information_schema*" untuk menganalisis metadata yang terkait dengan struktur tabel, relasi, dan kardinalitas data. *Information Schema* merupakan *database* bawaan DBMS yang digunakan untuk menyimpan metadata dari seluruh skema *database* yang didefinisikan pada MySQL. Metadata pada *information_schema* inilah yang akan dilakukan *reverse engineering* untuk mendapatkan struktur tabel, relasi dari fisik menjadi lojik skema dan nanti akan menjadi parameter utama dalam pengimplementasian migrasi *database*. ER secara umum memiliki gambaran seperti pada Gambar 2.12.



Gambar 2.12
Model Entity Relationship Diagram
(Ahmadi, 2012)

2.9.3.2 Konversi Data

Konversi data merupakan teknik mengubah *database* dari sistem lama agar dapat digunakan di sistem yang baru. Konversi data yang baik mampu mengubah sumber data ke *database* tujuan tanpa menghilangkan informasi penting yang ada di dalamnya (Ahmadi, 2012). Konversi data dapat dilakukan dengan proses ekstraksi termasuk pengambilan data dari MySQL tabel, lalu ubah data ini menjadi objek menggunakan pemetaan relational objek dan akhirnya memuatnya ke dokumen menggunakan JSON.

Konsep dasar pembentukan model dokumen ini adalah mengubah semua *tuple* data yang ada pada setiap tabel di relational *database* menjadi dokumen-dokumen pada MongoDB. Dokumen yang dibentuk dapat berupa representasi dari sebuah *tuple* atau gabungan beberapa *tuple*. Ditetapkan beberapa aturan baku yang digunakan sebagai dasar algoritma pembentukan model dokumen. Aturan-aturan ini akan mengacu pada jumlah, tipe, dan nilai kardinalitas relasi pada suatu tabel sebagai parameter utama dalam pembentukan model. Terdapat dua buah tipe kardinalitas. Kedua tipe kardinalitas tersebut yaitu *full reference* dan *partial reference*. Jika data pada tabel A diacu oleh data pada tabel B dan setiap data yang ada pada tabel A pasti diacu oleh data-data yang ada pada tabel B, maka tipe kardinalitas dari relasi antara tabel A dan tabel B dapat dikatakan sebagai *full reference*. Sedangkan jika data pada tabel A diacu oleh data pada tabel B dan tidak semua data yang ada pada tabel A diacu oleh data-data yang ada pada tabel B, maka tipe kardinalitas dari relasi antara tabel A dan tabel B dapat dikatakan sebagai *partial reference*.

Pembentukan aturan ini didasarkan pada analisis terhadap karakteristik relasi antar data dalam *database* relasional, dengan adanya analisis ini maka model dokumen yang terbentuk akan mengikuti alur relasi yang ada.

1. Aturan relasi A : pembentukan model untuk tabel yang tidak berelasi dengan tabel lain jika suatu tabel tidak berelasi dengan tabel lainnya di dalam skema *database*, maka semua *tuple* pada tabel tersebut akan menjadi dokumen tunggal pada MongoDB dan tergabung dalam *collection* yang sama.

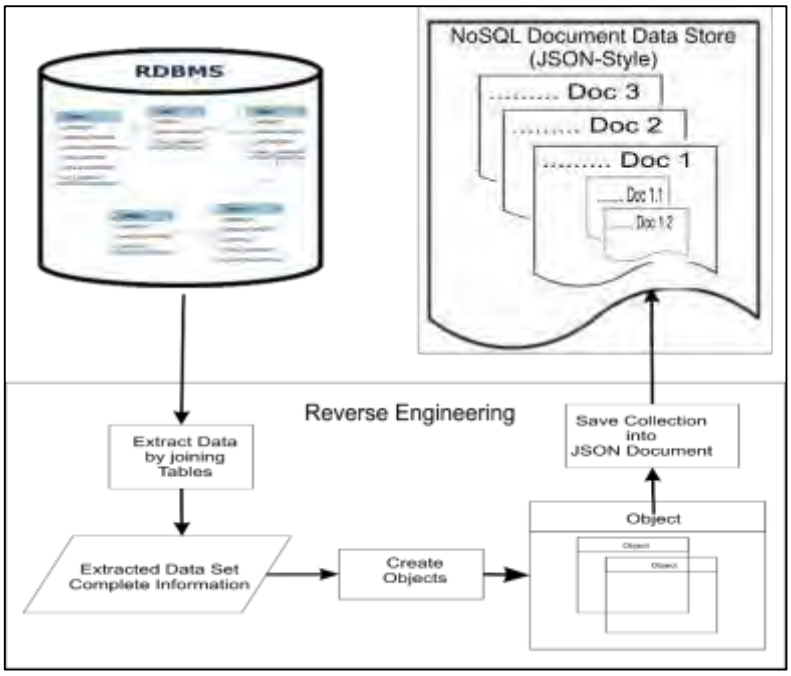
2. Aturan relasi B : pembentukan model untuk tabel yang hanya berelasi dengan sebuah tabel lain Jika suatu tabel (misalnya tabel A) hanya berelasi dengan satu buah tabel lain (misalnya tabel B), maka terdapat beberapa aturan yang akan diberlakukan dalam proses pembentukan model dokumen:
    a. Jika nilai kardinalitas relasi 1:1 atau relasi N:1 dan relasi antar kedua tabel tersebut bertipe *full reference*, maka model dokumen yang akan terbentuk adalah *embedded document*.
    b. Jika nilai kardinalitas N:M dan relasi antar kedua tabel tersebut bertipe *full reference*, maka model dokumen yang akan terbentuk adalah *embedded document* dimana *tuple* pada tabel A akan menjadi bagian atau sub-data dari *tuple* yang ada pada tabel B.
    c. Jika relasi antar tabel A dan tabel B bertipe *partial reference*, maka model dokumen yang akan terbentuk adalah *embedded document* dimana *tuple* pada tabel A akan menjadi bagian dari *tuple* yang ada pada tabel B. Selain itu pula, semua *tuple* pada tabel A juga akan menjadi dokumen tunggal.
3. Aturan relasi C : pembentukan model untuk tabel yang berelasi dengan beberapa tabel lain. Jika suatu tabel (misalnya tabel A) memiliki relasi dengan beberapa tabel lain (misalnya tabel X), maka untuk setiap tabel (tabel X[i]) yang berelasi dengan tabel A akan dikenakan aturan sebagai berikut :
    a. jika tabel $X_{[i]}$ hanya berelasi dengan tabel A, maka model dokumen yang akan terbentuk mengikuti aturan pembentukan model pada aturan B.
    b. Jika tabel $X_{[i]}$ tidak hanya berelasi dengan tabel A melainkan juga dengan tabel lainnya dan data pada tabel A mengacu pada data di tabel tersebut, maka model dokumen yang akan terbentuk adalah *referenced document* dimana setiap *tuple* pada tabel A akan mengacu data-data yang ada pada tabel $X_{[i]}$. Selain itu, semua *tuple* pada tabel $X_{[i]}$ juga akan menjadi dokumen tunggal.

2.9.3.3 Translasi Program *Database*

Tahap terakhir dalam rekayasa ulang *database* adalah translasi program *database*. Translasi program *database* adalah proses men-translasi *database* eksternal agar dapat diterima dan digunakan oleh sistem target. Translasi program *database* memiliki lima pendekatan yaitu :

1. *Rewriting*
2. *Software Interface*
3. *Emulation*
4. *Decompilation*
5. *Co-exisence*

Emulation merupakan salah satu metode dalam translasi program database dimana pendekatan dengan metode ini melibatkan dukungan software atau firmware pada sistem target yang dapat memetakan *commands* dari program/sistem sumber ke program/sistem target (Ahmadi, 2012). Gambar 2.13 memperlihatkan secara lengkap alur kerja dari metode rekayasa ulang *database*.



Gambar 2.13
Alur Kerja Rekayasa Ulang *Database*
(Hevner, 2010)

Aliran kerja pengembangan *database* NoSQL secara lengkap yang akan digunakan pada tesis ini menunjukkan langkah-langkah sebagai berikut :

1. Langkah 1 : Melakukan analisis mengenai kebutuhan organisasi akan *database* yang dilakukan dengan analisis proses bisnis
2. Langkah 2 : Translasi skema *database* dengan hubungan detail yang didefinisikan dalam skema *database*, dan selanjutnya merancang dan dikembangkan kriteria gabungan sesuai dengan hubungan antar tabel untuk mendapatkan informasi lengkap.
3. Langkah 3 : Konversi data dengan merancang dan mengembangkan skema implisit untuk menyimpan data ke model dokumen.
4. Langkah 4 : Merancang dan mengembangkan diagram kelas berdasarkan analisis data dan skema implisit.
5. Langkah 5 : Translasi program *database* untuk migrasi data.

2.10 Penelitian Terkait

Ada beberapa penelitian yang sudah dilakukan tentang implementasi dan perbandingan *database* MongoDB NoSQL pada sistem *database non-relational*. Tabel 2.4 menjelaskan beberapa penelitian terkait yang pernah dilakukan.

Tabel 2.4 Penelitian Terkait

No	Penelitian Tahun	Judul	Penjelasan
1.	Kurniawanto, dll, 2016	Implementasi database mongodb Untuk sistem informasi Bimbingan konseling berbasis web (Studi Kasus: SMP N 1 Sawahlunto)	Penelitian ini menjelaskan tentang cara membangun database mongoDB (bagaimana instalasi)
2.	Chaitanya, 2016	Implementation of an Efficient MongoDB NoSQL Explorer for Big Data Visualization	Menjelaskan tentang mongoDB di uji dan evalusi untuk penggunaan data yang besar yang lebih efisian
3.	Humasak, dll, 2016	Query response time comparison Nosqldb mongodb With sqldb oracle	Penelitian ini menjelaskan bagaimana kinerja dari respon query antara mongoDB dan Oracle

No	Penelitian Tahun	Judul	Penjelasan
4.	Bhaswara, dll, 2017	Perbandingan Kemampuan Database NoSQL dan SQL dalam Kasus ERP Retail	Makalah ini menjalaskan bagaimana kinerja dari mongoDB dan mySQL pada ERP retail
5.	Kumar, dll, 2016	Comparative analysis of NoSQL (MongoDB) with MySQL Database	Makalah ini menjalaskan bagaimana kinerja dari mongoDB dan mySQL.
6.	Ghotiya, dll, 2017	Migration from relational to NoSQL database	Menjelaskan tentang bagaimana cara-cara desain dan prinsip-prinsip untuk merubah database relational menjadi noSQL database
8.	Winaya, dll, 2016	Transformasi Skema Basis Data Reational Menjadi Model Data Berorientansi Dokumen pada MongoDB	Penelitian ini membahas tentang pengembangan sistem transformasi skema basis data relational menjadi model data berorientasi dokumen pada MongoDB dengan memanfaatkan
			struktur dan relasi yang ada pada skema relational
9.	Hanine, dll, 2015	Data Migration Methodology From relational to NoSQL Databases	Penelitian ini menjelaskan tentang metodologi migrasi dati RDBMS menjadi NoSQL dengan pembuatan algoritma yang dijalankan pada aplikasi menggunakan Java

Dari tulisan penelitian yang diperoleh pada umumnya menjelaskan masalah teknis tentang implementasi dan *design* MongoDB. Tidak ada penelitian yang menjelaskan tentang perhitungan jumlah *node* dan kebutuhan spesifikasi komputer yang digunakan. Penelitian umumnya menggunakan MongoDB yang merupakan *database* yang menggunakan model penyimpanan data berorientasi dokumen. Pada kenyataanya untuk beralih dari relational *database* ke suatu *database* NoSQL seperti MongoDB bukanlah perkara yang mudah terlebih lagi jika data yang dimiliki sangat kompleks dan suatu berjalan

pada suatu sistem. Berdasarkan dokumentasi yang telah dilakukan oleh beberapa perusahaan global dan penelitian terkait penggunaan MongoDB, dapat disimpulkan bahwa proses migrasi memakan waktu cukup banyak adalah transformasi skema yang terdapat pada relational *database* menjadi model data berorientasi dokumen pada MongoDB.

Dalam tesis ini membahas tentang :

1. Pengembangan sistem menggunakan MongoDB dengan sistem yang sebelumnya sudah berjalan dengan menambahkan proses transformasi dilakukan dengan memanfaatkan struktur dan relasi antar tabel yang ada di dalam skema sebagai parameter pembentukan model.
2. Dalam proses pembentukan model dokumen perlu dilakukan penyesuaian terhadap spesifikasi dokumen yang ditetapkan oleh MongoDB agar model dokumen yang terbentuk dapat diimplementasikan pada MongoDB.
3. Penerapan pada sistem informasi akademik yang memiliki proses *read* dan *write* yang tinggi karena diakses oleh seluruh cabang Ganesha Operation Se-Indonesia. Hal ini akan mempengaruhi performansi sistem.

BAB III

OBJEK DAN METODOLOGI PENELITIAN

3.1 PT Pendidikan Ganesha Operation

Ganesha Operation adalah lembaga pendidikan non formal yang telah mengalami perkembangan pesat. Semula yang hanya menempati satu garasi sekarang telah berkembang sampai ratusan gerai.

3.1.1 Sejarah Berdirinya Ganesha Operation

Pada tahun 80-an, persaingan masuk ke Perguruan Tinggi Negeri (PTN) sangat ketat, peserta yang merupakan lulusan Sekolah Lanjutan Tingkat Atas (SLTA) sangatlah banyak sedangkan daya tampung perguruan tinggi negeri sangat terbatas.Persaingan yang ketat menimbulkan keinginan siswa SLTA untuk menambah jam belajar agar bisa bersaing masuk ke PTN, fenomena ini memunculkan bisnis bimbingan belajar di Bandung di era tersebut.

Pada tanggal 2 Mei 1984, LPK Ganesha Operation (GO) yang sekarang PT Pendidikan Ganesha Operation berdiri untuk membantu siswa agar bisa bersaing masuk PTN. Setelah 34 tahun berdiri GO berada di 256 kota dengan jumlah gerai sebanyak 752 gerai.

3.1.2 Visi Ganesha Operation

Visi GO adalah “Menjadi lembaga bimbingan belajar terbaik dan terbesar se Indonesia”.

3.1.3 Misi Ganesha Operation

Untuk mencapai visi yang sudah dicanangkan, Ganesha Operation menetapkan misi yaitu :

1. Mendidik siswa agar berprestasi tingkat sekolah, kota/kabupaten, propinsi, nasional, dan internasional.
2. Melakukan inovasi pembelajaran melalui terobosan revolusi belajar dan teknologi informasi.
3. Meningkatkan budaya belajar siswa.
4. Meningkatkan mutu pendidikan.
5. Mencerdaskan kehidupan bangsa.

3.1.4 *Value* Ganesha Operation

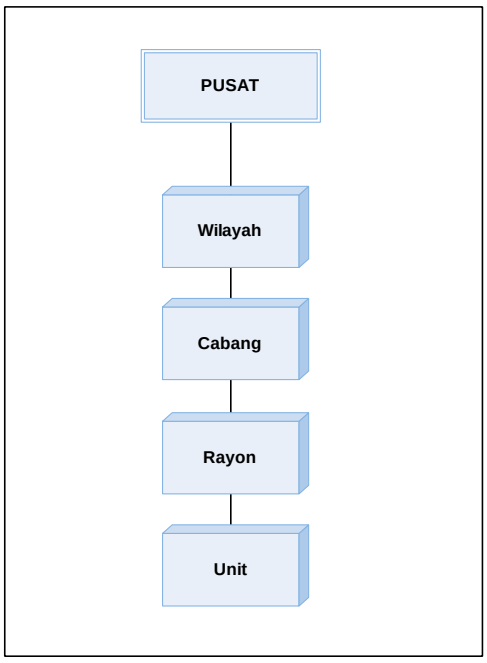
Dalam setiap kegiatan yang dilakukan oleh semua karyawan, ada batasan-batasan yang menjadi dasar tindakan karyawan.Batasan-batasan yang disebut *value* ini membatasi agar tindakan yang dilakukan tidak bertentangan dengan norma-norma dan menjadi pijakan untuk tetap meraih visi. *Value* yang dianut oleh Ganesha Operation disingkat IPEACE yang terdiri dari

1. *Integrity* : setiap karyawan GO harus jujur, melakukan apa yang dikatakan dan mengatakan apa yang dilakukan, bertanggung jawab terhadap pekerjaan yang diberikan dengan prinsip moral yang tinggi, membuat laporan kepada atasan secara jujur tanpa ditutup-tutupi.
2. *Passion* : setiap karyawan Ganesha Operation harus mengerjakan tugas dengan semangat yang tinggi, gigih, dan pantang menyerah, dan selalu mencari informasi untuk menangani masalah.
3. *Excellent* : setiap karyawan Ganesha Operation harus melakukan persiapan sebelum melakukan sesuatu, *pay attention to detail,* memberikan hasil yang melebihi dengan apa yang diminta, dan mencinta pekerjaan.
4. *Assist* : setiap karyawan Ganesha Operation harus punya empati, sinergi, bekerja sama, dan tidak mencari kambing hitam jika ada masalah.
5. *Consistent* : setiap karyawan Ganesha Operation harus disiplin, tekun, dan tetap fokus pada visi

6. *Enthusiasme* : setiap karyawan Ganesha Operation harus mengeluarkan potensi terbaik membuat sesuatu yang biasa menjadi luar biasa, dan mempengaruhi lingkungan agar tetap semangat.

3.1.5 Struktur Organisasi

Secara kewilayahan PT Ganesha Operation di seluruh Indonesia terbagi menjadi beberapa wilayah. Sebuah wilayah membawahi beberapa cabang; sebuah cabang membawahi beberapa rayon. Pada umumnya rayon merupakan sebuah kota yang terdiri lebih dari satu gerai. Gambar 3.1 menjelaskan struktur kewilayahan tersebut



Gambar 3.1
Struktur Kewilayahan

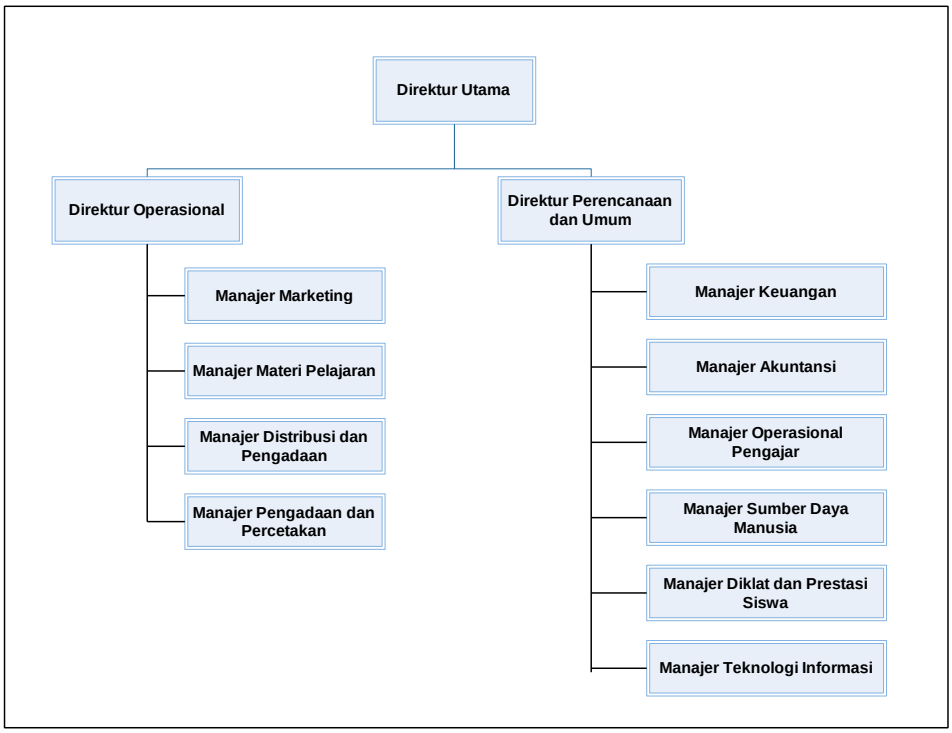
Ganesha Operation dipimpin oleh tiga orang direksi yaitu Direktur Utama, Direktur Operasional, dan Direktur Perencanaan dan Umum. Direktur Operasional membawahi Direktorat Operasional dan Direktur Perencanaan dan Umum membawahi Direktorat Perencanaan Umum. Dalam kesehariannya direksi dibantu oleh manajer untuk melaksanakan tugas perencanaan, pengorganisasian, pengkontrolan, dan pelaksanaan kegiatan. Direktorat Operasional membawahi empat bidang yang masing-masing dipimpin

oleh seorang manajer. Gambar 3.2 struktur organisasi yang ada di pusat, adapun keempat bidang Direktorat Operasional adalah :

1. Bidang Marketing
2. Bidang Distribusi dan Logistik (BDL)
3. Bidang Materi Pelajaran (BMP)
4. Bidang Pengadaan dan Percetakan (BPP)

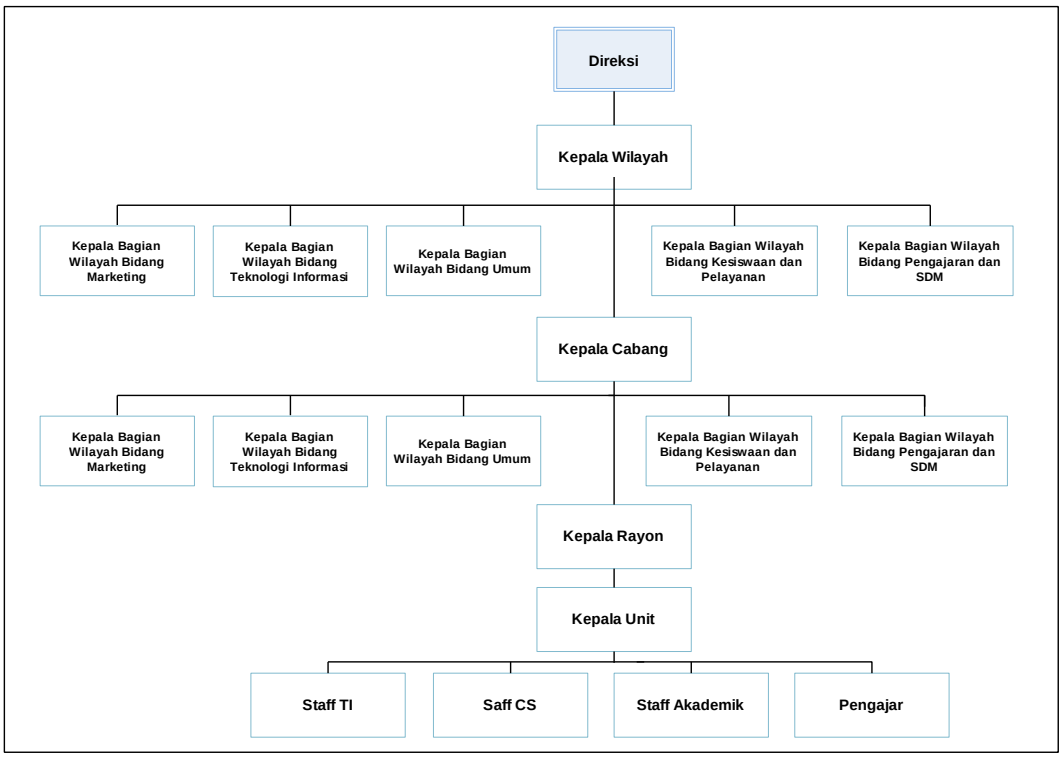
Direktorat Perencanaan dan Umum membawahi enam bidang yang masing-masing dipimpin oleh seorang manajer. Keenam bidang tersebut adalah :

1. Bidang Akuntansi
2. Bidang Keuangan
3. Bidang Operasional Pengajar
4. Bidang Sumber Daya Manusia
5. Bidang Teknologi Informasi
6. Bidang Diklat dan Prestasi Siswa



Gambar 3.2
Struktur Organisasi di pusat

Di masing-masing wilayah, terdapat seorang kepala wilayah yang dibantu oleh kepala bagian wilayah bidang terkait. Di masing-masing cabang yang membawahkan Kepala Wilayah dipimpin oleh Kepala Cabang. Kepala Cabang dibantu oleh kepala bagian cabang bidang terkait. Kepala Cabang membawahi beberapa kota yang dipimpin oleh Kepala Rayon, sedangkan Kepala Rayon membawahi Kepala Unit yang dilengkapi dengan staff seperti staff akademik, staff *customer service*, dan staff pendukung teknologi informasi. Gambar 3.3 memperlihatkan struktur organisasi di wlayah, cabang, dan unit.



