TUGAS AKHIR - KI141502

RANCANG BANGUN *NON RELATIONAL DATABASE* PADA *ENTERPISE RESOURCE PLANING* RETAIL YANG BERORIENTASIKAN *MULTITENANCY* DENGAN *SHARED DISTRIBUTED DATABASE*

FAIZAL ANUGRAH BHASWARA
NRP 5113100084

Dosen Pembimbing I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Dosen Pembimbing II
Dwi Sunaryono S.Kom, M.Kom

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

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

RANCANG BANGUN *NON RELATIONAL DATABASE* PADA *ENTERPISE RESOURCE PLANING* RETAIL YANG BERORIENTASIKAN *MULTITENANCY* DENGAN *SHARED DISTRIBUTED DATABASE*

FAIZAL ANUGRAH BHASWARA
NRP 5113100084

Dosen Pembimbing I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Dosen Pembimbing II
Dwi Sunaryono S.Kom, M.Kom

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

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

DESIGN AND IMPLEMENTATION OF NON RELATIONAL DATABASE ON ENTERPISE RESOURCE PLANING RETAIL WITH MULTITENANCY ORIENTED USING SHARED DISTRIBUTED DATABASE

FAIZAL ANUGRAH BHASWARA
NRP 5113100084

Supervisor I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Supervisor II
Dwi Sunaryono S.Kom, M.Kom

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN *NON RELATIONAL DATABASE* PADA *ENTERPISE RESOURCE PLANING* RETAIL YANG BERORIENTASIKAN *MULTITENANCY* DENGAN *SHARED DISTRIBUTED DATABASE*

TUGAS AKHIR

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

Oleh :
FAIZAL ANUGRAH BHASWARA
NRP : 5113 100 084

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D
NIP: 19590803 198601 1001 ..... (pembimbing 1)

Dwi Sunaryono, S.Kom, M.Kom
NIP: 19720528 199702 1001 ..... (pembimbing 2)

SURABAYA
JUNI, 2017

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN *NON RELATIONAL DATABASE* PADA *ENTERPISE RESOURCE PLANING* RETAIL YANG BERORIENTASIKAN *MULTITENANCY* DENGAN *SHARED DISTRIBUTED DATABASE*

Nama Mahasiswa	:	Faizal Anugrah Bhaswara
NRP	:	5113 100 084
Jurusan	:	Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	:	Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.
Dosen Pembimbing 2	:	Dwi Sunaryono, S.Kom, M.Kom.

ABSTRAK

Tugas akhir ini merancang dan membangun sebuah aplikasi Enterprise Resource Planning (ERP) pada studi kasus Retail dengan menggunakan Non Relational Database (NoSQL). Aplikasi ini berorientasikan multitenancy dengan shared distributed database. Dalam aplikasi ini setiap tenant memiliki skema yang berbeda-beda sesuai kebutuhannya.

Tantangan utama tugas akhir ini adalah membandingkan basis data NoSQL dengan basis data SQL dalam hal performa, fleksibilitas, dan skalabilitas. Setelah terbukti mana basis data yang lebih unggul, maka akan diterapkan pada aplikasi ERP Retail dengan tujuan aplikasi memiliki performa yang baik, fleksibel dalam penyimpanan data, juga mendukung dalam penyimpanan data yang terus berkembang seiring berjalannya waktu.

Dalam uji coba yang telah dilakukan, basis data NoSQL terbukti memiliki kecepatan lebih unggul untuk penyimpanan data dalam bentuk CRUD daripada SQL. Juga mempunyai struktur penyimpanan data yang fleksibel dikarenakan bentuk datanya berupa BSON (Binary JSON). Dan memiliki kemampuan untuk

skalabel dengan cara sharding. Maka dalam kasus ini basis data NoSQL akan lebih baik untuk diterapkan pada ERP Retail.

Kata kunci: *ERP (Enterprise Resource Planning) Retail, Multitenancy, NoSQL*

DESIGN AND IMPLEMENTATION OF NON RELATIONAL DATABASE ON ENTERPISE RESOURCE PLANING RETAIL WITH MULTITENANCY ORIENTED USING SHARED DISTRIBUTED DATABASE

Student Name : Faizal Anugrah Bhaswara
Student ID : 5113 100 084
Major : Informatics Department FTIf-ITS
Supervisor I : Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.
Supervisor II : Dwi Sunaryono, S.Kom, M.Kom.