Gambar 3.3
Struktur Organisasi Wilayah, Cabang, Rayon, dan Unit

3.1.6 Arsitektur *Database* Terdistribusi

Koneksi antar cabang di Ganesha Operation menggunakan *virtual private network* (VPN) berbasis OpenVPN dan menggunakan MySQL *Cluster*. Pembagian *replica set* mempertimbangkan masalah geografis agar mudah pengelolaanya. Selain itu dipilih kota dengan transportasi yang mudah dijangkau, kemudahan mencari penyedia jasa internet, dan kemampuan staf TI di cabang tersebut. Tabel 3.1 memperlihatkan pengelompokan *replica set* tersebut.

Tabel 3.1 Letak Node

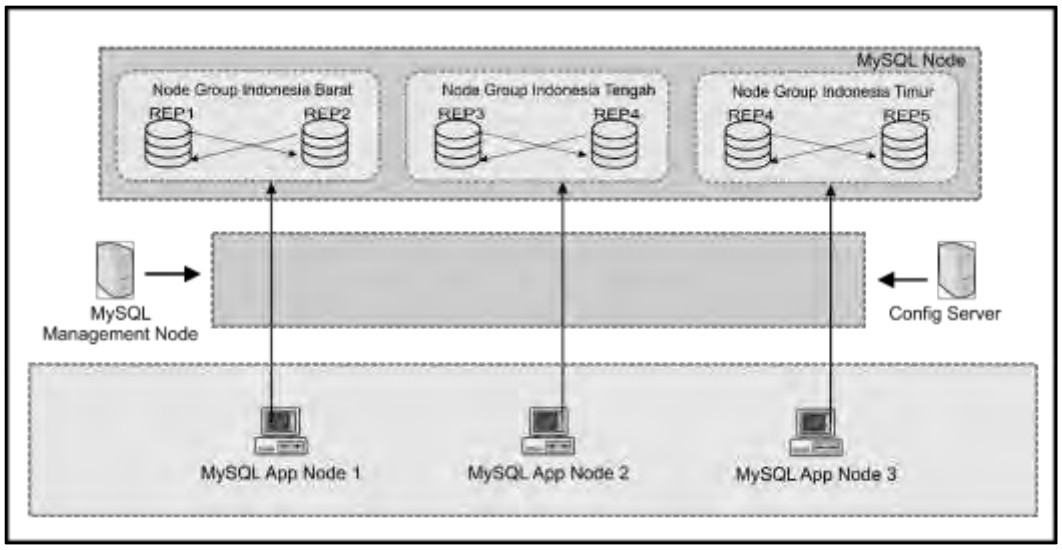
No	Kelompok Node	Anggota Node	Letak Node
1	Indonesia Bagian Barat	Replika 1	Medan, Sumut
		Replika 2	Jakarta, DKI
2	Indonesia Bagian Tengah	Replika 3	Bandung, Jabar
		Replika 4	Semarang, Jateng
3	Indonesia Bagian Timur	Replika 5	Surabaya, Jatim
		Replika 6	Denpasar, Bali

Gambar 3.4
Peta *Database* Terdistribusi di Ganesha Operation

Pada Gambar 3.4 memperlihatkan letak replikasi dan distribusi replikasi set tersebut. Dari gambar tersebut Replika 1 bertindak sebagai replikasi primer untuk daerah Sumatera bagian utara dan penyimpanan partisi sekunder untuk daerah Sumatera bagian selatan, Banten, dan Lampung dan juga sebaliknya. Replika 3 dan Replika 4 berpasangan dengan fungsi yang sama dengan Replika 1 dan Replika 2, begitu juga dengan Replika 5 dan Replika 6.

Selain merencanakan replikasi juga digunakan juga node management dan config server. Gambar 3.4 memperlihatkan replikasi dan node management tersebut. Untuk menghubungkan keseluruhan replika tersebut digunakan koneksi internet dengan lebar pita yang cukup (*bandwidth*). Setiap replika di sebuah kota ditempatkan di sebuah penyelenggara jasa internet dengan *colocation*.

Koneksi antar *node* tersebut harus memperhatikan masalah keamanan agar tidak tembus oleh pengguna yang tidak berwenang. Faktor keamanan tersebut dipenuhi dengan menggunakan koneksi *virtual private network* (VPN). VPN yang digunakan memanfaatkan koneksi intrenet yang ada yaitu menggunakan OpenVPN yang bersifat terbuka dan cuma-cuma. Server VPN dipasang dikantor pusat Bandung dan semua VPN *client*. Server OpenVPN ditempatkan di kota Bandung dan setiap client akan mendapatkan nomor IP yang digunakan sebagai koneksi antar *node*. Gambar 3.5 memperlihatkan Node pada MySQL *Cluster* di Ganesha Operation

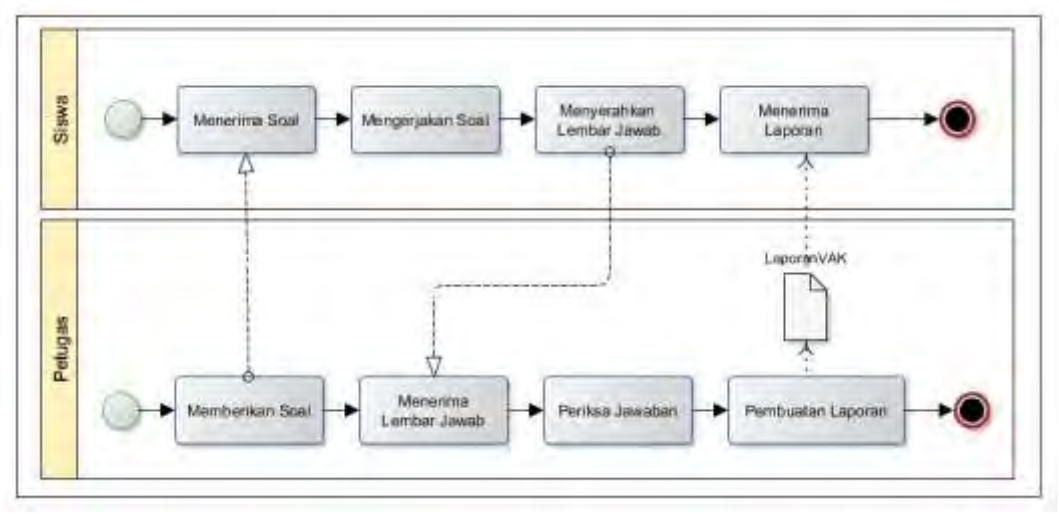


Gambar 3.5
Node MySQL *Cluster* Ganesha Operation

3.2 Proses Bisnis Di Beberapa Bidang

3.2.1 Bidang Operasional Pengajaran (BOP)

Setelah siswa telah melakukan pembayaran, siswa akan melakukan test VAK (Visual, *Auditory*, dan Kinestetis) dan test *assessment* dengan waktu yang telah disepakati oleh siswa dan petugas. Test VAK adalah test untuk mengetahui tipe pembelajaran siswa sehingga pemberian materi pelajaran lebih efektif dan konsultasi belajar bisa diarahkan sesuai dengan tipe pembelajaran siswa. Proses bisnis yang dilakukan untuk test VAK diperlihatkan pada Gambar 3.6.



Gambar 3.6
Proses Bisnis Pelaksanaan Test VAK

Setelah siswa menyelesaikan test VAK, petugas akan memeriksa hasil test tersebut. Hasil test akan disimpan dan siswa akan mendapatkan laporan test VAK. Informasi yang ada di laporan VAK diperlihatkan pada Tabel 3.2.

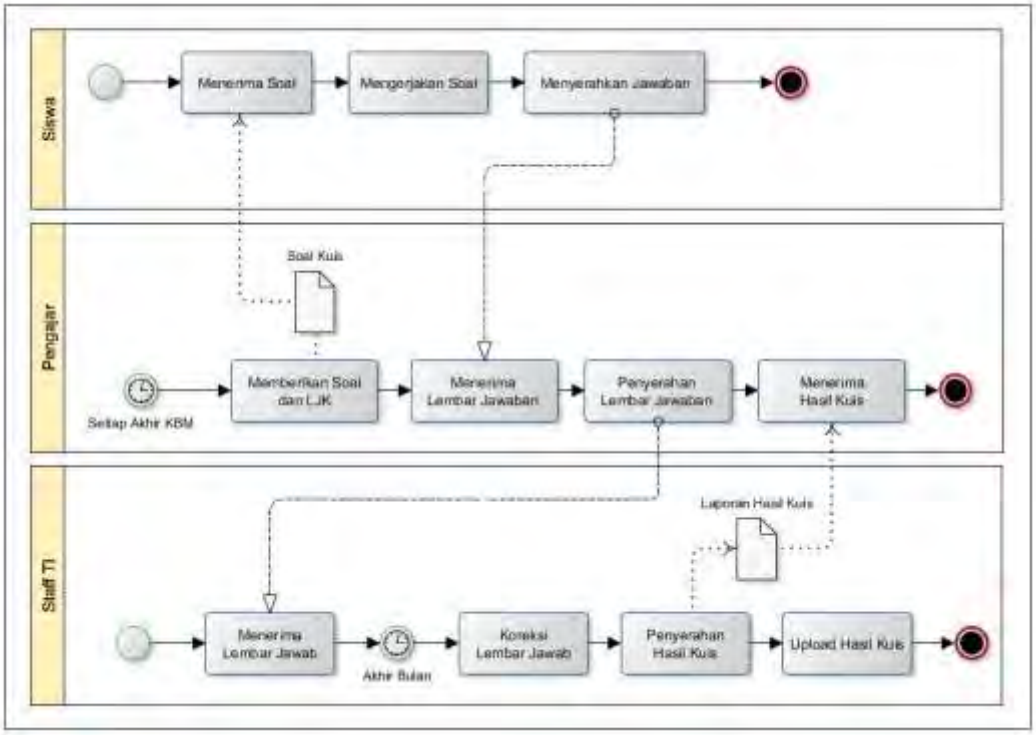
Tabel 3.2 Informasi pada laporan VAK

No	Informasi	Keterangan
1	NIS	Nomor Induk Siswa
2	Nama Siswa	Nama Siswa
3	Kelas	Kelas Siswa
4	Nilai Test	Berisi informasi nilai test untuk masing-masing modalitas. Nilai test dinyatakan dalam angka
5	Analisis belajar	Berisi analisis teknik belajar yang cocok untuk siswa.

3.2.2 Bidang Diklat dan Pelayanan Siswa (BDPS)

3.2.2.1 Proses Pelaksanaan Kuis

Pelaksanaan kuis dilakukan khusus untuk kelas minor yaitu 4SD, 5SD, 7SMP, 8SMP, 10SMA, dan 11SMA. Pelaksanaanya setiap akhir kegiatan belajar mengajar (KBM), pengajar memberikan 10 soal yang dikerjakan oleh siswa, bagi siswa yang tidak hadir pada saat pelaksanaan kuis tidak diberikan kuis susulan. Proses bisnis pelaksanaan kuis diperlihatkan pada Gambar 3.7.



Gambar 3.7
Proses Bisnis Pelaksanaan Kuis

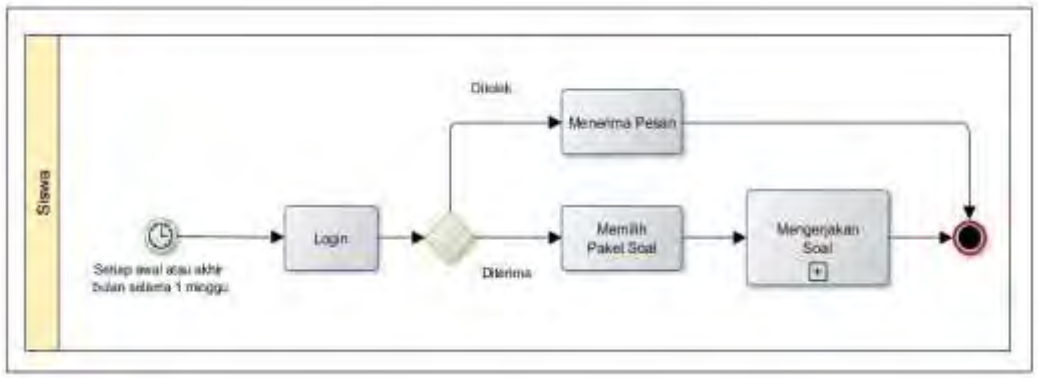
Pada akhir kegiatan belajar mengajar (KBM) siswa melaksanakan kuis harian yang hasilnya diberikan kepada staff TI untuk dikoreksi dan memberikan laporan serta mengupload hasil kuis tersebut. Laporan inilah yang menjadi dasar pengajar untuk menganalisa apakah siswa tersebut sudah menguasasi materi pelajaran atau belum. Informasi yang ada pada laporan kuis digambarkan pada Tabel 3.3

Tabel 3.3 Informasi pada laporan Kuis

No	Informasi	Keterangan
1	NIS	Nomor Induk Siswa
2	Nama Siswa	Nama Siswa
3	Kelas	Kelas Siswa
4	NIP	Nomor induk pengajar yaitu nomor unik yang merupakan identifikasi pengajar
5	Nama Pengajar	Nama pengajar yang bersangkutan
6	Tanggal Kuis	Tanggal dilaksanakannya kuis
7	Mata Pelajaran	Mata pelajaran yang diujikan pada saat kuis
8	Bab	Bab dari mata pelajaran yang diujikan pada saat kuis
9	Pertemuan	Pertemuan KBM pada saat kuis
10	Nilai Kuis	Berisi nilai kuis untuk masing-masing mata pelajaran dan bab.

3.2.2.2 Proses Kegiatan *Tryout Computer Based Test* (TO CBT)

Setiap awal atau akhir bulan seluruh siswa Ganesha Operation melaksanakan *tryout computer based test* dalam periode satu minggu dengan pola sesuai dengan tingkatkan kelas dan silabus dari bidang materi pelajaran. Pelaksanaan TO CBT bersifat online dimana waktu dan tempat pelaksanaan siswa menentukan sendiri. Proses bisnis TO CBT diperlihatkan pada Gambar 3.8



Gambar 3.8
Proses Bisnis Pelaksanaan TO CBT

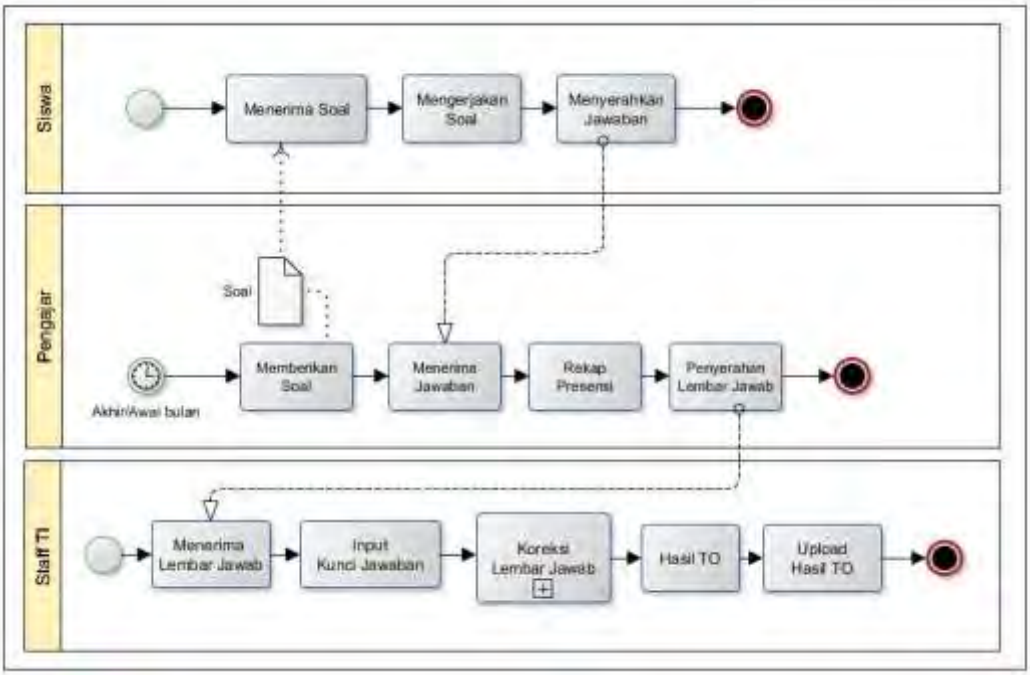
Informasi yang ada pada saat pelaksanaan *tryout computer based test* diperlihatkan pada Tabel 3.4

Tabel 3.4 Informasi pada TO CBT

No	Informasi	Keterangan
1	NIS	Nomor Induk Siswa
2	Nama Siswa	Nama Siswa
3	Kelas	Kelas Siswa
4	Tanggal TOCBT	Tanggal dilaksanakannya TOCBT
5	Kode Soal	Informasi kode soal yang mengidentifikasikan pola TOCBT yang dilaksanakan.
6	Nilai TOCBT	Berisi nilai TOCBT yang informasi di dalam terdapat Benar, Salah, Kosong untuk setiap nomor soal pada paket soal yang dikerjakan.

3.2.2.3 Proses Kegiatan *Tryout Paper Based Test* (TO PBT)

Pelaksanaan TO PBT dilaksanakan oleh seluruh siswa Ganesha Operation setelah periode TO CBT selesai pada setiap awal atau akhir bulan. Proses bisnis pelaksanaan TO PBT diperlihatkan pada Gambar 3.9



Gambar 3.9
Proses Bisnis Pelaksanaan TO PBT

Hasil *tryout* memberikan informas tentang hasil prestasi belajar siswa selama 1 bulannya. Adapun informasi yang akan disebarkan oleh staff TI diperlihatkan pada Tabel 3.5.

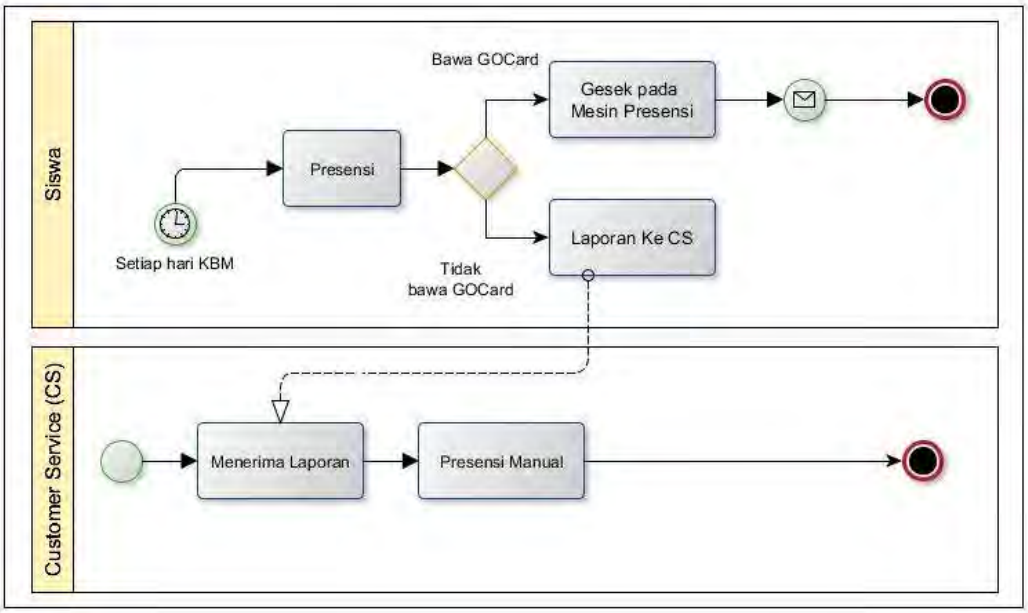
Tabel 3.5 Informasi hasil TO PBT

No	Informasi	Keterangan
1	NIS	Nomor Induk Siswa
2	Nama Siswa	Nama Siswa
3	Kelas	Kelas Siswa
4	Tanggal TOCBT	Tanggal dilaksanakannya TOPBT
5	Kode Soal	Informasi kode soal yang mengidentifikasikan pola TOPBT yang dilaksanakan.
6	Nilai TOPBT	Berisi nilai TOCBT yang informasi di dalam terdapat Benar, Salah, Kosong untuk setiap nomor soal pada paket soal yang dikerjakan.

3.2.2.4 Proses Presensi Siswa

Di Ganesha Operation siswa untuk setiap kedatangan untuk melaksanakan kegiatan bimbingan belajar wajib untuk melakukan presensi. Presensi digunakan dengan menggunakan kartu *magnetic* yang juga digunakan sebagai tanda pengenal sebagai siswa

Ganesha Operation dengan cara mengesekan pada mesin yang sudah disediakan. Proses lengkap dari presensi diperlihatkan pada Gambar 3.10.



Gambar 3.10
Proses Presensi Siswa

Dalam presensi siswa terdapat informasi yang diperlihatkan pada Tabel 3.6.

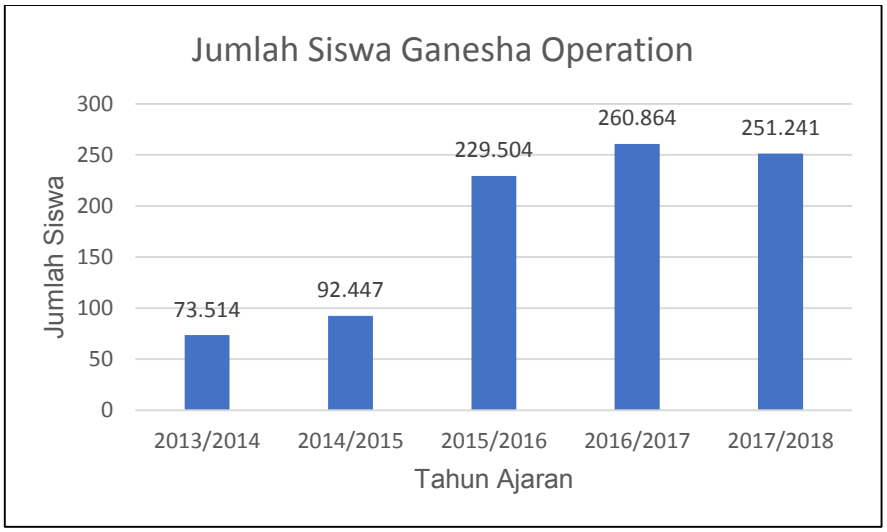
Tabel 3.6 Informasi Presensi

No	Informasi	Keterangan
1	NIS	Nomor Induk Siswa
2	Nama Siswa	Nama Siswa
3	Tanggal Presensi	Kelas Siswa
4	Jam ke	Informasi jam ke berapa siswa melakukan presensi
5	Jam Presensi	Informasi mengenai waktu (jam) pada saat siswa mengesekan kartu GOcard
6	Status	Berisi nilai otomatis dan manual, manual jika siswa lupa membawa kartu GOCard

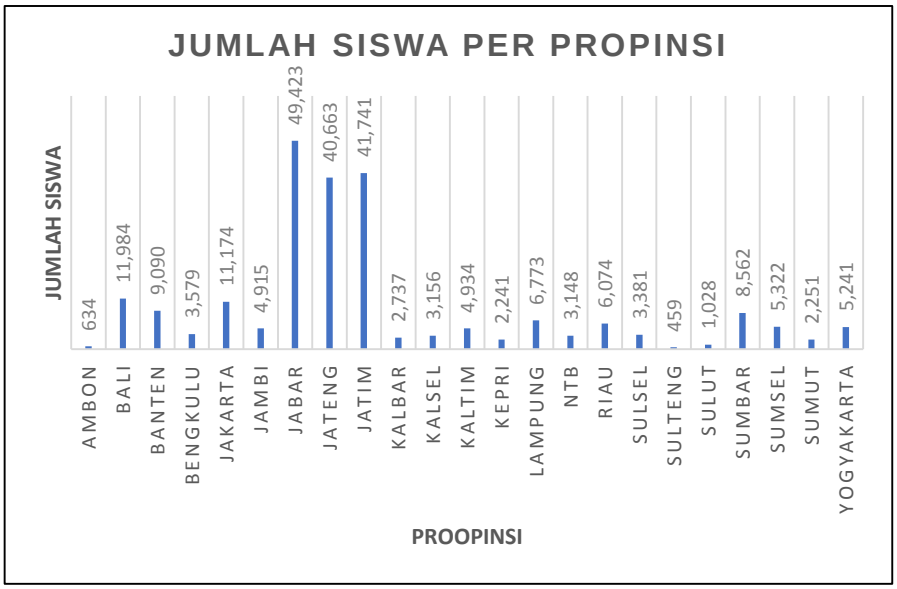
3.3 Penyebaran Jumlah Siswa

Dari tahun ke tahun Ganesha Operation mengalami perkembangan dengan ditunjukan jumlah siswa yang setiap tahun terus bertambah. Gambar 3.11 memberikan ilustrasi jumlah siswa selama lima tahun terakhir. Dibandingkan tahun 2016/2017, jumlah siswa tahun ajaran 2017/2018 terlihat menurun karena tahun ajaran 2017/2018 masih

berlangsung dan belum selesai ketika penelitian ini dibuat. Gambar 3.12 memperlihatkan distribusi jumlah siswa per propinsi.



Gambar 3.11
Grafik pertambahan jumlah siswa dari tahun ke tahun

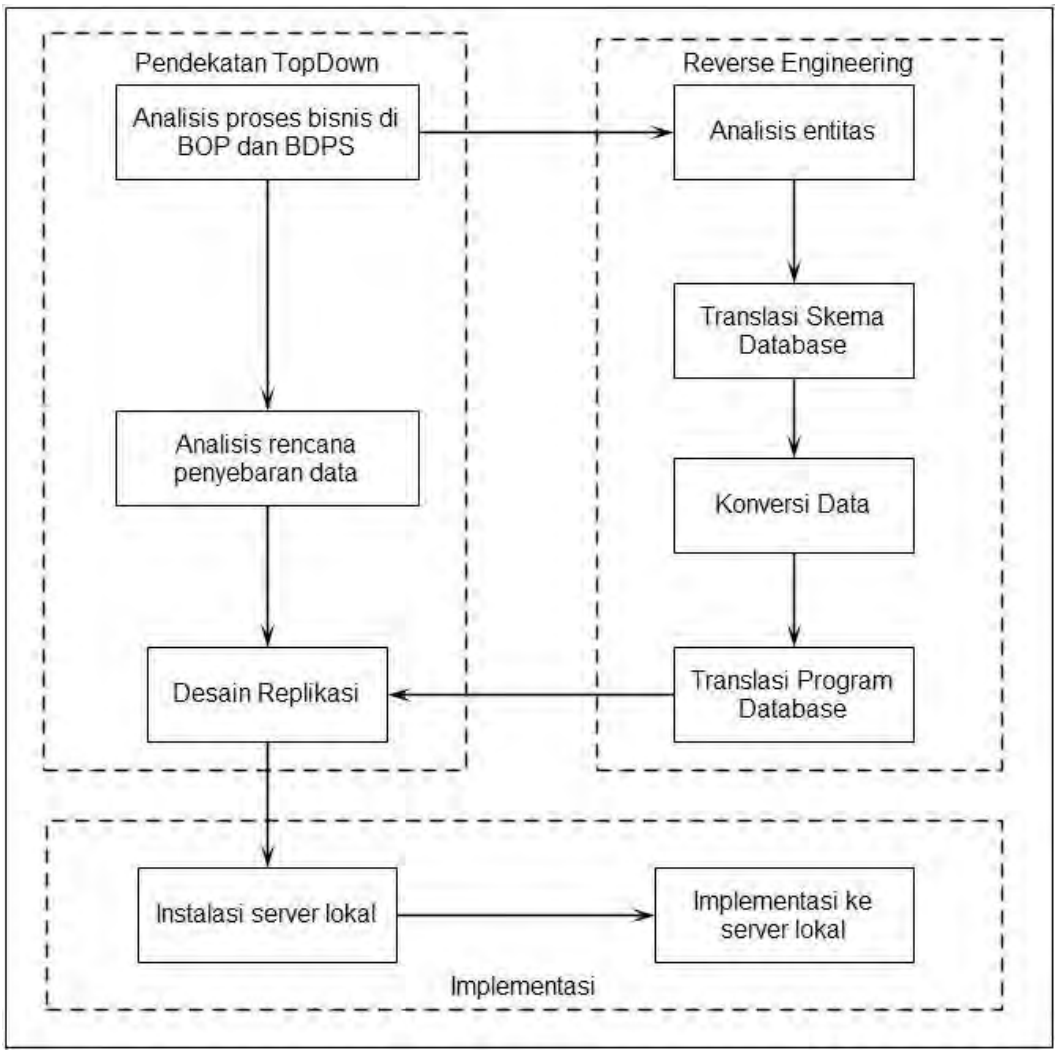


Gambar 3.12
Grafik distribusi jumlah siswa per Propinsi

3.4 Metodologi Pembangunan NoSQL *Database*

Metodologi yang digunakan untuk mengembangkan NoSQL *basisdata* adalah metodologi rekayasa ulang (*Reverse Engineering Database*) yang sudah dijelaskan pada

bab sebelumnya dengan perubahan sesuai kebutuhan. Gambar 3.13 memperlihatkan alur kerja yang dipakai untuk mengembangkan NoSQL *database* ini.



Gambar 3.13
Kerangka Kerja Pengembangan NoSQL *Database*

3.4.1 Analisis Kebutuhan Proses Bisnis

Langkah pertama dalam metodologi ini adalah analisis kebutuhan data. Analisis dilakukan di tiga bidang utama yang ada di objek penelitian yaitu Marketing, BOP, dan BDPS hanya proses terkait dengan sistem akademik yang akan dianalisis pada setiap bidang terkait tersebut. Analisis kebutuhan data ini didasarkan pada informasi yang telah diuraikan pada awal bab ini.

3.4.2 Analisis Penyebaran Data

Hasil analisis kebutuhan *database* dan informasi objek tentang letak cabang digunakan sebagai analisis rencana penyebaran data akan dilakukan dengan memperlihatkan biaya, jumlah siswa dan geografis. Meskipun data yang ada di *server* lokal tidak hanya data siswa namun jumlah data yang lain, sehingga analisis beban *server* akan didasarkan pada jumlah siswa yang akan ditangani *server*.

Analisa biaya juga didasarkan dari harga koneksi yang ada di cabang-cabang objek penelitian. Letak *server* lokal akan dipilih di kota-kota besar dimana harga jauh lebih murah dibandingkan dengan kota-kota kecil.

3.4.3 Analisis Skema *Database*

Sebagai bahan analisis adalah data-data yang dibutuhkan dalam proses bisnis yang terjadi. Ada beberapa entitas yang diperlukan yang tidak tercantum pada proses bisnis namun juga diperlukan karena berkaitan dengan struktur organisasi, struktur wilayah, dan letak geografis. Hal tersebut harus menjadi perhatian karena sangat penting untuk membangun entitas lain.

Tahap berikutnya adalah menganalisa hubungan antar entitas, hubungan antar entitas harus dianalisis sedetail mungkin. Pengembangan translasi skema relational *database* menjadi model data berorientasi dokumen dilakukan dengan memanfaatkan struktur dan relasi antar tabel yang ada di dalam skema yang sudah ada sebagai parameter utama pembentukan model yang dikembangkan dengan melakukan rekayasa ulang.

3.4.4 Proses Migrasi Data

Langkah selanjutnya adalah desain penyebaran data. Di tahap ini, akan dibuat desain data fisik yang kebutuhannya adalah dianalisis pada tahap sebelumnya. Dalam tahap desain fisik akan dibuat *database* dan dokumen-dokumen yang dibutuhkan oleh objek penelitian. Setelah desain fisik selesai, akan dibuatkan replikasi berdasarkan dokumen yang sudah dibuat pada tahap ini.

3.4.5 Implementasi

Langkah selanjutnya adalah implementasi rancangan yang sudah dilakukan. Implementasi diawali dengan instalasi server dengan Ubuntu Server 16.04, MongoDB dan openvpn. Akan digunakan tiga buah server sebagai model dengan koneksi yang sama dengan koneksi di objek penelitian mengingat selama ini objek penelitian menggunakan koneksi openvpn untuk koneksi dari pusat ke cabang atau sebaliknya.

Setelah instalasi maka desain fisik *database* yang sudah dirancang diaplikasikan pada *server* yang sudah dipilih.

BAB IV

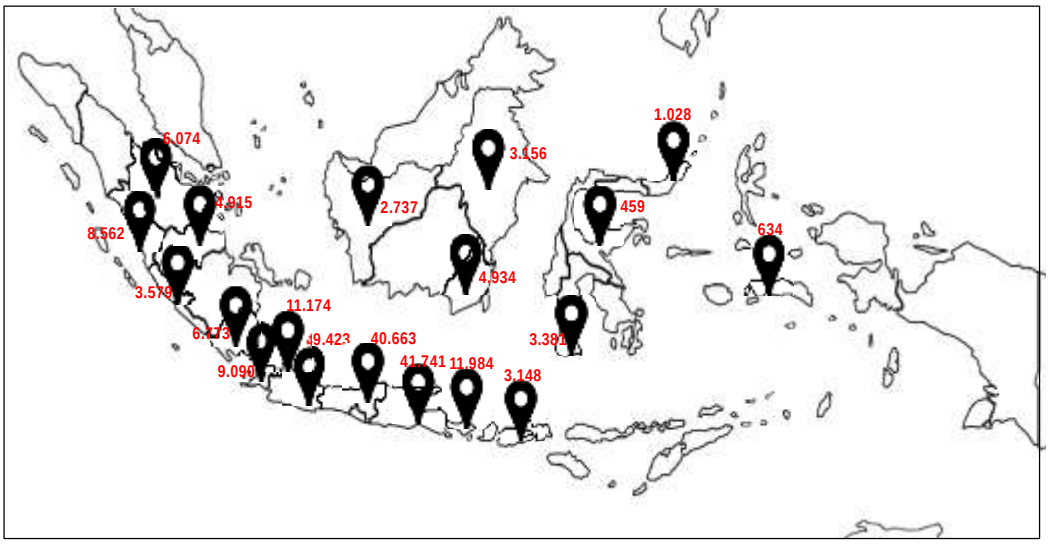
ANALISIS DAN DESAIN

4.1 Rencana Penyebaran Data

Salah satu metode analisis penyebaran data adalah analisis biaya. Analisis biaya harus memperhitungkan biaya koneksi, biaya pengadaan server, dan biaya operasional server. Biaya tersebut sebanding dengan jumlah situs dan pengguna yang akan dibuat. Semakin banyak situs yang dibuat semakin mahal biaya tersebut namun akan semakin cepat waktu tanggap yang dirasakan oleh *client.* Semakin sedikit situs yang dibuat akan semakin murah, namun waktu tanggap akses data yang dirasakan *client* akan semakin lama sehingga tidak memuaskan *client*. Hal ini tidak sesuai dengan tujuan penelitian ini yang ingin mempercepat akses sehingga *client* senang menggunakan aplikasi yang mengambil data dari server sebagai fasilitas yang diterima *client*.

Untuk itu dianalisis jumlah situs yang akan dibuat berdasarkan jumlah siswa dan jumlah cabang. Jika server menggunakan *random access memory* (RAM) sebesar 32GB dan sekitar 12,5% digunakan untuk mysql dan mongoDB sedangkan sisanya untuk aplikasi yang lain maka memory sebesar 4,5 GB mongoDB server bisa diakses oleh 600 koneksi secara bersamaan (www.mysqlcalculator.com) . Jika dalam satu kantor ada lima orang yang mengakses secara bersamaan, maka akan ada 120 kantor yang bisa ditangani oleh satu server. Dengan jumlah kantor seluruh Indonesia 752 maka dibutuhkan 6 s.d 7 server.

Pertimbangan berikutnya adalah jumlah siswa yang disimpan dalam satu server dan letak geografis dari wilayah yang ditangani. Gambar 4.1 memperlihatkan distribusi jumlah siswa per propinsi.



Gambar 4.1
Distribusi Jumlah Siswa

Dengan jumlah siswa sebanyak 250.000 siswa maka setiap situs akan optimum jika melayani sebanyak 36.000 sd 45.000 siswa.

Dengan memperlihatkan jumlah siswa, jumlah situs, dan jumlah koneksi akan ditangani oleh server maka letak server dan daerah yang ditangani dapat dilihat pada Tabel 4.1, dengan demikian dirancang *database* terdistribusi untuk enam buah situs untuk menangani siswa rata-rata 31.000 sd 49.500 siswa.

Tabel 4.1 Situs yang akan dibuat dan jumlah siswa yang ditangani

KOTA	SITUS1	SITUS2	SITUS3	SITUS4	SITUS5	SITUS6
AMBON						634
BALI						11.984
BANTEN		9.090				
BENGKULU	3.579					
JABAR			49.423			
JAKARTA		11.174				
JAMBI	4.915					
JATENG				40.663		
JATIM					41.741	
KALBAR						2.737
KALSEL						3.156
KALTIM						4.934
KEPRI	2.241					
LAMPUNG		6.773				
NTB						3.148

KOTA	SITUS1	SITUS2	SITUS3	SITUS4	SITUS5	SITUS6
RIAU	6.074					
SULSEL						3.381
SULTENG						459
SULUT						1.028
SUMBAR	8.562					
SUMSEL		5.322				
SUMUT	2.251					
YOGYA				5.241		
Jumlah	27.622	32.359	49.423	45.904	41.741	31.461

4.2 Analisis Kebutuhan Data Berdasarkan Rakayasa Ulang *Database*

Untuk menentukan kebutuhan data digunakan acuan *database* rekayasa terbalik untuk metode pengembangannya. Pengembangan proses transformasi skema *database* relasional menjadi model data dokumen pada mongoDB dilakukan dengan memanfaatkan struktur dan relasi antar tabel yang ada didalam skema. Tidak semua skema dalam sistem berjalan mengingat dalam tesis ini hanya dikhususkan pada bidang akademik saja.

4.2.1 Penentuan Entitas

Dari proses bisnis yang ada di tiga bidang utama dapat didaftarkan entitas yang sangat diperlukan untuk membangun dokumen pada MongoDB. Semua data siswa, pengajar, karyawan, dan kegiatan akademik lainnya serta yang berhubungan letak geografis diperlihatkan pada Tabel 4.2

Tabel 4.2 Entitas yang berhubungan dengan struktur Kewilayahan

No.	Entitas	Penjelasan
1	Gedung	Gedung dimana siswa belajar
2.	Unit	Unit adalah struktur terkecil dalam organisasi ini , Unit tak lain merupakan outlet. Sebuah unit ditangani oleh Kepala Unit
3.	Rayon	Rayon terdiri dari paling sedikit satu unit dan dikepalai oleh seorang Kepala Rayon. Sebuah rayon berasosiasi dengan kota dalam struktur kewilayahan..
4.	Cabang	Cabang terdiri dari beberapa rayon dan dikepalai oleh seorang Kepala Cabang.
5.	Wilayah	Wilayah merupakan sekumpulan beberapa cabang dan dikepalai oleh Kepala Wilayah.

Banyak entitas yang berkaitan erat dengan geografis Indonesia sehingga di daftarkan entitas yang berkaitan dengan geografis. Tabel 4.3 menggambarkan entitas yang berkaitan erat dengan geografis Indonesia.

Tabel 4.3 Entitas yang Berkaitan Dengan Struktur Wilayah Indonesia

No.	Entitas	Penjelasan
1.	Kota	Merupakan entitas yang sama dengan Daerah tingkat di Indonesia
2.	Propinsi	Merupakan entitas yang sama dengan Daerah tingkat di Indonesia

Dalam bidang marketing berkaitan dengan pelanggan (siswa) yang membeli produk dalam dunia pendidikan sering dinamakan program. Produk yang dijual mempunyai spesifikasi dan fasilitas yang spesifik seperti tingkat kelas dan kelas yang dipilih sehingga kelas juga merupakan entitas yang harus didata. Setiap pelanggan akan memberikan uang untuk pembayaran produk yang dibeli sehingga pembayaran juga merupakan entitas yang penting. Entitas yang berkaitan dengan bidang marketing diperlihatkan pada Tabel 4.4.

Tabel 4.4 Entitas yang berkaitan dengan bidang marketing

No.	Entitas	Penjelasan
1.	Siswa	Merupakan pelanggan yang pada tahun ajaran berlangsung membeli produk/program.
2.	Kelas	Merupakan kelas dimana siswa ditempatkan.
3.	Sekolah	Asal sekolah dari siswa.
4.	Pembayaran	Pembayaran oleh siswa karena membeli produk/program.
5.	Tingkat Kelas	Tingkat kelas produk tersebut, diperlukan apakah siswa tersebut XII IPA, XIIIPS, dll.
6.	Bundling	Merupakan daftar program yang dijual beserta dengan fasilitas yang akan diterima.

Dalam bidang SDM entitas yang perlu didata adalah entitas berkaitan dengan karyawan non pengajar dan karyawan pengajar. Entitas bidang SDM diperlihatkan pada Tabel 4.5.

Tabel 4.5 Entitas yang berkaitan dengan bidang SDM

No.	Entitas	Penjelasan
1.	Karyawan	Karyawan Ganesha Operation.
2.	Pengajar	Entitas Karyawan dengan tugas khusus mengajar.

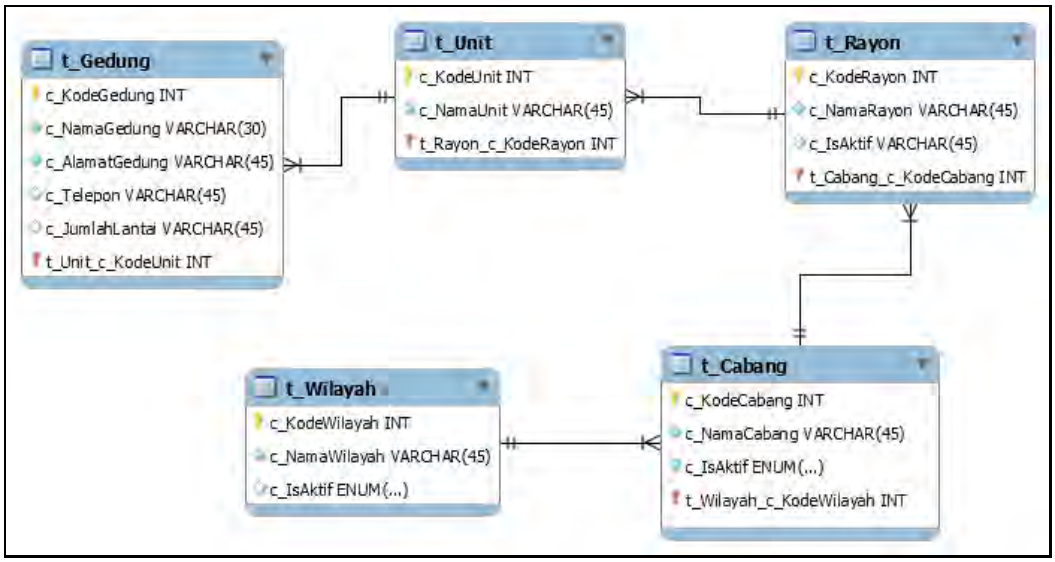
4.2.2 Penentuan Relasi Antar Entitas

Entity Relational Diagram (ERD) yang dirancang adalah ERD yang berkaitan dengan Akademik, Marketing, dan Sumber Daya Manusia (SDM). Tidak semua tabel yang dibutuhkan oleh ketiga bidang tersebut dibuat. Di bidang marketing akan dibuat tabel yang berhubungkan dengan data siswa, asal sekolah, tingkat kelas dan pembelian produk. Di bidang Sumber Daya Manusia (SDM) tabel yang dibuat karyawan non pengajar dan karyawan pengajar sedangkan pada bidang Akademik akan dibuatkan keseluruhan yang berhubungan sistem informasi akademik di Ganesha Operation. Namun ada hubungan yang terlibat pada ketiga bidang tersebut yaitu ERD yang berkaitan dengan geografis dan struktural kewilayahan Ganesha Operation.

4.2.2.1 Translasi Skema Geografis

Proses bisnis di beberapa bidang yang ada di objek penelitian tidak semua kebutuhan tabel dijelaskan pada bab tersebut. Hanya tabel-tabel utama yang menjadi perhatian utama sehingga penambahan tabel-tabel lain di kemudian hari menjadi lebih mudah. Kebutuhan data yang ada menjadi dasar pembentukan dokumen data di dalam *database* yang akan dirancang.

Semua data yang dibutuhkan pada geografis yaitu unit, rayon, cabang, dan wilayah. Tabel yang akan dibangun yang pertama adalah tabel yang berkaitan erat dengan geografis tersebut. Gambar 4.2 memperlihatkan ERD yang berkaitan dengan geografis.



Gambar 4.2
Entity Relationship Diagram Kelompok Geografis

Tabel gedung menyimpan informasi gedung seperti nama gedung, alamat, jumlah lantai, status gedung, dan lain-lain. Status gedung menjelaskan gedung tersebut masih aktif atau sudah tidak digunakan lagi. Relasi antara gedung dan unit adalah relasi “satu ke banyak” karena satu hanya ada di satu gedung dan satu gedung paling sedikit mempunyai satu unit.

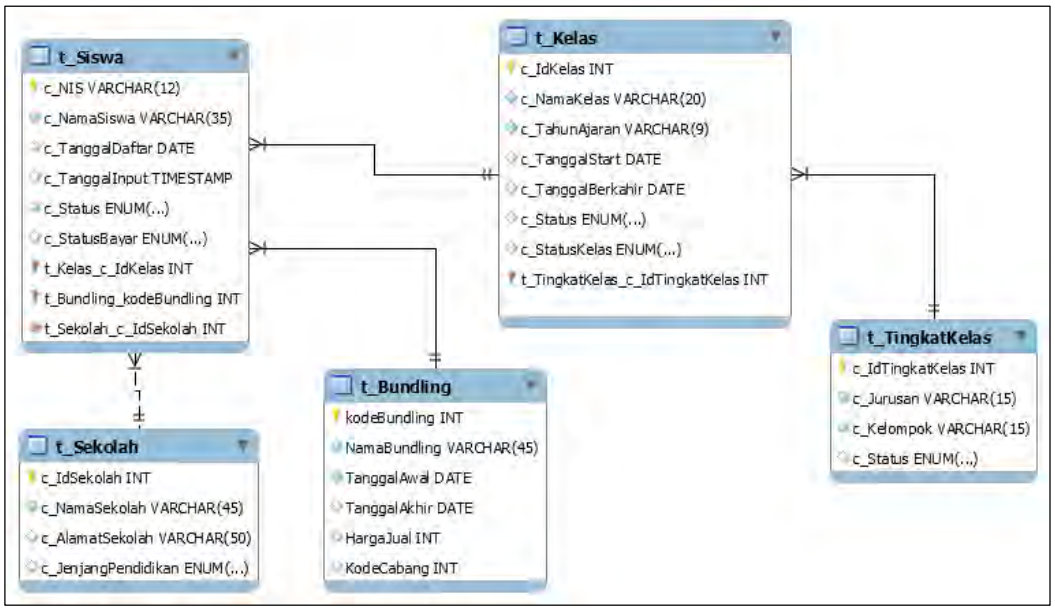
Tabel unit adalah tabel yang merekam informasi kantor terkecil di PT Ganesha Operation. Di dalam tabel ini informasi yang direkam adalah nama unit, tanggal awal unit berdiri dan tanggal akhir unit tidak aktif. Sebuah unit bisa saja tidak aktif lagi karena sesuatu hal.

4.2.2.2 Translasi Skema Bidang Marketing

ERD bidang marketing berkaitan erat dengan produk dan pendaftaran siswa. Karena program belajar diciptakan untuk dijual di masing-masing unit. Maka ERD bidang marketing akan banyak berkaitan dengan ERD Geografis. Gambar 4.3 menjelaskan tabel yang berkaitan erat antara bidang marketing dengan bidang akademik.