ABSTRACT

This final project is to design and implementation an application of Enterprise Resource Planning (ERP in Retail study case using Non Relational Database (NoSQL). This application is multitenancy oriented with shared distributed database. In this application every tenat has different scheme as per its requirement.

The main challenge of this final task is to compare the NoSQL database with SQL in terms of performance, flexibility, and scalability. After being proven which the right database, it will be applied to Retail ERP applications with the aim of the application has a good performance, flexible in data storage, also supports in data storage that continues to grow over time.

In the trials that has been done, NoSQL database proved to have superior speed for data storage in the case of CRUD rather than SQL. Also has a flexible structure data storage because the model of data in the form of BSON (Binary JSON). And has the ability to be scalable by sharding. So in this case NoSQL database will be better to apply to ERP Retail.

Keywords: *ERP (Enterprise Resource Planning) Retail, Multitenancy, NoSQL*

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Puji syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

RANCANG BANGUN *NON RELATIONAL DATABASE* PADA *ENTERPISE RESOURCE PLANING* RETAIL YANG BERORIENTASIKAN *MULTITENANCY* DENGAN *SHARED DISTRIBUTED DATABASE*

Melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Bapak, Ibu, kakak dan keluarga besar yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Bapak Riyanarto Sarno beserta keluarga dan Bapak Dwi Sunaryono beserta keluarga selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan tugas akhir ini.
3. Danang Adi Nugroho dan Achmad Faisal Yanuar yang selalu ada untuk menemani dan memberi support penulis di saat susah, sedih, dan bahagia.
4. Seluruh keluarga besar PPM Khoirul Huda II yang selalu menemani dan membantu penulis selama menimba ilmu di pondok.
5. Andi, Renanda, Zaenal dan Ichal selaku rekan kelompok Tugas Akhir yang telah menemani dan membantu pengerjaan Tugas Akhir ini.
6. Teman-teman Keputih Galaxy ’13 yang sebagai teman seperjuangan kuliah di Surabaya.

7. Reffany, Fahmi, Puguh, Tasya, Sabrina, Danar, Adit, dan teman-teman penulis lainnya yang tidak tersebut diatas.
8. Teman-teman seperjuangan anak didik Tugas Akhir Prof. Riyanarto yaitu Eko, Gery, Arga, Donny, Hari, Dhiaz, dan Fidi.
9. Teman-teman angkatan 2013 jurusan Teknik Informatika ITS yang telah menemani dan mendukung penulis selama kuliah.
10. Google.com yang membantu penulis mengerjakan Tugas Akhir.
11. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu per satu.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun Tugas Akhir ini, namun penulis mohon maaf apabila terdapat kekurangan yang penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2017

Faizal Anugrah Bhaswara

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Database atau basis data adalah kumpulan data yang disimpan secara sistematis di dalam komputer yang dapat diolah maupun direkayasa menggunakan perangkat lunak untuk menghasilkan informasi. Basis data merupakan aspek yang sangat penting dalam sistem informasi dimana basis data merupakan gudang penyimpanan data yang akan diolah lebih lanjut. Basis data menjadi penting karena dapat mengorganisasi data, menghidari duplikasi data, hubungan antar data yang tidak jelas dan juga update yang rumit. Dan beberapa sistem yang telah ada masih mengadopsi cara kerja *single-tenant database.* Cara kerja seperti ini cenderung akan membatasi proses bisnis yang akan berkembang kedepannya.

Perencanaan sumber daya perusahaan, atau sering disingkat ERP dari istilah bahasa Inggris-nya *Enterprise Resource Planning*, adalah sistem informasi yang diperuntukkan bagi perusahan manufaktur maupun jasa yang berperan mengintegrasikan dan mengotomasikan proses bisnis yang berhubungan dengan aspek operasi, produksi maupun distribusi di perusahaan bersangkutan.

Ritel/eceran atau disebut pula *retail* dalam bahasa inggris adalah salah satu cara pemasaran produk meliputi semua aktivitas yang melibatkan penjualan barang secara langsung ke konsumen akhir untuk penggunaan pribadi dan bukan bisnis. Jadi ERP Retail merupakan sebuah sistem

informasi yang menangani kebuhan perusahaan dalam hal pemasaran produk termasuk penjualan barang hingga sampai ke konsumen.