Tabel siswa adalah tabel yang mendata pelanggan yang membeli produk di Ganesha Operation. Jika pelanggan membeli produk maka ia dinamakan siswa dan

disimpan dalam tabel siswa. Tabel siswa merekam atribut siswa yang berubah bergantung saat membeli produk tersebut misalkan asal sekolah, kelas, dsb.

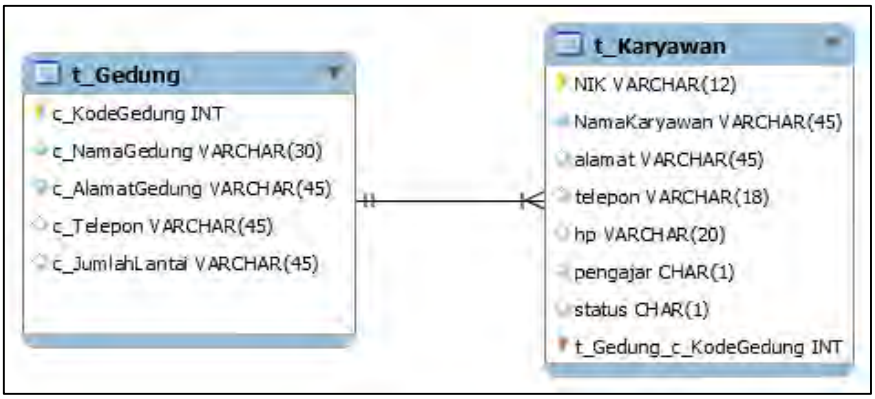


Gambar 4.3
Entity Relationship Diagram Kelompok Marketing

Ada beberapa tabel yang berkaitan dengan kondisi saat pelanggan membeli produk yaitu tabel sekolah, tabel kelas penempatan siswa, dan tabel tingkat kelas yang merupakan tingkat dari siswa tersebut.

4.2.2.3 Translasi Skema Bidang SDM

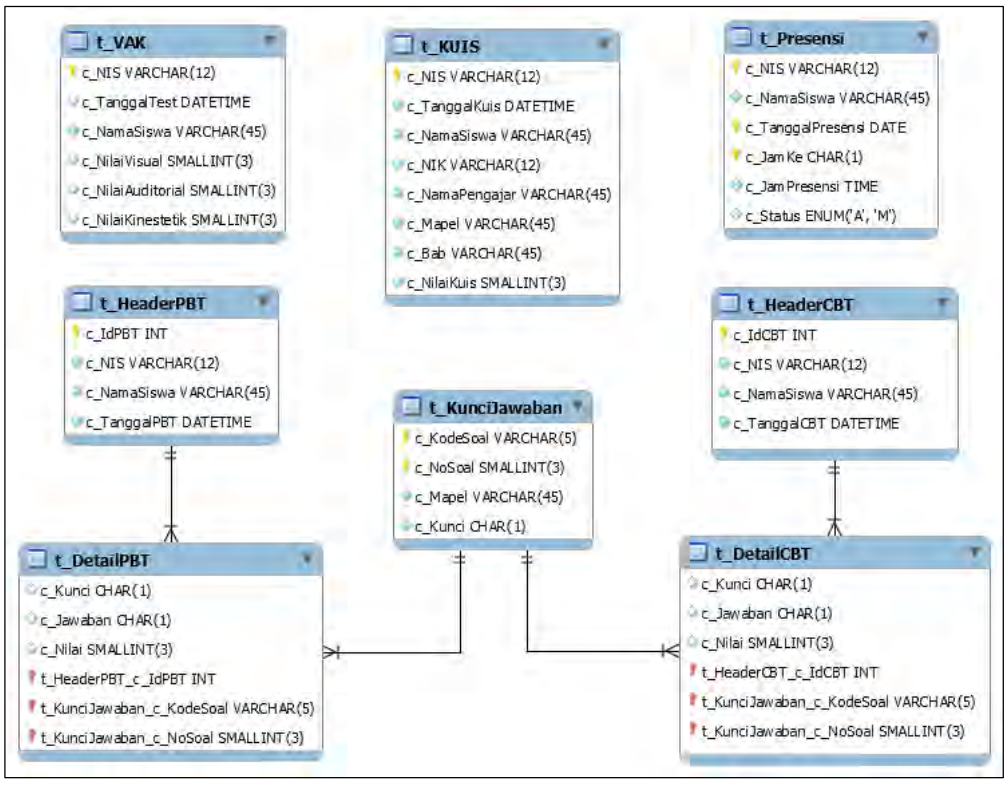
ERD bidang SDM berkaitan erat dengan pengajar, karyawan dan penempatan di gedung yang ada di Ganesha Operation. Karena setiap pengajar pasti akan ditempatkan di suatu gedung tertentu walaupun zona kerja bisa lebih dari satu gedung. Maka ERD bidang SDM akan banyak berkaitan dengan ERD Geografis. Gambar 4.4 menjelaskan tabel yang berkaitan erat antara bidang SDM dengan bidang Geografis.



Gambar 4.4
Entity Relationship Diagram Kelompok SDM

4.2.2.4 Translasi Skema Bidang Akademik

Tabel yang berkaitan dengan bidang akademik banyak berkaitan dengan fasilitas yang diberikan selama siswa mengikuti kegiatan belajar mengajar (KBM) seperti kuis, tes modalitas, dan tryout. Gambar 4.5 menggambarkan relasi antar tabel tersebut.



Gambar 4.5
Entity Relationship Diagram Kelompok Akademik

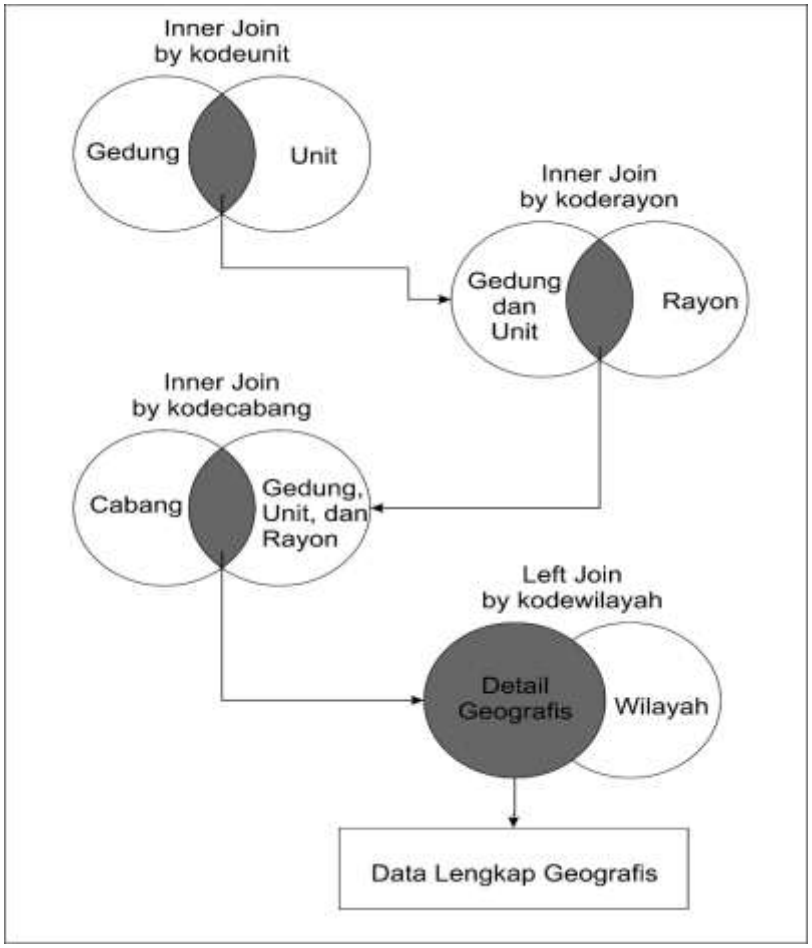
4.3 Proses Konversi Data

Dari ERD yang sudah dirancang pada sub bab sebelumnya, dapat dirancang data model dari dokumen MongoDB. Bentuk data model dokumen tidak jauh berbeda dengan ERD tersebut. Pembentukan model dokumen ini adalah mengubah semua *tuple* data yang ada pada setiap tabel di *database* relasional menjadi dokumen-dokumen pada MongoDB. Dokumen yang dibentuk dapat berupa representasi dari sebuah *tuple* atau gabungan beberapa *tuple*.

Pada *database* relasional sudah ditentukan jenis data dari masing-masing atribut entitas. Juga dibuatkan *foreign key* dari sebuah tuple karena ada relasi dari tabel yang lain. Jumlah, tipe data, dan nilai kardinalitas pada suatu tabel sebagai parameter pembentukan dokumen pada MongoDB. Model dokumen yang terbentuk dari proses konversi data ini dapat berupa dokumen tunggal, *embedded document*, *referenced document* ataupun kombinasi ketiganya.

4.3.1 Konversi Data Kelompok Geografis

Dikelompok geografis Tabel Cabang merekam data daftar cabang dan berelasi dengan Tabel Wilayah dimana kode wilayah tidak dimiliki oleh semua cabang. Pada tabel diberikan nilai “1” jika ada cabang yang tidak memiliki wilayah, hal ini membutuhkan banyak kolom untuk menampung data yang belum tentu diisi semua (tidak efisien). Relasi antar tabel pada kelompok geografis akan menggunakan *embedded document* karena dari jumlah data yang disisipkan tidak berpotensi untuk menjadi besar dan kedalaman dari *embedded document* juga tidak lebih dari 100. Berdasarkan ERD yang berkaitan dengan geografis maka dibuatkan struktur relasi untuk membuat informasi yang lengkap tentang geografis menggunakan data *query*. Gambar 4.6 memperlihatkan struktur gabungan tabel sebagai informasi geografis lengkap.



Gambar 4.6
Struktur Gabungan Tabel sebagai Informasi Geografis Lengkap

Data yang disimpan dalam dokumen diwakili oleh kumpulan dokumen masing-masing. Dokumen JSON akan dibuat berdasarkan informasi lengkap dari geografis yang akan disimpan dengan koleksi MongoDB sebagai dokumen JSON. Setiap dokumen geografis terdiri dari himpunan bagian dari objek mewakili dokumen JSON individu yang disarangkan kedalam dokumen geografis yang menyertakan unit, rayon, cabang, dan wilayah. Gambar 4.7 memperlihatkan skema hasil yang diusulkan untuk koleksi berorientasi dokumen pada MongoDB.

```
{
    _id             : ObjectID ,
    kodegedung : ..... ,
    namagedung : ..... ,
    alamat          : ..... ,
    telepon         : ..... ,
    status          : ..... ,
    eUnit           : [
          {
              kodeunit : ..... ,
              namaunit : .....
    } ],
    eRayon          : [
          {
              koderayon : ..... ,
              namarayon : .....
    } ],
    eCabang         : [
          {
              kodecabang : .... ,
              namacabang: ...
    } ],
    eWilayah        : [
          {
              kodewilayah: .... ,
              namawilayah: ....
    } ]
}
```

```
{
    _id             : ObjectID ,
    kodegedung : ..... ,
    namagedung : ..... ,
    alamat          : ..... ,
    telepon         : ..... ,
    status          : ..... ,
    eUnit           : [
          {
              kodeunit : ..... ,
              namaunit : .....
    } ],
    eRayon          : [
          {
              koderayon : ..... ,
              namarayon : .....
    } ],
    eCabang         : [
          {
              kodecabang : .... ,
              namacabang: ...
    } ]
}
```

Gambar 4.7
Collection Document Oriented Geografis Pada NoSQL *Database*

Pada Gambar 4.7 dapat dilihat untuk satu *collection* dapat mempunyai *field* yang berbeda, ini berguna untuk jumlah *memory* yang dibutuhkan untuk menyimpan *collection*. Selain itu juga jumlah tabel atau collection pada MongoDB menjadi lebih efisien.

Untuk mencocokan data yang ada di tabel asal dengan data yang ada pada dokumen hasil dilakukan dengan membandingkan hasil pengaksesan data yang ada di tabel asal dengan menjalankan *query* data dalam format sebagai berikut :

```
“SELECT
t_Gedung.c_KodeGedung,t_Gedung.c_NamaGedung,
t_Unit.c_NamaUnit,t_Rayon.c_NamaRayon,
t_Cabang.c_NamaCabang,t_Wilayah.c_NamaWilayah
FROM
t_Gedung
INNER JOIN t_Unit ON t_Gedung.c_KodeUnit = t_Unit.c_KodeUnit
INNER JOIN t_Rayon ON t_Unit.c_KodeRayon = t_Rayon.c_kodeRayon
INNER JOIN t_Cabang ON t_Rayon.c_KodeCabang = t_Cabang.c_KodeCabang
```

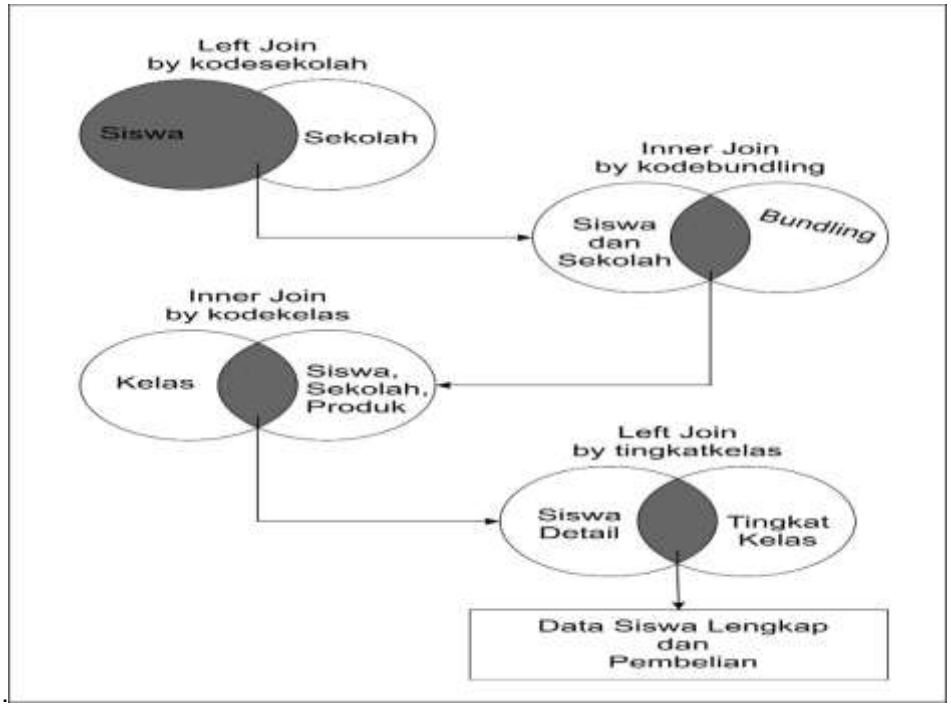
INNER JOIN t_Wilayah ON t_Cabang.c_KodeWilayah = t_Wilayah.c_KodeWilayah”

Sedangkan pengaksesan data yang terdapat pada dokumen mongodb dapat dilakukan dengan menjalankan *query* dalam format sebagai berikut :

```
db.geografis.find({},
{"kodegedung":1,"namagedung":1,"eUnit.namaunit":1,
"eRayon.namarayon":1,"eCabang.namacabang":1,
"eWilayah.namawilayah":1})
```

4.3.2 Konversi Data Kelompok Marketing

Pada kelompok marketing, entitas asosiasi yang penting adalah tabel siswa. Tabel t_siswa ini merupakan entitas asosiasi yang muncul karena hubungan antar tabel t_siswa dengan tabel t_bundling. Tabel t_bundling berisi daftar nama produk yang akan dijual disuatu daerah, sedangkan tabel t_siswa merupakan penduduk yang membeli produk dan kemudian menjadi siswa. Dalam tabel t_siswa direkam atribut seseorang yang mungkin berubah ketika ia membeli produk baru tersebut yaitu asal sekolah, tingkat kelas, kelasgo, dan tanggal daftar. Gambar 4.8 memperlihatkan struktur gabungan tabel sebagai informasi siswa lengkap.



Gambar 4.8
Struktur Gabungan Tabel sebagai Informasi Siswa Lengkap

Relasi antar tabel pada kelompok marketing menggunakan *embedded document* dan *referenced document* karena kedalaman hanya mencapai tiga dan tipe kardinalitasnya yaitu partial referenced dimana data pada tabel t_sekolah diacu oleh tabel t_siswa dan tabel t_sekolah tidak semua data yang ada pada tabel t_sekolah diacu oleh data yang ada pada tabel t_siswa. Gambar 4.9 memperlihatkan skema dokumen pada NoSQL *Database*

```
{
    _id             : ObjectID ,
    NIS : ..... ,
    namasiswa : ..... ,
    tkelas          : ..... ,
    kelago          : ..... ,
    tgldaftar       : ..... ,
    tglinput        : ..... ,
    status          : ..... ,
    eBundling       : [{ namabundling : ..... , harga : .....  }],
    kodegedung : {
                        "$ref"    : "geografis",
                        "$id"     : ObjectId,
                        "$db"     : "akademik"
                    },
    eSekolah        : [{ idsekolah : ....., namasekolah :....}]
}
```

Gambar 4.9
Collection Kelompok Marketing Pada NoSQL *Database*

4.3.3 Konversi Data Kelompok SDM

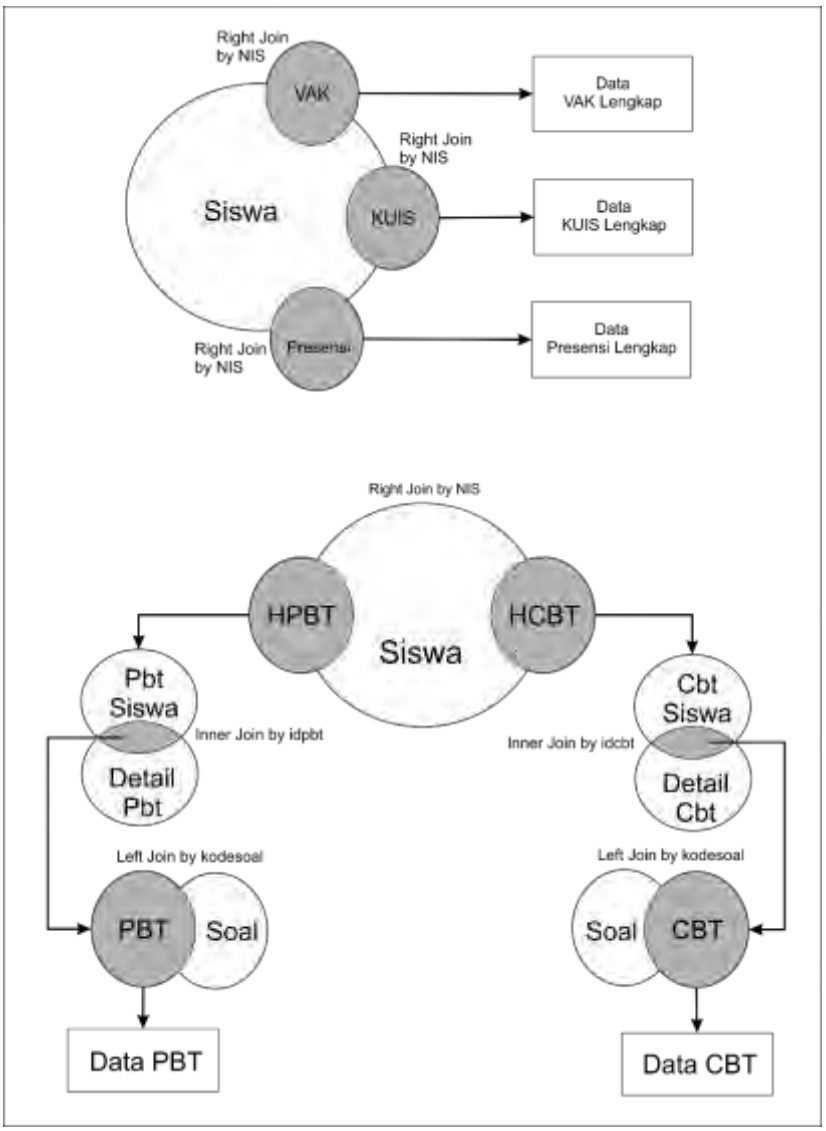
Dikelompok karyawan (SDM) akan digunakan satu entitas yaitu tabel t_karyawan dan menggunakan *referenced document* yang tidak memiliki sub dokumen dan akan menjadi dokumen tunggal yang diperlihatkan pada Gambar 4.10.

```
{
    _id                 : ObjectID ,
    NIK                 : ..... ,
    namakaryawan        : ..... ,
    kodegedung          : ..... ,
    alamat              : ..... ,
    telepon             : ..... ,
    hp                  : ..... ,
    pengajar            : ..... ,
    status              : .....
}
```

Gambar 4.10
Collection Kelompok SDM Pada NoSQL *Database*

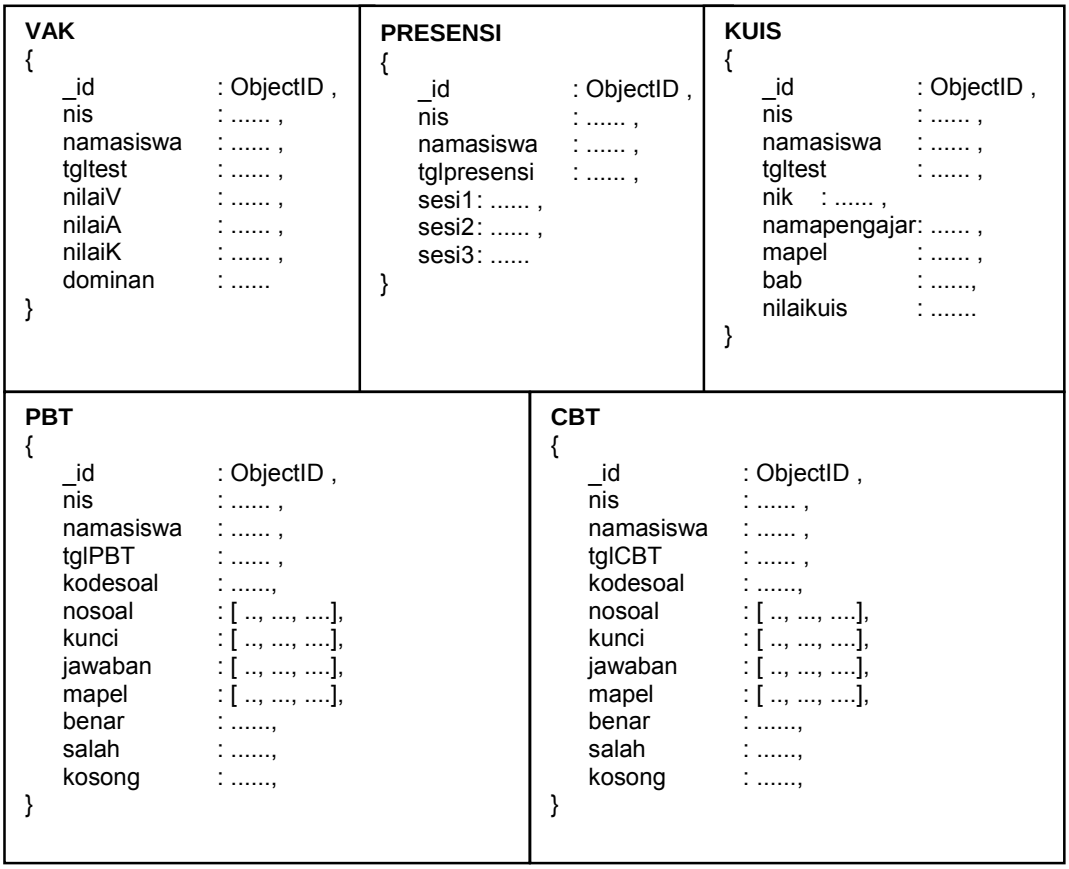
4.3.4 Konversi Data Kelompok Akademik

Dikelompok akademik semua entitas akan menjadi dokumen tunggal dengan nilai kardinalitas *full referenced* dan digabungkan dengan *referenced document* untuk *collection* mengacu pada siswa melalui NIS. Gambar 4.11 memperlihatkan gabungan tabel lengkap pada kelompok akademik.Hasil transformasi pada kelompok akademik diperlihatkan pada gambar 4.11.



Gambar 4.11
Struktur Gabungan Tabel sebagai Informasi Akademik Lengkap

Dikelompok akademik akan menggunakan lima *collection* dan menggunakan *referenced document* yang tidak memiliki sub dokumen dan akan menjadi dokumen tunggal yang diperlihatkan pada Gambar 4.12.



Gambar 4.12
Collection Kelompok Akademik Pada NoSQL *Database*

4.3.5 Desain Model Data

Setiap tipe data pada MySQL harus dimodelkan sesuai dengan tipe data yang ada pada MongoDB. Pemodelan ini dilakukan dengan mentranslasikan tipe data MySQL menjadi tipe data pada MongoDB. Mekanisme translasi ini memperhatikan spesifikasi masing-masing tipe data agar pemodelan yang dilakukan dapat mempertahankan konsistensi data. Daftar translasi tipe data pada migrasi data yang dibangun ditunjukan pada tabel 4.6.

Tabel 4.6 Daftar translasi tipe data

Kategori	Tipe Data MySQL	Tipe Data MongoDB
Numeric	Bigint	NumberLong()
	Bit	String
	Decimal	String
	Double	Numeric
	Float	Numeric
	Int	NumberLong()
	Mediumint	NumberInt()
String	Binary	BinData()
	Blob	BinData()
	Char	String
	Enum	String
	Longblob	BinData()
	Longtext	String
Data and Time	Date	String
	Datetime	String
	Timestamp	ISODate()
	Year	string

Dalam tesis ini dokumen ada yang memang dirancang sama berdasarkan *database* sebelumnya dan ada dokumen yang merupakan perbaikan dari tabel sebelumnya di lokasi objek penelitian. Tabel 4.7 memperlihatkan perubahan tabel-tabel fisik menjadi dokumen pada MongoDB di tempat objek penelitian.

Tabel 4.7 Daftar *collection* yang ada, diubah, dan diperbaruhi

No	Nama Collection	Penjelasan	Perubahan
1	cGeografis	Merupakan tabel asosiasi antara tabel gedung, unit, rayon, cabang, dan wilayah pada *database* sebelumnya yang digabungkan menjadi satu *collection* dengan menggunakan *embedded document*	Diubah
2	cKaryawan	*Collection* data karyawan	Sudah ada
3	cSiswa	Merupakan tabel asosiasi antara tabel ssiwa, sekolah, kelas, dan tingkat kelas pada *database* sebelumnya yang digabungkan menjadi satu *collection* dengan menggunakan *embedded* dan *referenced document*. Pada *collection* ini juga di tambahkan satu field baru yaitu password.	Diubah
4	cVAK	*Collection* data modalitas	Sudah ada
5	cPresensi	*Collection* data presensi siswa	Sudah ada
6	cKuis	*Collection* data kegitan Kuis	Sudah ada
7	cPBT	*Collection* data tryout dengan menggunakan lembar jawab komputer	Diubah
8	cCBT	*Collection* data tryout yang dilakukan secara *online*	Diubah

4.4 Translasi Program *Database*

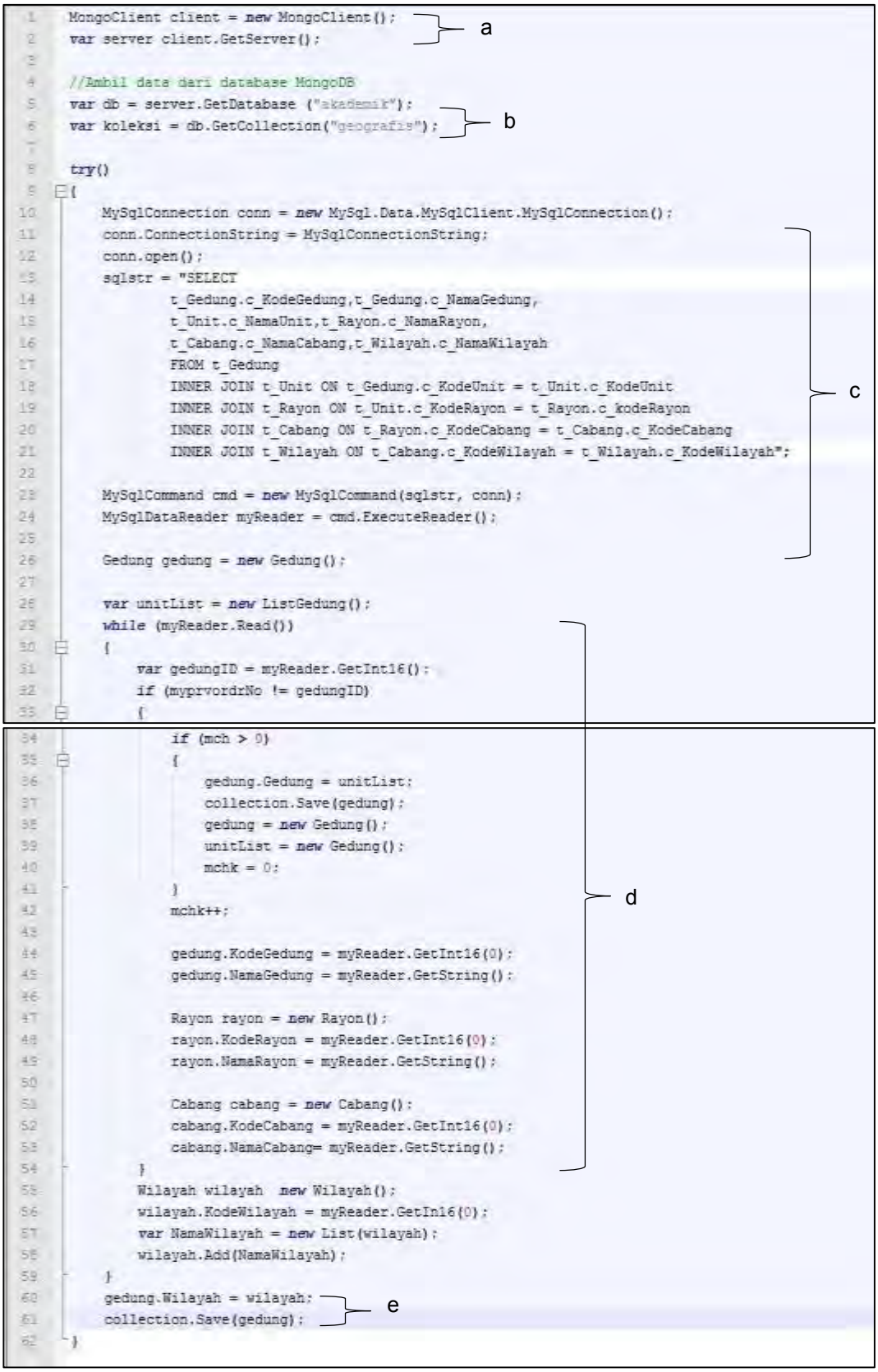
Tahap berikutnya adalah berdasarkan hasil konversi data ke dalam skema dokumen MongoDB, maka perlu dilakukan cara agar data yang ada pada *database* relational dapat dimigrasikan ke NoSQL *database* MongoDB, pada saat ini beberapa *tool* ETL (*Extract Transform Load*) seperti mongify dan pentaho telah memiliki fitur untuk melakukan migrasi dari database relational ke MongoDB, akan tetapi fitur-fitur tersebut hanya mengakomodasi proses perpindahan data.

Data yang ada pada *database* relational akan diexktraksi sedemikian rupa sesuai dengan format yang ada agar dapat diubah menjadi bentuk dokumen pada MongoDB. Untuk dokumen tunggal yang berasal dari satu tabel dapat langsung dilakukan proses migrasi datanya, tetapi untuk dokumen yang merupakan gabungan dari beberapa tabel harus dibuatkan terlebih dahulu sebelum data siap dipindahkan.

Berdasarkan permasalahan ini maka, perlu dibuatkan algoritma dalam satu *platform* translasi program *database* menggunakan bahasa pemograman php dan javascript sehingga meningkatkan efisiensi sumber daya dan waktu yang dibutuhkan pada proses migrasi *database*.

4.4.1 Koding Untuk Migrasi Data Kelompok Geografis

Sesuai dengan hasil konversi data didapatkan bahwa *collection* geografis merupakan gabungan dari beberapa tabel pada *database* relational. Langkah ini akan mewakili pengkodean yang mencakup mendapatkan dan membuat data sebagai dokumen JSON. Gambar 4.13 memperlihatkan pengkodean untuk translasi program *database* pada kelompok geografis.



```
MongoClient client = new MongoClient();
var server client.GetServer();

//Ambil data dari database MongoDB
var db = server.GetDatabase ("akademik");
var koleksi = db.GetCollection("geografis");

try()
{
    MySqlConnection conn = new MySql.Data.MySqlClient.MySqlConnection();
    conn.ConnectionString = MySqlConnectionString;
    conn.open();
    sqlstr = "SELECT
            t_Gedung.c_KodeGedung,t_Gedung.c_NamaGedung,
            t_Unit.c_NamaUnit,t_Rayon.c_NamaRayon,
            t_Cabang.c_NamaCabang,t_Wilayah.c_NamaWilayah
            FROM t_Gedung
            INNER JOIN t_Unit ON t_Gedung.c_KodeUnit = t_Unit.c_KodeUnit
            INNER JOIN t_Rayon ON t_Unit.c_KodeRayon = t_Rayon.c_kodeRayon
            INNER JOIN t_Cabang ON t_Rayon.c_KodeCabang = t_Cabang.c_KodeCabang
            INNER JOIN t_Wilayah ON t_Cabang.c_KodeWilayah = t_Wilayah.c_KodeWilayah";

    MySqlCommand cmd = new MySqlCommand(sqlstr, conn);
    MySqlDataReader myReader = cmd.ExecuteReader();

    Gedung gedung = new Gedung();

    var unitList = new ListGedung();
    while (myReader.Read())
    {
        var gedungID = myReader.GetInt16();
        if (myprvordrNo != gedungID)
        {
            if (mch > 0)
            {
                gedung.Gedung = unitList;
                collection.Save(gedung);
                gedung = new Gedung();
                unitList = new Gedung();
                mchk = 0;
            }
            mchk++;

            gedung.KodeGedung = myReader.GetInt16(0);
            gedung.NamaGedung = myReader.GetString();

            Rayon rayon = new Rayon();
            rayon.KodeRayon = myReader.GetInt16(0);
            rayon.NamaRayon = myReader.GetString();

            Cabang cabang = new Cabang();
            cabang.KodeCabang = myReader.GetInt16(0);
            cabang.NamaCabang= myReader.GetString();
        }
        Wilayah wilayah  new Wilayah();
        wilayah.KodeWilayah = myReader.GetIn16(0);
        var NamaWilayah = new List(wilayah);
        wilayah.Add(NamaWilayah);
    }
    gedung.Wilayah = wilayah;
    collection.Save(gedung);
}
```

a

b

c

d

e

Gambar 4.13
Koding Translasi Program *Database* Kelompok Geografis

Gambar 4.13 memperlihatkan bagian a menunjukan baris perintah untuk membuat objek koneksi ke *database* MongoDB, lalu bagian b dimuat nama *database* dan koleksi yang akan digunakan. Bagian c melakukan query pengambilan data dari MySQL untuk disimpan pada database MongoDB. Terakhir bagian d melakukan pengulangan dari data yang diambil yang selanjutnya disimpan pada koleksi terpilih yang diperlihatkan pada bagian e.

4.4.2 Koding Untuk Migrasi Data Kelompok Marketing

Sama dengan kelompok geografis, kelompok marketing juga merupakan gabungan dari beberapa tabel pada *database* relational. Gambar 4.14 memperlihatkan pengkodean untuk translasi program *database* pada kelompok marketing.

```
MongoClient client = new MongoClient();
var server client.GetServer();

var db = server.GetDatabase ("akademik");
var koleksi = db.GetCollection("siswa");

try()
{
        MySqlConnection conn = new MySql.Data.MySqlClient.MySqlConnection();
        conn.ConnectionString = MySqlConnectionString;
        conn.open();
        sqlstr = "SELECT
                t_Siswa.c_NIS,t_Siswa.c_NamaLengkap,t_Siswa.c_Status,
                t_Kelas.c_NamaKelas,t_Gedung.c_NamaGedung,
                t_Bundling.c_NamaBundling,t_Bundling.c_Harga
                FROM t_Siswa
                INNER JOIN t_Kelas ON t_Siswa.c_IdKelas = t_Kelas.c_IdKelas
                INNER JOIN t_Gedung ON t_Gedung.c_IdGedung =
                t_Gedung.c_IdGedung
                INNER JOIN t_Bundling ON t_Siswa.c_IdBundling =
                t_Bundling.c_IdBundling";

        MySqlCommand cmd = new MySqlCommand(sqlstr, conn);
        MySqlDataReader myReader = cmd.ExecuteReader();

        while (myReader.Read())
        {
                var kelasID = myReader.GetInt16();
                var gedungID = myReader.GetInt16();
                collection.Save.References(kelasID);
                collection.Save.Embedded(gedungID);

                Siswa siswa = new Siswa();
                siswa.nis = myReader.GetInt16(0);
                siswa.NamaLengkap = myReader.GetString();
```

```
                    siswa.Status = myReader.GetString();

                    Kelas kelas = new kelas();
                    kelas.NamaKelas = myReader.GetString();

                    Gedung gedung = new Gedung();
                    gedung.NamaGedung = myReader.GetString();

                    Bundling bundling  new Bundling();
                    bundling.Idbundling = myReader.GetIn16(0);
                    bundling.Harga = myReader.GetIn16()

                    var KoleksiSiswa = new List(koleksisiswa);
                    koleksisiswa.Add();
          }
          collection.Save(koleksisiswa);
}
```

Gambar 4.14
Koding Translasi Program *Database* Kelompok Marketing

4.4.3 Koding Untuk Migrasi Data Kelompok SDM

Pada kelompok sdm berdasarkan hasil konversi data pada dokumen akan menggunakan dokumen tunggal, maka untuk translasi program *database* di perlihatkan pada Gambar 4.15

```
Class Sdm {
      $db['default'] = array
           ( 'hostname' => 'localhost', 'username' => 'root',
             'password' => 'rakyatjelata1945', 'database' => 'db_GOIcons',
             'dbdriver' => 'mysqli'
          );                                                              a

   protected $mongo;
   protected $db;                                                         b
   protected $table;

   function __construct() {
      try {
         //Connect to Mongo
         $this->mongo = new Mongo('127.0.0.1:27017');
         $this->db = $this->mongo->selectDB('akademik');
         $tableName = 'karyawan';                                        c
         $this->table = $this->db->$tableName;
      } catch(Exception $e) {
         exit();
      }
                    $this->load->Model('sdm')
   }
```

```
    function GetKaryawan () {
            JSON_encode($this->sdm->getData());
    }

    function sdm_getData() {
        $sql = "SELECT * FROM v_konversiKaryawan";
        $data = $this->db->query($sql)->result_array();

        foreach ($data->result() as $rowKaryawan){
            $data["nik"] = $rowKaryawan->c_NIK;
            $data["namakaryawan"] = $rowKaryawan->c_NamaKaryawan;
            $data["kodegedung"] = $rowKaryawan->c_KodeGedung;
            $data["alamat"] = $rowKaryawan->c_Alamat;
            $data["hp"] = $rowKaryawan->c_HP;
            $data["pengajar"] = $rowKaryawan->c_IsPengajar;
            $data["status"] = $rowKaryawan->c_Status;
        }
        return $data;
    }
}
```

d

e

Gambar 4.15
Koding Translasi Program *Database* Kelompok SDM

Gambar 4.15 memperlihatkan bagian a menunjukan baris perintah untuk membuat objek koneksi ke *database* MySQL, lalu bagian b dimuat nama *database* dan koleksi yang akan digunakan. Bagian c melakukan koneksi ke *database* MongoDB, setelah berhasil terkoneksi ke *database* selanjutnya memanggil fungsi getData yang dijelaskan pada bagian d, bagian e melakukan query pengambilan data dari MySQL untuk disimpan dengan mengembalikan nilai berupa format JSON.

4.4.4 Koding Untuk Migrasi Data Kelompok Akademik

Pada kelompok akademik berdasarkan hasil konversi data pada dokumen akan menggunakan dokumen tunggal, maka untuk program *database* di akan dibuatkan pada masing-masing kegiatan yang ada pada kelompok akademik.

4.4.4.1 Koding Data VAK

Gambar 4.16 memperlihatkan baris perintah untuk translasi program *database*. bagian a menunjukan baris perintah untuk membuat objek koneksi ke *database* MySQL, lalu bagian b dimuat nama *database* dan koleksi yang akan digunakan. Bagian c melakukan koneksi ke *database* MongoDB, setelah berhasil terkoneksi ke database

selanjutnya memanggil fungsi getData yang dijelaskan pada bagian d, bagian e melakukan query pengambilan data dari MySQL untuk disimpan dengan mengembalikan nilai berupa format JSON.

```
Class Akademik {
      $db['default'] = array
           ( 'hostname' => 'localhost', 'username' => 'root',
             'password' => 'rakyatjelata1945', 'database' => 'db_GOIcons',
             'dbdriver' => 'mysqli'
           );                                                                    a

   protected $mongo;
   protected $db;
   protected $table;                                                             b

   function __construct() {
      try {
         //Connect to Mongo
         $this->mongo = new Mongo('127.0.0.1:27017');
         $this->db = $this->mongo->selectDB('akademik');
         $tableName = vak';
         $this->table = $this->db->$tableName;
      } catch(Exception $e) {
         echo "Something Went Wrong.";
         exit();
      }
                  $this->load->Model(akademik)
   }                                                                             c

   function GetVAK () {
         JSON_encode($this-> akademik ->getData());
   }                                                                             d

   function akademik _getData() {
      $sql = "SELECT * FROM v_konversiModalitas";
      $data = $this->db->query($sql)->result_array();

      foreach ($data->result() as $rowVAK){
         $data["nis"] = $rowVAK->c_NIS;
         $data["namasiswa"] = $rowVAK->c_NamaLengkap;
         $data["tgltest"] = $rowVAK->c_Tanggal;
         $data["nilaiV"] = $rowVAK->c_visual;
         $data["nilaiA"] = $rowVAK->c_auditorial
         $data["nilaiK"] = $rowVAK->c_kinestetik;
         $data["dominan"] = $rowVAK->c_dominan;
      }
      return $data;
   }                                                                             e
}
```

Gambar 4.16
Koding Translasi Program *Database* Kelompok Akademik (VAK)

4.4.4.2 Koding Data Presensi

Gambar 4.17 memperlihatkan baris perintah untuk translasi program *database*. bagian a menunjukan baris perintah untuk membuat objek koneksi ke *database* MySQL, lalu bagian b dimuat nama *database* dan koleksi yang akan digunakan. Bagian c melakukan koneksi ke *database* MongoDB, setelah berhasil terkoneksi ke *database* selanjutnya memanggil fungsi getData yang dijelaskan pada bagian d, bagian e melakukan query pengambilan data dari MySQL untuk disimpan dengan mengembalikan nilai berupa format JSON.

```
Class Akademik
{

    $db['default'] = array
          ( 'hostname' => 'localhost', 'username' => 'root',
            'password' => 'rakyatjelata1945', 'database' => 'db_GOIcons',
            'dbdriver' => 'mysqli'
          );

  protected $mongo;
  protected $db;
  protected $table;

  function __construct()
  {
     try
     {
        $this->mongo = new Mongo('127.0.0.1:27017');
        $this->db = $this->mongo->selectDB('akademik');
        $tableName = presensi';
        $this->table = $this->db->$tableName;
     } catch(Exception $e)
     {
        exit();
     }
                  $this->load->Model(akademik)
  }

  function GetPresensi ()
  {
        JSON_encode($this-> akademik ->getData());
  }

  function sdm_getData() {
     $sql = "SELECT * FROM v_konversiPresensi";
     $data = $this->db->query($sql)->result_array();

     foreach ($data->result() as $rowPresensi){
        $data["nis"] = $ rowPresensi ->c_NIS;
        $data["namasiswa"] = $ rowPresensi ->c_NamaLengkap;
```

a

b

c

d

e

```
            $data["tglpresensi"] = $ rowPresensi ->c_TanggalAbsen;
            $data["sesi1"] = $ rowPresensi ->c_Sesi1;
            $data["sesi2"] = $ rowPresensi ->c_Sesi2;
            $data["sesi3"] = $ rowPresensi ->c_Sesi3;
        }
        return $data;
    }
}
```

e

Gambar 4.17
Koding Translasi Program *Database* Kelompok Akademik (Presensi)

4.4.4.3 Koding Data Kuis

Gambar 4.18 memperlihatkan baris perintah untuk translasi program *database*. bagian a menunjukan baris perintah untuk membuat objek koneksi ke *database* MySQL, lalu bagian b dimuat nama *database* dan koleksi yang akan digunakan. Bagian c melakukan koneksi ke *database* MongoDB, setelah berhasil terkoneksi ke *database* selanjutnya memanggil fungsi getData yang dijelaskan pada bagian d, bagian e melakukan query pengambilan data dari MySQL untuk disimpan dengan mengembalikan nilai berupa format JSON.

```
Class Akademik {
        $db['default'] = array
            ( 'hostname' => 'localhost', 'username' => 'root',
              'password' => 'rakyatjelata1945', 'database' => 'db_GOIcons',
              'dbdriver' => 'mysqli'
            );

    protected $mongo;
    protected $db;
    protected $table;

    function __construct() {
        try {
            //Connect to Mongo
            $this->mongo = new Mongo('127.0.0.1:27017');
            $this->db = $this->mongo->selectDB('akademik');
            $tableName = kuis';
            $this->table = $this->db->$tableName;
        } catch(Exception $e) {
            echo "Something Went Wrong.";
            exit();
        }
                    $this->load->Model(akademik)
    }
```

a

b

c

```
    function GetKuis () {
            JSON_encode($this-> akademik ->getData());
    }

    function sdm_getData() {
        $sql = "SELECT * FROM v_konversiKuis";
        $data = $this->db->query($sql)->result_array();

        foreach ($data->result() as $rowKuis){
            $data["nis"] = $rowKuis ->c_NIS;
            $data["namasiswa"] = $ rowKuis ->c_NamaLengkap;
            $data["tglkuis"] = $ rowKuis ->c_TanggalKuis;
            $data["NIK"] = $ rowKuis ->c_NIK;
            $data["namapengajar"] = $ rowKuis ->c_NamaPengajar;
            $data["mapel"] = $ rowKuis ->c_Mapel;
            $data["bab"] = $ rowKuis ->c_BAB;
            $data["nilaikuis"] = $ rowKuis ->c_NilaiKuis;
        }
        return $data;
    }
}
```

d

e

Gambar 4.18
Koding Translasi Program *Database* Kelompok Akademik (Kuis)

4.4.4.4 Koding Data PBT

Gambar 4.19 memperlihatkan baris perintah untuk translasi program *database*. bagian a menunjukan baris perintah untuk membuat objek koneksi ke *database* MySQL, lalu bagian b dimuat nama *database* dan koleksi yang akan digunakan. Bagian c melakukan koneksi ke *database* MongoDB, setelah berhasil terkoneksi ke *database* selanjutnya memanggil fungsi getData yang dijelaskan pada bagian d, bagian e melakukan query pengambilan data dari MySQL untuk disimpan dengan mengembalikan nilai berupa format JSON.

```
Class Akademik
{
        $db['default'] = array
            ( 'hostname' => 'localhost', 'username' => 'root',
              'password' => 'rakyatjelata1945', 'database' => 'db_GOIcons',
              'dbdriver' => 'mysqli'
            );

    protected $mongo;
    protected $db;
    protected $table;
```

a

b

```
    function __construct() {
        try {
            //Connect to Mongo
            $this->mongo = new Mongo('127.0.0.1:27017');
            $this->db = $this->mongo->selectDB('akademik');
            $tableName = 'pbt';
            $this->table = $this->db->$tableName;
        } catch(Exception $e) {
            echo "Something Went Wrong.";
            exit();
        }
                    $this->load->Model(akademik)
    }

    function GetPBT () {
            JSON_encode($this-> akademik ->getData());
    }

    function sdm_getData() {
        $sql = "SELECT * FROM v_konversiPBT";
        $data = $this->db->query($sql)->result_array();

        $sqldetail = "SELECT nosoal,jawab,kunci,mapel FROM v_konversiPBTDetail";
        $dataDetail = $this->db->query($sqldetail)->result_array();

        foreach ($data->result() as $rowPBT){
            $data["nis"] = $rowPBT ->c_NIS;
            $data["namasiswa"] = $rowPBT ->c_NamaLengkap;
            $data["tglpbt"] = $rowPBT ->c_TanggalKuis;
            foreach ($dataDetail ->result() as $rowPBTDetail){
                $data["nosoal"] = $rowPBTDetail ->nosoal;
                $data["jawab"] = $rowPBTDetail ->jawab;
                $data["kunci"] = $rowPBTDetail ->kunci;
            }
            $data["benar"] = $rowPBT ->c_Benar;
            $data["salah"] = $rowPBT ->c_Salah;
            $data["kosong"] = $rowPBT ->c_Kosong;
        }
        return $data;
    }
}
```

c

d

e

Gambar 4.19
Koding Translasi Program *Database* Kelompok Akademik (PBT)

4.4.4.5 Koding Data CBT

Gambar 4.20 memperlihatkan baris perintah untuk translasi program *database*. bagian a menunjukan baris perintah untuk membuat objek koneksi ke *database* MySQL, lalu bagian b dimuat nama *database* dan koleksi yang akan digunakan. Bagian c melakukan koneksi ke *database* MongoDB, setelah berhasil terkoneksi ke *database*

selanjutnya memanggil fungsi getData yang dijelaskan pada bagian d, bagian e melakukan query pengambilan data dari MySQL untuk disimpan dengan mengembalikan nilai berupa format JSON.

```
Class Akademik
{
      $db['default'] = array
           ( 'hostname' => 'localhost', 'username' => 'root',                    a
             'password' => 'rakyatjelata1945', 'database' => 'db_GOIcons',
             'dbdriver' => 'mysqli'
          );

   protected $mongo;
   protected $db;                                                               b
   protected $table;

   function __construct()
    {
     try {
         //Connect to Mongo
         $this->mongo = new Mongo('127.0.0.1:27017');
         $this->db = $this->mongo->selectDB('akademik');
         $tableName = 'cbt';
         $this->table = $this->db->$tableName;                                  c
     } catch(Exception $e)
     {
         echo "Something Went Wrong.";
         exit();
     }
                    $this->load->Model(akademik)
   }

   function GetCBT ()
  {
          JSON_encode($this-> akademik ->getData());                            d
   }

   function sdm_getData()
   {
      $sql = "SELECT * FROM v_konversiCBT";
      $data = $this->db->query($sql)->result_array();

      $sqldetail = "SELECT nosoal,jawab,kunci,mapel FROM v_konversiCBTDetail";
      $dataDetail = $this->db->query($sqldetail)->result_array();

      foreach ($data->result() as $rowCBT)                                      e
     {
         $data["nis"] = $rowCBT ->c_NIS;
         $data["namasiswa"] = $rowCBT ->c_NamaLengkap;
         $data["tglpbt"] = $rowCBT ->c_TanggalKuis;
         foreach ($dataDetail ->result() as $rowCBTDetail){
           $data["nosoal"] = $rowCBTDetail ->nosoal;
           $data["jawab"] = $rowCBTDetail ->jawab;
           $data["kunci"] = $rowCBTDetail ->kunci;
```

```
            }
            $data["benar"] = $rowCBT ->c_Benar;
            $data["salah"] = $rowCBT ->c_Salah;
            $data["kosong"] = $rowCBT ->c_Kosong;
        }
        return $data;
    }
}
```

e

Gambar 4.20
Koding Translasi Program *Database* Kelompok Akademik (CBT)

4.5 Desain Replikasi

Dari rancangan yang ada didapatkan bahwa ada enam replika yang akan dibangun. Akan dihitung jumlah kelompok replikasi (*replica set*) dalam rancangan ini dan agar ketersediaan data tinggi (*high availability*) dan kinerja kecepatan tinggi yang dapat mengurangi *I/O* pada sistem MongoDB. Ini berarti ada dua replikasi di masing-masing *replica set* sehingga jumlah *replica set* adalah sejumlah tiga *replica set*.

Pembagian *replica set* harus memperhitungkan masalah geografis agar mudah pengelolaanya. Selain itu dipilih kota dengan transportasi yang mudah dijangkau, kemudahan mencari penyedia jasa internet, dan kemampuan staf TI di cabang tersebut. Tabel 4.8 memperlihatkan rencana pengelompokan *replica set* tersebut.

Tabel 4.8 Perencanaan Replikasi *Set*

No	Replica Set	Anggota Replica	Letak Replica
1	Indonesia Bagian Barat	Replika 1	Medan, Sumut
		Replika 2	Jakarta, DKI
2	Indonesia Bagian Tengah	Replika 3	Bandung, Jabar
		Replika 4	Semarang, Jateng
3	Indonesia Bagian Timur	Replika 5	Surabaya, Jatim
		Replika 6	Denpasar, Bali

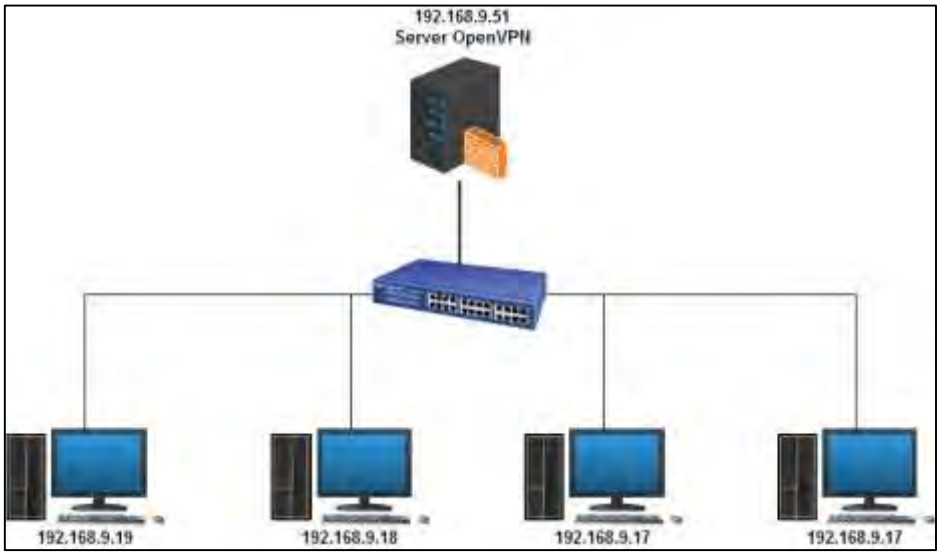
BAB V

IMPLEMENTASI DAN EVALUASI

5.1 Persiapan Instalasi Server

Koneksi antar cabang Ganesha Operation menggunakan *virtual private network* (VPN) berbasis openVPN. Agar simulasi rancangan NoSQL *database* MongoDB mendekati ke keadaan yang sebenarnya, maka simulasi dilakukan pada jaringan lokal berbasis openVPN.

Untuk membuat jaringan VPN digunakan jaringan lokal dengan satu komputer sebagai server openVPN dan empat lainnya sebagai *client*. Topologi jaringan yang digunakan diperlihatkan pada Gambar 5.1.



Gambar 5.1
Topologi Jaringan VPN PT Ganesha Operation

Pada komputer dipasang openVPN dengan sertifikat untuk masing-masing *client* harus dibuatkan terlebih dahulu di komputer server dan dikirimkan ke masing-masing *client*. Setelah konfigurasi selesai, alamat IP yang sesuai dengan konfigurasi alamat IP pada simulasi diperlihatkan pada Tabel 5.1. Lampiran 1 memperlihatkan ruangan dan server yang digunakan.

Tabel 5.1 Alamat IP openVPN

Alamat IP lokal	Alamat IP dari openVPN
192.168.9.51	10.10.10.1
192.168.9.18	10.10.10.2
192.168.9.17	10.10.10.3
192.168.9.16	10.10.10.4

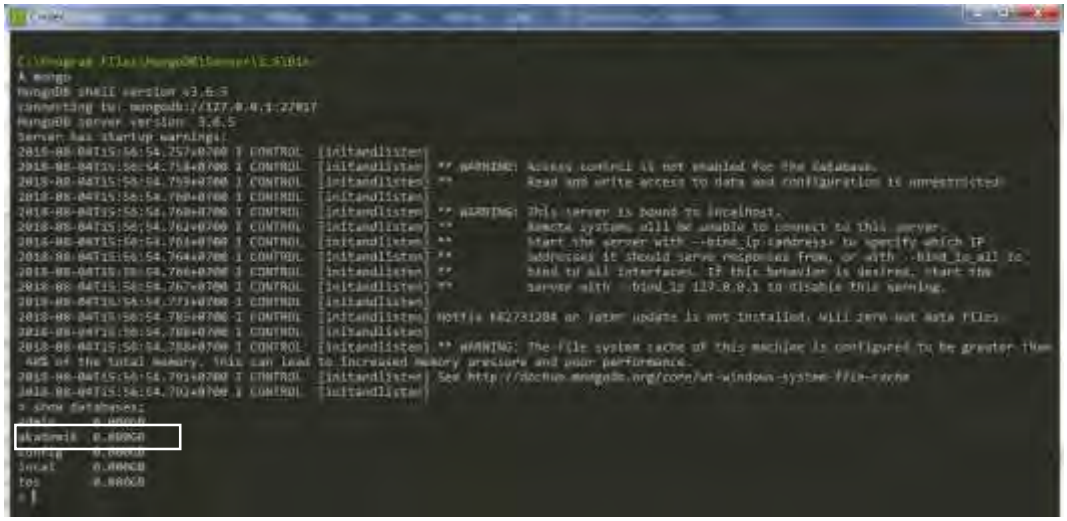
5.2 Simulasi Dan Instalasi MongoDB

Simulasi dilakukan dengan menggunakan dua komputer sebagai server NoSQL database MongoDB, setelah semua komputer terhubung ke openVPN server, langkah selanjutnya adalah memasang paket MongoDB disemua komputer. Untuk sistem operasi Linux berbasis Debian cari menginstall dengan menggunakan perintah (# sudo apt-get install mongodb).

Untuk menyalakan MongoDB digunakan perintah mongod seperti yang diperlihatkan pada Gambar 5.2

Gambar 5.2
Pemantauan MongoDB

Untuk melihat kesiapan *database* MongoDB dapat digunakan perintah (mongo), akan muncul informasi seperti gambar 5.3, terlihat server MongoDB sudah dapat diakses dan *database* akademik juga sudah tersedia.



Gambar 5.3
Server MongoDB *Running*

5.3 Evaluasi

Bab ini terdiri dari evaluasi proses migrasi pada server yang sudah terinstall sebagai simulasi. Proses evaluasi mencakup verifikasi proses migrasi data dengan perbandingan kinerja berdasarkan operasi yang sama antara MySQL dan MongoDB *database*. Berikut langkah-langkah yang digunakan sebagai tujuan evaluasi :

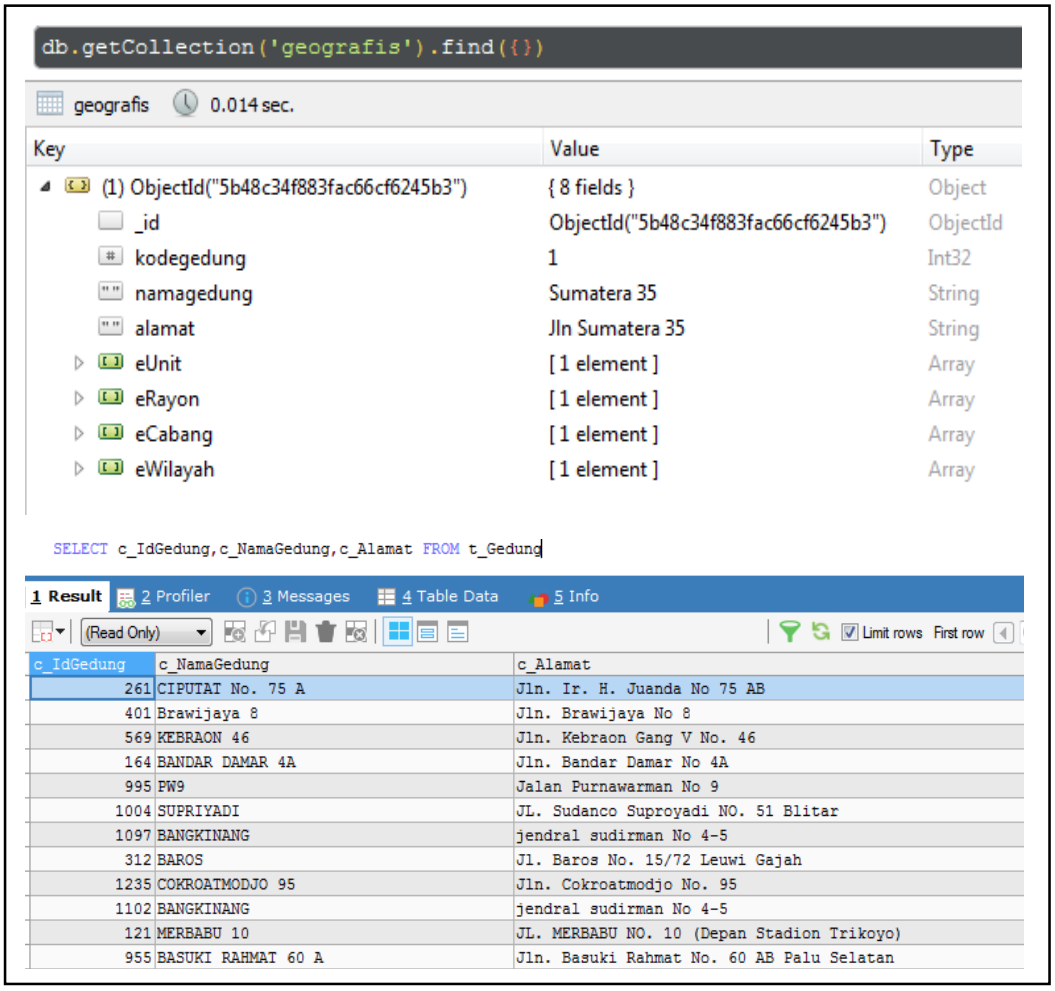
1. Verifikasi Migrasi Data
2. Penilaian Kinerja *Database*
3. Kesederhaan kueri

5.3.1 Verifikasi Migrasi Data

Untuk memverifikasi apakah proses migrasi data berhasil dilakukan atau tidak, maka beberapa bagian operasi dasar *database* MongoDB yang indentik dengan MySQL *database* digunakan seperti INSERT, UPDATE, DELETE, SELECT. Untuk data MySQL digunakan MySQL Workbench dan untuk MongoDB digunakan Robotomongo.

Gambar 5.4 memperlihatkan verifikasi data awal dengan membandingkan total jumlah data pada salah satu koleksi yang ada di dokumen MongoDB dan tabel yang ada pada database MySQL. Terlihat pada Gambar 5.4 menunjukan 10 data pada MongoDB

yang terdaftar dilakukan dengan menggunakan fungsi kueri “FIND” yang indentik dengan pernyataan “SELECT” pada MySQL.



```
db.getCollection('geografis').find({})
```

geografis 0.014 sec.

Key	Value	Type
(1) ObjectId("5b48c34f883fac66cf6245b3")	{ 8 fields }	Object
_id	ObjectId("5b48c34f883fac66cf6245b3")	ObjectId
kodegedung	1	Int32
namagedung	Sumatera 35	String
alamat	Jln Sumatera 35	String
eUnit	[1 element]	Array
eRayon	[1 element]	Array
eCabang	[1 element]	Array
eWilayah	[1 element]	Array

```
SELECT c_IdGedung,c_NamaGedung,c_Alamat FROM t_Gedung
```

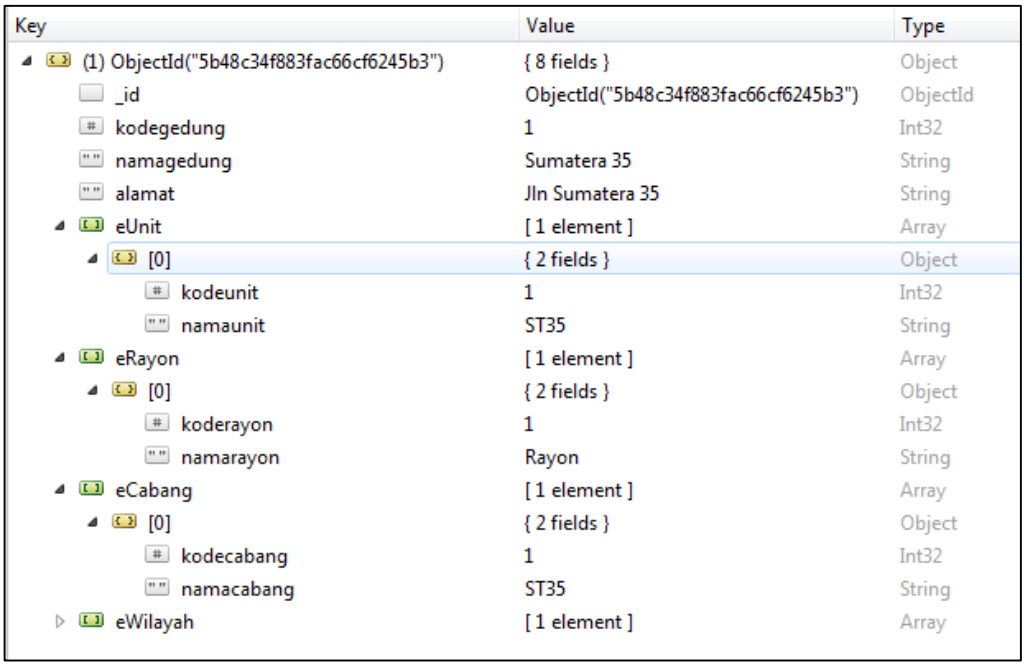
1 Result | 2 Profiler | 3 Messages | 4 Table Data | 5 Info

(Read Only) Limit rows First row

c_IdGedung	c_NamaGedung	c_Alamat
261	CIPUTAT No. 75 A	Jln. Ir. H. Juanda No 75 AB
401	Brawijaya 8	Jln. Brawijaya No 8
569	KEBRAON 46	Jln. Kebraon Gang V No. 46
164	BANDAR DAMAR 4A	Jln. Bandar Damar No 4A
995	PW9	Jalan Purnawarman No 9
1004	SUPRIYADI	JL. Sudanco Suproyadi NO. 51 Blitar
1097	BANGKINANG	jendral sudirman No 4-5
312	BAROS	Jl. Baros No. 15/72 Leuwi Gajah
1235	COKROATMODJO 95	Jln. Cokroatmodjo No. 95
1102	BANGKINANG	jendral sudirman No 4-5
121	MERBABU 10	JL. MERBABU NO. 10 (Depan Stadion Trikoyo)
955	BASUKI RAHMAT 60 A	Jln. Basuki Rahmat No. 60 AB Palu Selatan

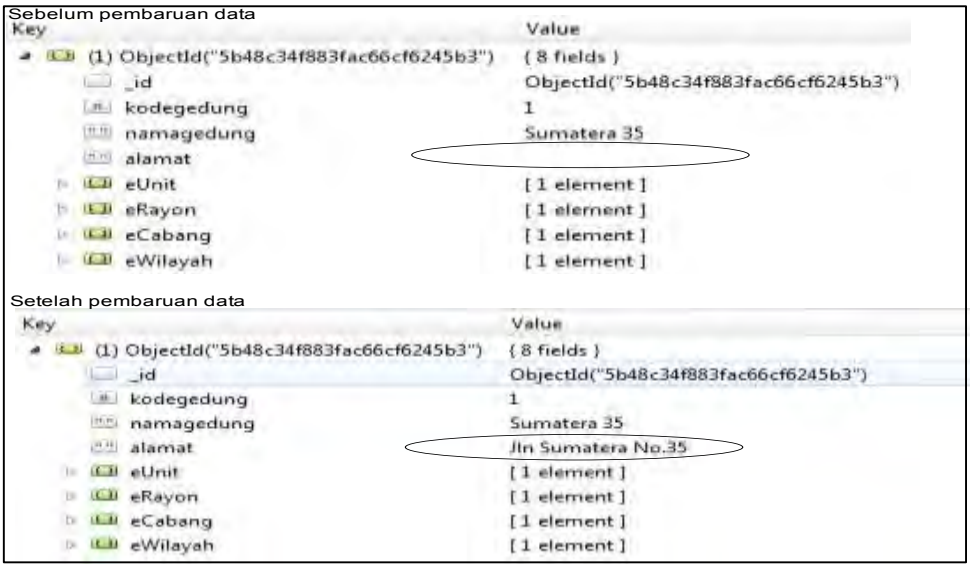
Gambar 5.4
Pengambilan Data Pada MongoDB dan MySQL

Contoh berikutnya mewakili pengambilan data pada koleksi dengan dokumen bersarang yaitu gedung yang mencakup informasi detail geografis unit, rayon, cabang, dan wilayah. Disini ditambahkan parameternya yang berfungsi sebagai “WHERE” pada pernyataan dari *database* MySQL seperti yang diperlihatkan pada Gambar 5.5.



Gambar 5.5
Pengambilan Data Dengan Parameter “WHERE”

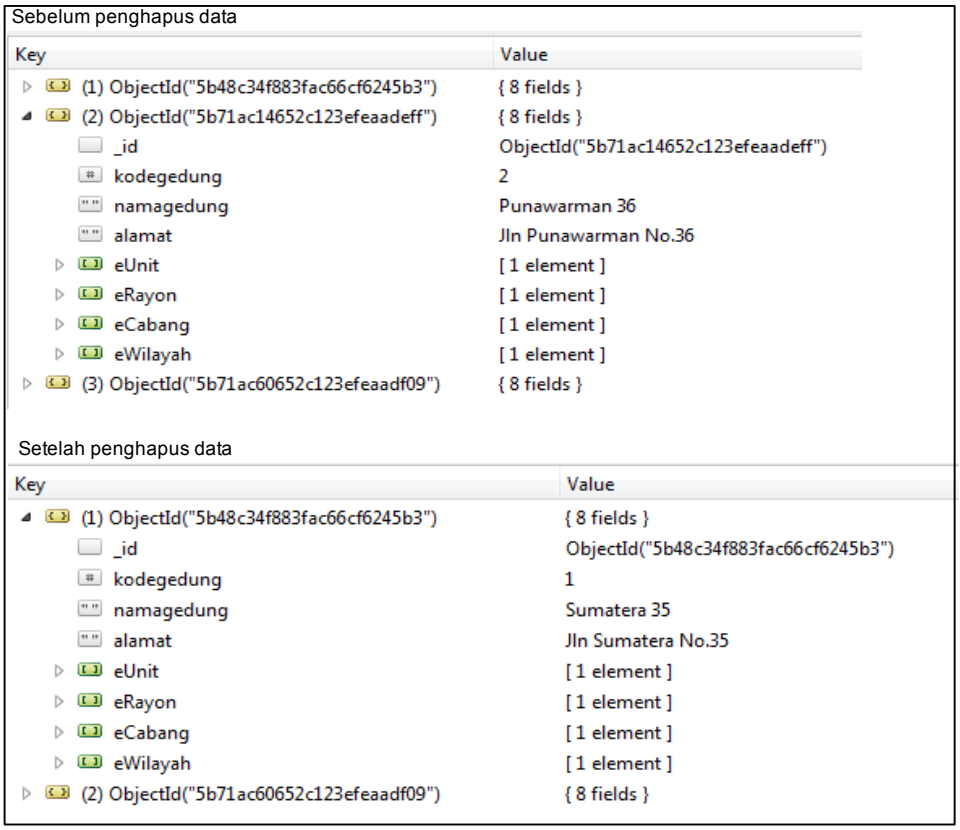
Berikut ini adalah contoh untuk mewakili operasi pembaruan dalam *database* MongoDB. Untuk melakukan pembaruan informasi apapun dalam *database* MongoDB digunakan fungsi “EDIT” yang mirip dengan pernyataan pada *database* MySQL “UPDATE”. Fungsi “EDIT” milik MongoDB juga digunakan untuk menghapus dokumen bersarang dari dokumen utama.



Gambar 5.6
Contoh Pembaruan Data Pada Operasi MongoDB

Gambar 5.6 menyajikan dua operasi pembaruan data yang dengan pembaruan data dengan merubah alamat yang sebelumnya kosong lau diberikan nilai baru.

Seperti informasi dalam model *database* relational, koleksi pun dapat dihapus dari *database* MongoDB mengunakan fungsi “DELETE”



Gambar 5.7
Contoh Penghapusan Data Pada Operasi MongoDB

5.3.2 Penilaian Kinerja *Database*

MongoDB adalah *database* open source yang tujuan utamanya berfokus pada kinerja yang tinggi. Analisis kinerja dilakukan berdasarkan perbandingan waktu antara MySQL dan MongoDB untuk operasi dasar *database*. Untuk mendapatkan hasil perbandingkan yang adil untk kedua *database* maka lingkungan kerja *hardware* dan software juga memiliki spesifikasi yang sama yaitu untuk *hardware* menggunakan processor Intel (R) Core i5-4170 CPU @3.70GHz, RAM 12GB, System Type 64-Bit. Sedangkan untuk software menggunakan sistem operasi windows 10, *database* editor

untuk MySQL menggunakan sqlYOG dan database MongoDB menggunakan Robo 3T. Berdasarkan jenis operasi, bagian ini mencakup dua bagian sebagai berikut :

5.3.2.1 Kinerja Terkait Penyimpanan Data

Tabel 5.2 Hasil Perbandingan Kinerja Terkait Penyimpan Data

Jumlah Data	INSERT (Waktu dalam milidetik)		UPDATE (Waktu dalam milidetik)		DELETE (Waktu dalam milidetik)	
	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB
100	2707	82	361	68	648	94
200	5335	121	705	105	1490	121
300	7826	147	1162	134	2135	143
400	11341	174	1532	165	2793	167
500	15510	204	1851	195	3581	196
600	19718	228	2207	227	4159	222
700	23922	250	3541	260	4768	246
800	26859	283	2882	289	5248	270
900	30086	313	3213	318	5795	294
1000	34087	343	3549	346	6134	324
5000	143000	606	5608	386	6982	367
10000	304000	995	7667	405	7206	401
50000	1519000	1401	9305	467	8233	423

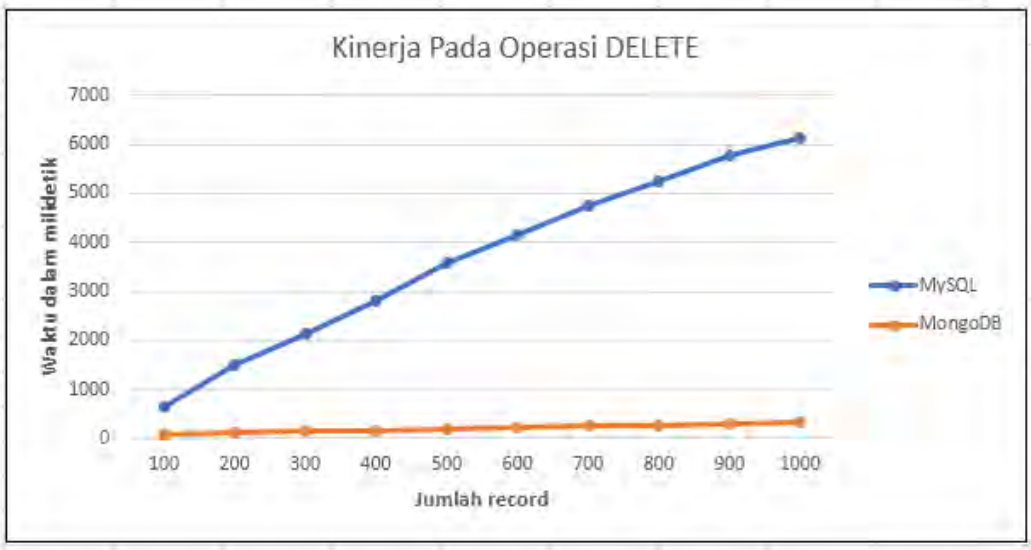
Gambar 5.8 memperlihatkan grafik dari perbandingan kinerja MySQL dan MongoDB untuk operasi INSERT

Gambar 5.8
Grafik Kinerja Perbandingan Operasi INSERT

Gambar 5.9 memperlihatkan grafik dari perbandingan kinerja MySQL dan MongoDB untuk operasi UPDATE.

Gambar 5.9
Grafik Kinerja Perbandingan Operasi UPDATE

Gambar 5.10 memperlihatkan grafik dari perbandingan kinerja MySQL dan MongoDB untuk operasi DELETE.



Gambar 5.10
Kinerja Perbandingan Operasi DELETE

5.3.2.2 Kinerja Terkait Pengambilan Data

Bagian ini mencakup analisis kinerja berdasarkan pengambilan data yang diterapkan pada MongoDB dan MySQL yang secara umum dikenal dengan “SELECT”. Untuk setiap operasi akan diamati dan dianalisis bahwa bagaimana kedua *database*

membutuhkan waktu untuk melakukan kueri. Setelah 3 kasus menunjukan hasil kinerja berasal dari kriteria pemilihan data yang berbeda sesuai dengan analisis data. Untuk semua dari 3 kasus operasi dilakukan dengan jumlah set data yang berbeda mulai dari 1.000 hingga 10.000 dengan total record yang ada pada tabel presensi 12.160.300, tabel kelas 64.265 dan tabel gedung 1154 record. Waktu yang dibutuhkan oleh setiap kriteria dicatat dalam milidetik.

1. Kasus pertama : Pengambilan data sederhana

   Dalam kasus ini kinerja diamati berdasarkan kriteria pengambilan data sederhana tanpa menerapkan kondisi atau fitur apapun. Kinerja diukur berdasarkan waktu yang diambil untuk memuat semua data dengan informasi lengkap presensi dari MySQL dan MongoDB JSON dokumen. Dengan menggunakan dua perintah untuk memuat data dari MySQL dan MongoDB seperti yang diperlihatkan pada Gambar 5.11.

MySQL Query :

```
SELECT `t_siswa`.`c_NIS`,`t_siswa`.`c_NamaLengkap`,
  `t_kelas`.`c_NamaKelas`, `t_absensi`.`c_Tgl`,
  `t_absensi`.`c_Sesi1`, `t_absensi`.`c_Sesi2`,
  `t_absensi`.`c_Sesi3`, `t_gedung`.`c_NamaGedung`
FROM
  `t_siswa`
  INNER JOIN `t_kelas` ON `t_siswa`.`c_IdKelas` = `t_kelas`.`c_IdKelas`
  INNER JOIN `t_absensi` ON `t_siswa`.`c_NIS` = `t_absensi`.`c_NIS`
  INNER JOIN `t_gedung` ON `t_kelas`.`c_IdGedung` = `t_gedung`.`c_IdGedung`;
```

MongoDB Query :

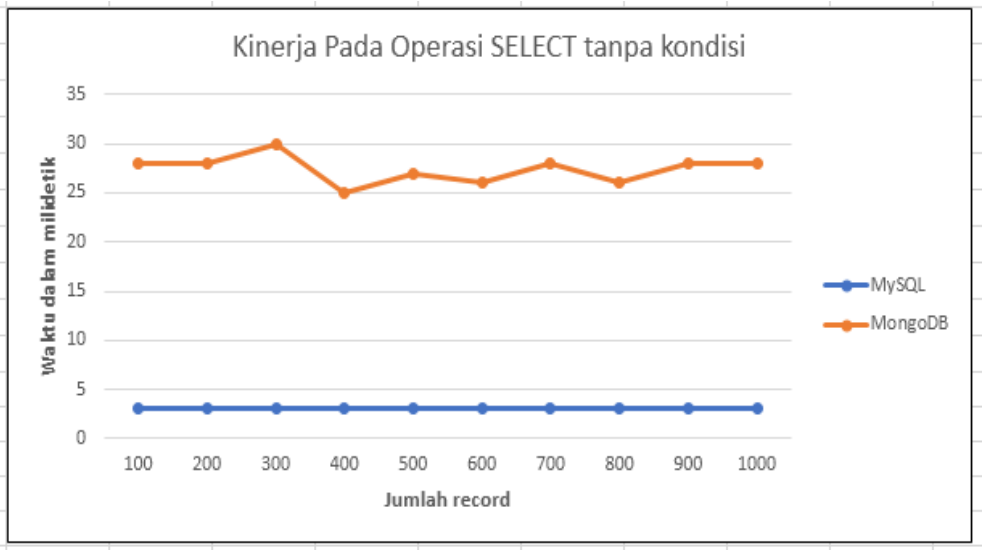
```
For MongoDB Shell
   db.presensi.Find();
```

Gambar 5.11
Perintah Sederhana Pengambilan Data Pada MySQL dan MongoDB

Tabel 5.3 Hasil Pengambilan Data Sederhana

Jumlah Record	Waktu Untuk Pengambilan Data (dalam milidetik)	
	MySQL	MongoDB
1000	3	28
2000	3	28
3000	3	30
4000	3	25
5000	3	27
6000	3	26
7000	3	28
8000	3	26
9000	3	28

Gambar 5.12 menunjukan grafik dari hasil data pengambilan data sederhana pada *database* MySQL dan MongoDB. MySQL menunjukan lebih baik dibandingkan dengan MongoDB. Untuk setiap ujicoba *database* MySQL menunjukan kinerja yang signifikan atas database MongoDB yang stabil dan tidak bervariasi dengan jumlah data yang diambil. Namun di sisi lain MongoDB membutuhkan lebih banyak waktu untuk memuat data sederhana, tetapi kinerjanya tetap stabil dengan jumlah data yang diambil.



Gambar 5.12
Grafik Sederhana Pengambilan Data Pada MySQL dan MongoDB

2. Kasus kedua : Pengambilan data dengan kondisi Order BY

Kasus ini menunjukan bagaimana memuat data dengan MySQL dengan pernyataan ORDER BY dan sintak MongoDB. Hasil diukur dengan waktu yang diambil untuk memuat atau memilih semua data dengan informasi lengkap dari MySQL yang berbeda MongoDB dengan menerapkan ORDER BY dan sintak yang identik. Data yang direkam dalam pengamatan diwakili dalam tabel 5.4 dan Gambar 5.14 memrepresentasikan grafik untuk analisis komparatif yang diperloleh dari data observasi yang tersedia pada tabel 5.4. Gambar 5.13 memperlihatkan sintak yang digunakan pada MySQL dan MongoDB dalam pengambilan data dengan kondisi tertentu.

MySQL Query :

```
SELECT `t_siswa`.`c_NIS`,`t_siswa`.`c_NamaLengkap`,
  `t_kelas`.`c_NamaKelas`, `t_absensi`.`c_Tgl`,
  `t_absensi`.`c_Sesi1`, `t_absensi`.`c_Sesi2`,
  `t_absensi`.`c_Sesi3`, `t_gedung`.`c_NamaGedung`
FROM
  `t_siswa`
  INNER JOIN `t_kelas` ON `t_siswa`.`c_IdKelas` = `t_kelas`.`c_IdKelas`
  INNER JOIN `t_absensi` ON `t_siswa`.`c_NIS` = `t_absensi`.`c_NIS`
  INNER JOIN `t_gedung` ON `t_kelas`.`c_IdGedung` = `t_gedung`.`c_IdGedung`
ORDER BY `t_siswa`.`c_NIS`;
```

MongoDB Query :

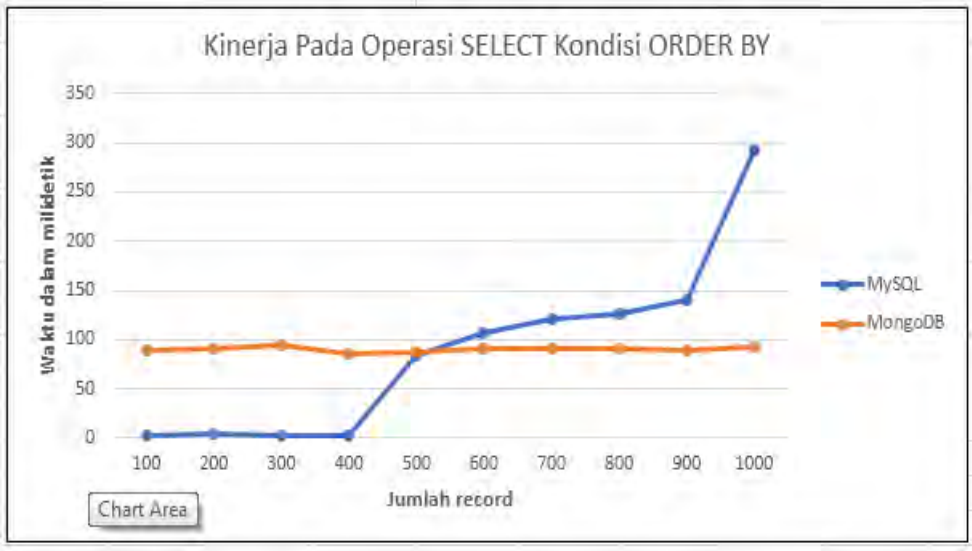
```
For MongoDB Shell
   db.presensi.Find().sort ({nis: 1});
```

Gambar 5.13
Perintah Pengambilan Data Kondisi ORDER BY
Pada MySQL dan MongoDB

Tabel 5.4 Hasil Pengambilan Data Dengan Kondisi ORDER BY

Jumlah Record	Waktu Untuk Pengambilan Data (dalam milidetik)	
	MySQL	MongoDB
100	3	89
200	4	91
300	3	94
400	3	86
500	84	87
600	107	91
700	121	91
800	127	90
900	140	89
1000	293	93

Seperti yang diperlihatkan pada Gambar 5.14, analisis komparatif berasal dari ujicoba menggunakan ORDER BY mencerminkan kinerja yang signifikan dari *database* MongoDB atas *database* MySQL. Menurut representasi grafik waktu yang diperlukan untuk melakukan ekstraksi data hampir tidak berubah dan stabil dengan peningkatan jumlah data untuk MongoDB. Disisi lain kinerja MySQL lebih baik dan stabil hingga 4000 data. Tapi setelah 4000 data MySQL tiba-tiba mulai mengambil lebih banyak waktu dan kinerjanya secara bertahap menurun dengan peningkatan jumlah data.



Gambar 5.14
Grafik Pengambilan Data Dengan Kondisi *ORDER BY*
Pada MySQL dan MongoDB

3. Kasus ketiga : Pengambilan data dengan kondisi Group BY

Pada kasus ketiga ini MySQL dan MongoDB menerapkan pernyataan *aggregate* yang relevan. Pada kasus ini akan menggunakan GROUP BY pada pengamatan data yang berasal dari sepuluh uji yang berbeda untuk masing-masing jumlah data mulai 1.000 hingga 10.000. Adapun fungsi *aggregate* yang digunakan pada MySQL dan MongoDB diperlihatkan pada Gambar 5.15 dan dicatat waktu yang dibutuhkan untuk mengambil dan menampilkan informasi dicatat pada Tabel 5.5

MySQL Query :

```
SELECT `t_siswa`.`c_NIS`,`t_siswa`.`c_NamaLengkap`,
  `t_kelas`.`c_NamaKelas`, `t_absensi`.`c_Tgl`,
  `t_absensi`.`c_Sesi1`, `t_absensi`.`c_Sesi2`,
  `t_absensi`.`c_Sesi3`, `t_gedung`.`c_NamaGedung`
FROM
  `t_siswa`
  INNER JOIN `t_kelas` ON `t_siswa`.`c_IdKelas` = `t_kelas`.`c_IdKelas`
  INNER JOIN `t_absensi` ON `t_siswa`.`c_NIS` = `t_absensi`.`c_NIS`
  INNER JOIN `t_gedung` ON `t_kelas`.`c_IdGedung` = `t_gedung`.`c_IdGedung`
GROUP BY `t_siswa`.`c_NIS`, `t_absensi`.`c_Tgl`, `t_absensi`.`c_Sesi1`, `t_absensi`.`c_Sesi2` ;
```

MongoDB Query :

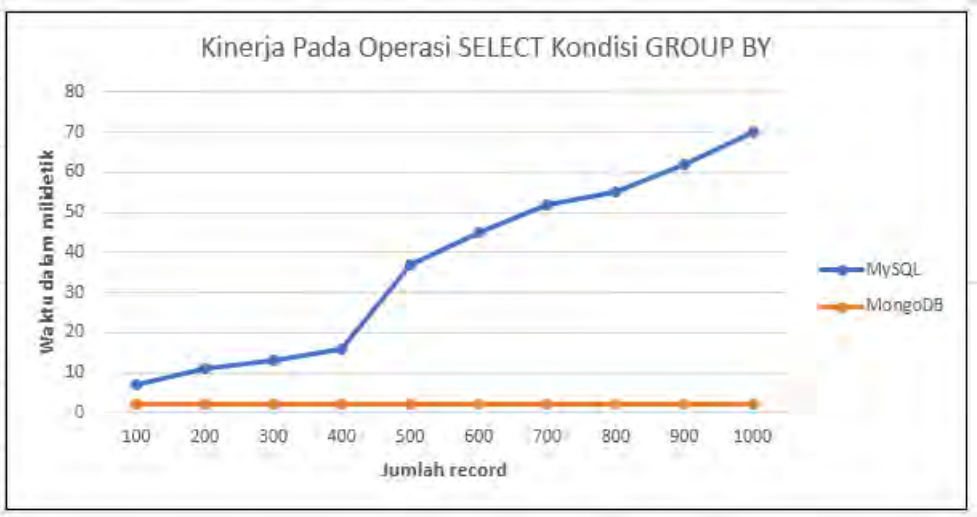
```
For MongoDB Shell
   db.presensi.aggregate (
   [
     { $unwind : "$presensi"},
        {
           $group :
           {$nis, $tglpresensi, $sesi1, $sesi2}
        }
   ]
);
```

Gambar 5.15
Perintah Pengambilan Data Kondisi GROUP BY
'Pada MySQL dan MongoDB

Tabel 5.5 Hasil Pengambilan Data Dengan Kondisi GROUP BY

Jumlah Record	Waktu Untuk Pengambilan Data (dalam milidetik)	
	MySQL	MongoDB
100	7	2
200	11	2
300	13	2
400	16	2
500	37	2
600	45	2
700	52	2
800	55	2
900	62	2
1000	70	2

Gambar 5.16 menunjukan grafik kinerja signifikan yang dilakukan oleh MongoDB melalui MySQL *database*. Menurut grafik yang berasal, waktu yang diperlukan untuk mengambil data hampir tidak berubah dan stabil dengan peningkatan volume data untuk NoSQL database MongoDB, sementara kinerja MySQL secara bertahap menurun dengan peningkatan volume data.



Gambar 5.16
Grafik Pengambilan Data Dengan Kondisi *GROUP BY*
Pada MySQL dan MongoDB

5.3.3 Kesederhanaan *Kueri*

Sebagai *database* yang fleksibel, MongoDB menawarkan fitur yang kaya untuk pemodelan data dan kueri data. Fitur kueri yang kaya membuat para pengembang mudah untuk menulis penyataan untuk mengakses *database*. Perintah MongoDB dilakukan mengunakan fungsi yang tersedia dari API JavaScript tempat kueri berada dikirim ke *database* MongoDB sebgai objek JSON. Gambar 5.17 memperlihatkan bagaimana kesederhaan kueri pada MongoDB dibandingkan dengan MySQL.

MySQL Query :

```
SELECT `t_siswa`.`c_NIS`,`t_siswa`.`c_NamaLengkap`,
  `t_kelas`.`c_NamaKelas`, `t_absensi`.`c_Tgl`,
  `t_absensi`.`c_Sesi1`, `t_absensi`.`c_Sesi2`,
  `t_absensi`.`c_Sesi3`, `t_gedung`.`c_NamaGedung`
FROM
  `t_siswa`
  INNER JOIN `t_kelas` ON `t_siswa`.`c_IdKelas` = `t_kelas`.`c_IdKelas`
  INNER JOIN `t_absensi` ON `t_siswa`.`c_NIS` = `t_absensi`.`c_NIS`
  INNER JOIN `t_gedung` ON `t_kelas`.`c_IdGedung` = `t_gedung`.`c_IdGedung`;
```

MongoDB Query :

```
For MongoDB Shell
   db.presensi.Find();
```

Gambar 5.17
Perbandingan Perintah Pada MySQL dan MongoDB

Perintah query berikut akan mengambil data dengan kondisi rentang suatu tanggal yang sering digunakan pada permintaa data seperti yang diperlihatkan pada Gambar 5.18.

MySQL Query:

```
SELECT orders.order_ID, orders.order_date, orders.order_cust_ID, customer.cust_fname,
        customer.cust_lname, customer.cust_addrs_street, customer.cust_addrs_city,
        customer.cust_addrs_postcode, customer.cust_addrs_country,
        customer.cust_addrs_phone, customer.cust_email, orders.order_chk_completed,
        orders.order_completion_date, order_details.order_prod_id, product.prod_name,
        order_details.order_prod_qty, order_details.order_prod_price,
        order_details.order_chk_shipped, order_details.order_ship_date,
        order_details.order_chk_delivered, order_details.order_delivery_date, supplier.splr_id,
        supplier.splr_fname, supplier.splr_lname, supplier.splr_addrs_street,
        supplier.splr_addrs_city, supplier.splr_addrs_postcode, supplier.splr_addrs_country,
        supplier.splr_addrs_phone, supplier.splr_email
FROM customer INNER JOIN (((order_details INNER JOIN product ON
        order_details.order_prod_ID = product.prod_id) INNER JOIN supplier ON
        product.prod_splr_id = supplier.splr_id) INNER JOIN Orders ON
        order_Details.order_ID = orders.order_ID) ON customer.cust_id = orders.order_cust_ID
        where orders.order_date between '2014-06-01' and '2014-06-30'
```

Gambar 5.18
Perbandingan Perintah Dengan Kondisi Rentang Tanggal
Pada MySQL dan MongoDB

5.4 Temuan

Berdasarkan langkah-langkah yang sudah dilakukan pada implementasi dan evaluasi dapat diringkas hasil temuan sebagi berikut :

1. Verifikasi data menunjukan bahwa proses migrasi data dari *database* MySQL untuk *database* NoSQL MongoDB dilakukan dengan sukses. MongoDB juga melakukan semua operasi perintah dasar seperti INSERT, UPDATE, DELETE dan SELECT yang identik dengan MySQL.
2. MongoDB melakukan secara signifikan lebih baik dari MySQL untuk operasi penyimpanan data terkait INSERT, UPDATE, dan DELETE.
3. MongoDB menunjukan hasil yang luar biasa untuk agregasi data dan penyortiran data yang merupakan aspek untuk manajemen data analisis.

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari analisis, implementasi dan evalusi yang sudah dilakukan dapat diperoleh beberapa kesimpulan sebagai berikut :

1. Proses bisnis yang ada di objek penelitian memerlukan perubahan *database* dan perubahan struktur data. Perubahan dilakukan karena *database* sebelumnya tidak memenuhi kebutuhan organisasi untuk penanganan jumlah data yang besar dan manajemen data analisis. Perubahan struktur *database* diperlukan agar dapat digunakan pada *database* NoSQL.
2. Data yang ada pada tabel di *database* lama akan tetap digunakan, cukup dengan diimport untuk tabel tunggal dengan skrip SQL dan tabel-tabel yang berelasi akan diimport dengan skrip yang sudah dibuatkan sesuai dengan analisis pada kelompok masing-masing.
3. Replikasi pada *database* dilakukan secara otomatis oleh *database* MongoDB dengan melakukan sedikit konfigurasi.

6.2 Saran

Ada beberapa saran agar perancangan dan implementasi *database* NoSQL bisa berjalan dengan baik yaitu :

1. Diperlukan perancangan tabel di bidang lain dan proses bisnis yang lain di bidang yang sama agar dokumen pada database NoSQL di objek penelitian semakin lengkap. Kesulitan akan didapatkan jika penambahan tabel terjadi ketika *database* sudah diimplementasikan terutama untuk tabel-tabel yang berkaitan dengan tabel-tabel yang lain.
2. Untuk mendapatkan model dokumen yang lebih sesuai dengan kebutuhan aplikasi dapat dilakukan penelitian tentang pembentukan model dokumen yang tidak hanya didasarkan pada skema *database*, tetapi harus disesuaikan pula dengan query

pengaksesan data yang dilakukan selama berjalannya aplikasi. Dengan demikian model dokumen yang dibentuk tidak hanya digunakan sebagai model penyimpanan.

DAFTAR PUSTAKA

Ahmadi. (2012). Automatic Data Migration between Two Databases with Different Structure. *International journal of Aoolied Information Systems (IJAIS)*.

Anomin. (2016). *GoalKicker.com*. Diambil kembali dari http://GoalKicker.com/MongoDBBook

Anomin. (2017). *MongoDB NoSQL Document Database.* Tutotials Point.

Anomin. (2018). *RDBMS to MongoDB Migration Guide.* mongoDB.

Arganata, G., Pramukantoro, E. S., & Yahya, W. (2017, Juli 7). Pengembangan Sistem Penyimpanan Data Berbasis MongoDB dan GridFS. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2*, 9.

Bhaswara, F. A., Sarno, R., & Sunaryono, D. (2017). Perbandingan Kemampuan Database NoSQL dan SOL dalam Kasus ERP Retail. *Jurnal Teknik ITS, 6*, 5.

Boyce, R. F. (1974). SEQUEL: A Structured English Query. *ACM SIGFIDET (now SIGMOD).*

Brewer, E. (2010, 05). Diambil kembali dari Better explaining the CAP Theorem: http://architects.dzone.com/articles/better-explaining-cap-theorem

Carlo, S. (2017). *http://www.strozzi.it*. Dipetik Maret 05, 2018, dari NoSQL – A relational database management system: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page

Chaitanya, P. (2016). Implementation of an Efficient MongoDB NoSQL. *International Journal of Advanced Networking & Applications (IJANA), 2*, 4.

Chistof, S. (2014). *NoSQL Database.* Stuttgart: Stuttgart Media University.

Codd, E., Codd, S., & Salley, C. (2015, Mei). *Providing OLAP to User-Analysts: An IT Mandate.* Diambil kembali dari Providing OLAP to User-Analysts: An IT Mandate: http://www.minet.uni-jena.de/dbis/lehre/ss2005/sem_dwh/lit/Cod93.pdf

Coronel, C., & Morris, S. (2015). *Design Implementation and Management 10th Edition Course Technology.* Cengage Learning.

Couchbase. (2015). *Making the Shift from Relational to NoSQL.* Whitepapers.

Date, C. (2013). *An Introduction to Database Systems (8th Edition).* United State: Pearson Education.

Davenport, R. (2007). Design or Centralised Data Base. *The Computer Journal Vol 24 No1, 2*.

Dean, J., & Ghemawat, S. (2009). MapReduce: Simplified Data Processing on Large Clusters. *OSDI.*

Elmasri, R. (2011). *Fundamental Research of Distributed 6th Edition.* Addison-Wesley.

Francia, S. (2012). *MongoDB and PHP.* USA: O'Reilly Media.

Ghotiya, S., Mandal, J., & Kandasamy, S. (2017). Migration from relational to NoSQL database. *Materials Science and Engineering, 4*, 8.

Gupta, S., Kuntal, S., & Bhawna. (2011). Fundamental Reseacrh of Distributed Database. *International Journal of Computer Science and Management Studies, 11*(02).

Hartati, S., & Nugroho, A. (2017). MongoDB: Implementasi VLDB (Very Large Database). *UGM, 7*.

Haryanto. (2007). *Kumpulan Resep Query Menggunakan MYSQL.* Dian Rakyat.

Hevner, A. (2010). *Design Research in Information Systems: Theory and Practice (Integrated Series in Information Systems).* Springer.

Kang, W. (2010). *An Investigation of No-SQL Data Stores*. Diambil kembali dari http://www.comp.nus.edu.sg/~ozsu/cs5225/Projects/CS5225Final_v15.pdf

Khan, S. (2013). SQL Support over MongoDB using Metadata. *IJSRP, 3*, 10.

Kumar, L., Rajawat, S., & Joshi, K. (2015). Comparative analysis of NoSQL (MongoDB) with MySQL Database. *International Journal of Modern Trends in Engineering and Research (IJMTER)*.

Kung, L., & Adrian, G. (2012). Comparing Top-Down with Bottom-Up Approaches Teaching Data Modeling. *Proceeding of the Information Educators Conference.*

Leavitt, N. (2010). Will NO-SQL Basis datas Live Up to Their Promise. *IEEE Computer Society, 9*, 7.

Mannino, M. (2014). *Database Design, Application Development, and Administration, Sixth Edition by Michael Mannino.* Mass Market Paperback.

Mazilu, M. (2010). Database Replication. *dbJournal, 4*.

MongoDB nosql document database. (2010). tutotialspoint.

MongoDB, T. (t.thn.). *mongoDB*. Diambil kembali dari https://docs.mongodb.com/manual/introduction/

Moniruzzaman, A., & Hossain, S. (2013). NoSQL Database New Era of Databases. *Internasional Journal of Database Theory and Application, 6*.

Orina, & Fitri, M. (2013). TREND PENGGUNAAN NOSQL UNTUK BASIS DATA. *Teknosains, 7*, 8.

Oszum, T., & Patrik, V. (2011). *Principles of Distributed Database System.* Springer Science Business Media.

Powel, G. (2009). *Beginning Database Design.* Wiley Publishing.

Putra, E. K., & Rahmayeni, F. (2016, April 1). IMPLEMENTASI DATABASE MONGODB UNTUK SISTEM INFORMASI. *Jurnal TEKNIOF, Vol 4*, 7.

Ramakrishnan. (2005). *Database Management System.* Mc Graw Hill 2nd Edition.

Rohman, M. A., Beta, N., & Djalal, E. R. (2014). Pembangunan Prototype Informasi Administrasi Kependudukan Berbasis Data Terdistribusi. *Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Diponegoro*.

Rummier, B. (2010). *Improving Performance How to Manage to White Space On The Organizational Chart.* Jossey-Bass.

Seguin, K. (2016). *The Little MongoDB Book*. Diambil kembali dari http://github.com/karlseguin/the-little-mongodb-book

Simanjuntak, H., Simanjuntak, L., & Situmorang, G. (2015, Januari 1). QUERY RESPONSE TIME COMPARISON NOSQLDB MONGODB. *Jurnal Ilmiah Teknologi Informasi*, 12.

Siregar, N. P., Kemas, R., & Alfian, A. (2015). Analisis dan Implementasi Basis Data Terdistribusi Horizontal pada MongoDB untuk ClikKB BKKBN. *e-Proceeding of Engineering. 2*, hal. 11. ISSN.

Steven, M. (2007). *Database Design, Application development, And Administration.* Graw Hill.

Talab. (2013). *Organizational Behavior (17th Edition).* Pearson.

Vaisg, G. (2013). *Getting Started with NoSQL Retrieved through Ryerson Library*. Diambil kembali dari http://proquest.safaribooksonline.com/book/databases/9781849694988

Valentine, D. (2013, Februari). Diambil kembali dari IngeniousSQL: http://www.ingenioussql.com/2013/02/

Wahyudi, D. (2013). Rancang bangun Heterogenous Distributed Database System Untuk Meningkatkan Kapasitas Oracle XE 10g Pada Studi Kasus Sistem Informasi Akademik. *Integration and Interconnection Islam and Science, IX*.

Wei ping, Z. (2011). Using MongoDB to Implement Textbook Management System instead. *IEEE,978-1-61284-486*.

White, T. (2010). *Hadoop The Definitive Guide.* O'Reily.

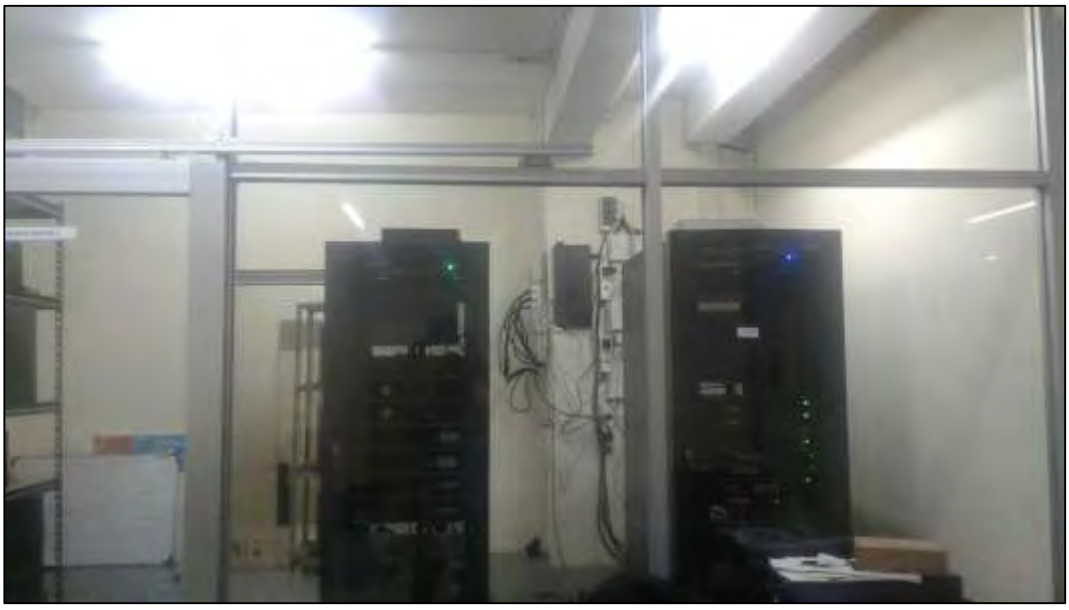
Winaya, I., & Ashari, A. (2016, Januari 1). Transformasi Skema Basis Data Relasional Menjadi Model Data Berorientasi Dokumen pada MongoDB. *IJCCS, Val 10*, 12.

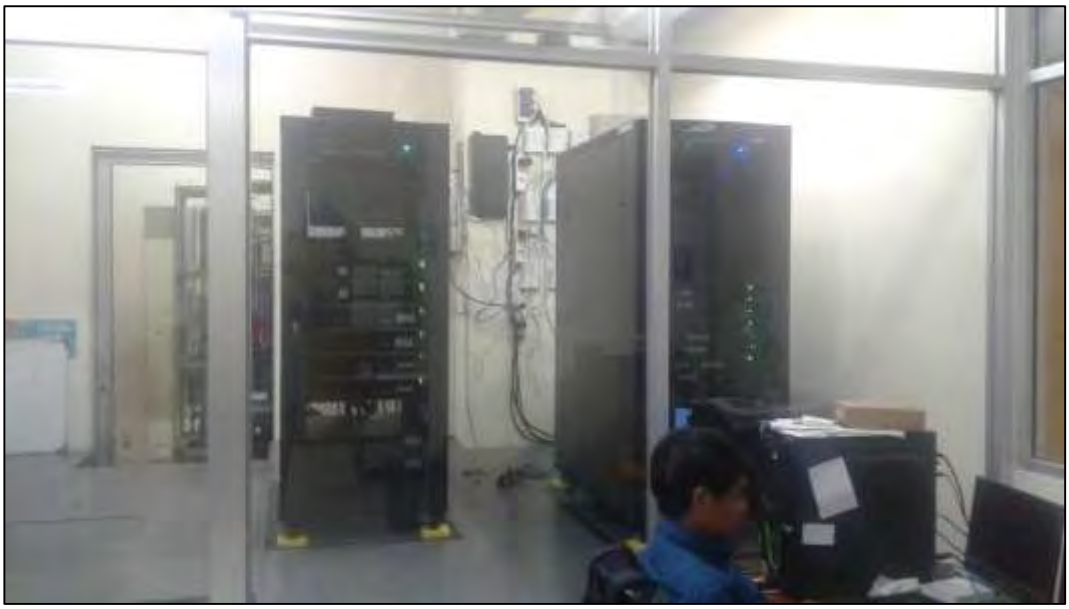
Zins, C. (2010). *Conseptual Approaches for defining data. information dan knowlegde.*

LAMPIRAN

Lampiran 1.

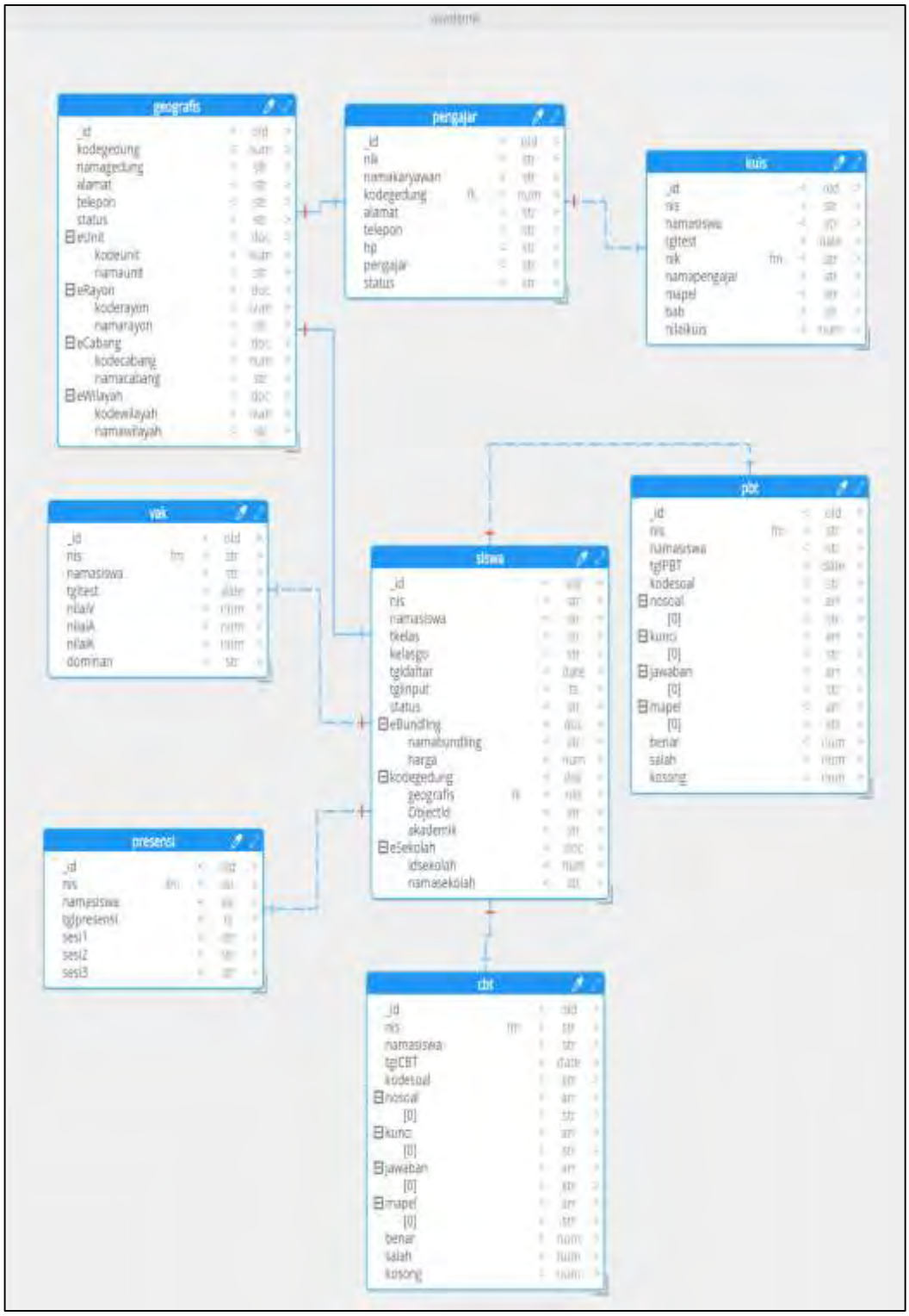
Foto ruangan dan server





Lampiran 2.

Model Data Geografis, Siswa, Pengajar, Akademik



Lampiran 3.

MongoDBScript Geografis, Siswa, Pengajar, Akademik

Geografis

```
db.createCollection( "geografis",{
   "storageEngine": {
      "wiredTiger": {}
   },
   "capped": false,
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "additionalProperties": false,
         "properties": {
            "_id": {
               "bsonType": "objectId"
            },
            "kodegedung": {
               "bsonType": "number"
            },
            "namagedung": {
               "bsonType": "string"
            },
            "alamat": {
               "bsonType": "string"
            },
            "telepon": {
               "bsonType": "string"
            },
            "status": {
               "bsonType": "string"
            },
            "eUnit": {
               "bsonType": "object",
               "properties": {
                  "kodeunit": {
                     "bsonType": "number"
                  },
                  "namaunit": {
                     "bsonType": "string"
                  }
               },
               "additionalProperties": false
            },
            "eRayon": {
               "bsonType": "object",
               "properties": {
                  },
                  "namarayon": {
                     "koderayon": {
                        "bsonType": "number"
                        "bsonType": "string"
                     }
                  },
                  "additionalProperties": false
               },
               "eCabang": {
                  "bsonType": "object",
                  "properties": {
                     "kodecabang": {
                        "bsonType": "number"
                     },
                     "namacabang": {
                        "bsonType": "string"
                     }
                  },
                  "additionalProperties": false
               },
               "eWilayah": {
                  "bsonType": "object",
                  "properties": {
                     "kodewilayah": {
                        "bsonType": "number"
                     },
                     "namawilayah": {
                        "bsonType": "string"
                     }
                  },
                  "additionalProperties": false
               }
            }
         }
      },
      "validationLevel": "off",
      "validationAction": "warn"
});
```

Pengajar

```
db.createCollection( "pengajar",{
   "storageEngine": {
      "wiredTiger": {}
   },
   "capped": false,
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "additionalProperties": false,
         "properties": {
            "_id": {
               "bsonType": "objectId"
            },
            "nik": {
               "bsonType": "string"
            },
            "namakaryawan": {
               "bsonType": "string"
            },
            "kodegedung": {
               "bsonType": "number"
            },
            "alamat": {
               "bsonType": "string"
            },
            "telepon": {
               "bsonType": "string"
            },
            "hp": {
               "bsonType": "string"
            },
            "pengajar": {
               "bsonType": "string"
            },
            "status": {
               "bsonType": "string"
            }
         }
      }
   },
   "validationLevel": "off",
   "validationAction": "warn"
});
```

Kuis

```
db.createCollection( "kuis",{
   "storageEngine": {
      "wiredTiger": {}
   },
   "capped": false,
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "additionalProperties": false,
         "properties": {
            "_id": {
               "bsonType": "objectId"
            },
            "nis": {
               "bsonType": "string"
            },
            "namasiswa": {
               "bsonType": "string"
            },
            "tgltest": {
               "bsonType": "date"
            },
            "nik": {
               "bsonType": "string"
            },
            "namapengajar": {
               "bsonType": "string"
            },
            "mapel": {
               "bsonType": "string"
            },
            "bab": {
               "bsonType": "string"
            },
            "nilaikuis": {
               "bsonType": "number"
            }
         }
      }
   },
   "validationLevel": "off",
   "validationAction": "warn"
});
```

VAK

```
db.createCollection( "vak",{
   "storageEngine": {
      "wiredTiger": {}
   },
   "capped": false,
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "additionalProperties": false,
         "properties": {
            "_id": {
               "bsonType": "objectId"
            },
            "nis": {
               "bsonType": "string"
            },
            "namasiswa": {
               "bsonType": "string"
            },
            "tgltest": {
               "bsonType": "date"
            },
            "nilaiV": {
               "bsonType": "number"
            },
            "nilaiA": {
               "bsonType": "number"
            },
            "nilaiK": {
               "bsonType": "number"
            },
            "dominan": {
               "bsonType": "string"
            }
         }
      }
   },
   "validationLevel": "off",
   "validationAction": "warn"
});
```

Siswa

```
db.createCollection( "siswa",{
   "storageEngine": {
      "wiredTiger": {}
   },
   "capped": false,
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "additionalProperties": false,
         "properties": {
            "_id": {
               "bsonType": "objectId"
            },
            "nis": {
               "bsonType": "string"
            },
            "namasiswa": {
               "bsonType": "string"
            },
            "tkelas": {
               "bsonType": "string"
            },
            "kelasgo": {
               "bsonType": "string"
            },
            "tgldaftar": {
               "bsonType": "date"
            },
            "tglinput": {
               "bsonType": "timestamp"
            },
            "status": {
               "bsonType": "string"
            },
            "eBundling": {
               "bsonType": "object",
               "properties": {
                  "namabundling": {
                     "bsonType": "string"
                  },
                  "harga": {
                     "bsonType": "number"
                  }
               },
               "additionalProperties": false
            },
            "kodegedung": {
               "bsonType": "object",
               "properties": {
                  "geografis": {
                     "bsonType": "objectId"
                  },
                  "ObjectId": {
                     "bsonType": "string"
                  },
                  "akademik": {
                     "bsonType": "string"
                  }
               },
               "additionalProperties": false,
               "required": [
                  "geografis",
                  "ObjectId",
                  "akademik"
               ]
            },
            "eSekolah": {
               "bsonType": "object",
               "properties": {
                  "idsekolah": {
                     "bsonType": "number"
                  },
                  "namasekolah": {
                     "bsonType": "string"
                  }
               },
               "additionalProperties": false
            }
         }
      }
   },
   "validationLevel": "off",
   "validationAction": "warn"
});
```

PBT

```
db.createCollection( "pbt",{
   "storageEngine": {
      "wiredTiger": {}
   },
   "capped": false,
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "additionalProperties": false,
         "properties": {
            "_id": {
               "bsonType": "objectId"
            },
            "nis": {
               "bsonType": "string"
            },
            "namasiswa": {
               "bsonType": "string"
            },
            "tglPBT": {
               "bsonType": "date"
            },
            "kodesoal": {
               "bsonType": "string"
            },
            "nosoal": {
               "bsonType": "array",
               "additionalItems": true,
               "uniqueItems": false,
               "items": {
                  "bsonType": "string"
               }
            },
            "kunci": {
               "bsonType": "array",
               "additionalItems": true,
               "uniqueItems": false,
               "items": {
                  "bsonType": "string"
               }
            },
            "jawaban": {
               "bsonType": "array",
               "additionalItems": true,
               "uniqueItems": false,
               "items": {
                  "bsonType": "string"
               }
            },
            "mapel": {
               "bsonType": "array",
               "additionalItems": true,
               "uniqueItems": false,
               "items": {
                  "bsonType": "string"
               }
            },
            "benar": {
               "bsonType": "number"
            },
            "salah": {
               "bsonType": "number"
            },
            "kosong": {
               "bsonType": "number"
            }
         }
      }
   },
   "validationLevel": "off",
   "validationAction": "warn"
});
```

CBT

```
db.createCollection( "cbt",{
   "storageEngine": {
      "wiredTiger": {}
   },
   "capped": false,
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "additionalProperties": false,
         "properties": {
            "_id": {
               "bsonType": "objectId"
            },
            "nis": {
               "bsonType": "string"
            },
            "namasiswa": {
               "bsonType": "string"
            },
            "tglCBT": {
               "bsonType": "date"
            },
            "kodesoal": {
               "bsonType": "string"
            },
            "nosoal": {
               "bsonType": "array",
               "additionalItems": true,
               "uniqueItems": false,
               "items": {
                  "bsonType": "string"
               }
            },
            "kunci": {
               "bsonType": "array",
               "additionalItems": true,
               "uniqueItems": false,
               "items": {
                  "bsonType": "string"
               }
            },
            "jawaban": {
               "bsonType": "array",
               "additionalItems": true,
               "uniqueItems": false,
               "items": {
                  "bsonType": "string"
               }
            },
            "mapel": {
               "bsonType": "array",
               "additionalItems": true,
               "uniqueItems": false,
               "items": {
                  "bsonType": "string"
               }
            },
            "benar": {
               "bsonType": "number"
            },
            "salah": {
               "bsonType": "number"
            },
            "kosong": {
               "bsonType": "number"
            }
         }
      }
   },
   "validationLevel": "off",
   "validationAction": "warn"
});
```