Tantangan dalam Tugas Akhir ini adalah membandingkan antara *database relational* (SQL) dengan *database non-relational* (NoSQL) dalam hal *performance*, *flexibility*, dan *scalability*. Setelah terbukti *database* mana yang lebih unggul dari keduanya maka akan diterapkan untuk aplikasi ERP Retail yang berorientasikan *multitenancy*. Dengan begitu diharapkan ERP Retail yang dibangun akan memiliki performa yang optimal dalam hal *store* data. Juga menjadikan aplikasi yang bersifat fleksibel agar mampu mensuport penyimpanan data yang dinamis berdasarkan skema setiap *tenant*, dan perkembangan proses bisnis yang akan selalu berkembang kedepannya. Dan menjadi aplikasi yang *scalable* dalam artian mampu mengatasi jumlah data yang selalu berkembang dengan *distributed database*.

Arsitektur *database* pada aplikasi ini menerapkan *shared database* dengan *shared schema*. Dengan tujuan menghemat jumlah database untuk menangani banyak *tenant* dengan optimal. Juga database mampu menerima skema dan struktur setiap *tenant* yang berbeda-beda.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini antara lain:

1. Diantara database jenis SQL dan NoSQL manakah yang lebih cepat performanya ?
2. Bagaimana mengimplementasikan *multitenancy* dengan memanfaatkan sifat *flexible* dari database NoSQL?
3. Bagaimana mengimplementasikan *scalability* dalam aplikasi ERP Retail dengan database NoSQL?

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, antara lain:

1. Hasil dari tugas akhir ini adalah sebuah aplikasi ERP Retail dengan query yang telah di optimasi dengan NoSQL.
2. NoSQL tidak mendukung *query* yang kompleks.
3. *Flexibility* tidak berjalan pada *level interface* aplikasi.
4. Ukuran maksimal dokumen BSON untuk MongoDB adalah 16MB
5. Fungsional Modul:
    a. Fungsi Create
    b. Fungsi Read
    c. Fungsi Update
    d. Fungsi Delete
6. Sistem ERP yang dibangun dalam aspek *Multitenancy* dengan *shared database.*
7. *Platform* menggunakan framework Yii2 dengan teknologi:
    a. Eloquent
    b. Modularity

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

1. Membandingkan database SQL dan NoSQL dalam hal performa, fleksibilitas, dan skalabilitas.
2. Membuat aplikasi ERP Retail yang dinamis dengan database NoSQL.
3. Mengimplementasikan *multitenancy* pada aplikasi ERP Retail dengan *shared database*.

1.5 Manfaat

Manfaat dari hasil tugas akhir ini adalah memberikan kontribusi untuk membangun aplikasi ERP (Enterprise Resource Planning) yang berguna untuk kalangan UKM di Indonesia demi meningkatkan pendapatan mereka. Dengan adanya

aplikasi ini diharapkan aplikasi ERP lebih berkembang untuk penggunaan skala UKM.

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

a. Penyusunan proposal tugas akhir

Proposal tugas akhir ini berisi latar belakang pembuatan tugas akhir, rumusan masalah, batasan masalah, tujuan pembuatan, manfaat, metodologi hingga jadwal kegiatan pembuatan tugas akhir. Selain itu proposal tugas akhir ini memberikan ringkasan dari tugas akhir. Proposal tugas akhir juga berisi tinjauan pustaka yang digunakan sebagai referensi pembuatan tugas akhir ini.

b. Studi literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan program yaitu mengenai *NoSQL database*, *MongoDB, Yii2 Framework*, dan *ERP System.*

c. Analisis dan desain perangkat lunak

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

d. Implementasi perangkat lunak

Implementasi perangkat lunak ini dibangun dengan bahasa pemrograman PHP dan *database NoSQL*. Selain itu untuk

memudahkan monitoring data maka digunakan Robomongo sebagai GUI Mongodb.

e. Pengujian dan evaluasi

Pada tahapan ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat.Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya system, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

f. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir
    d. Tujuan
    e. Metodologi
    f. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7 Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka aplikasi.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan dan implementasi fitur-fitur penunjang aplikasi.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode kotak hitam (*black box testing*) untuk mengetahui aspek nilai fungsionalitas dari perangkat lunak dan nilai kegunaan yang dibuat dengan juga memperhatikan ketertarikan pada calon partisipan untuk menggunakan aplikasi ini.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir.

2.1 Penelitian Terkait

Tugas Akhir ini merupakan pengembangan riset berkelanjutan dari riset atau penelitian tentang ERP. Pada Tugas Akhir sebelumnya telah dikembangkan sebuah ERP bernama EZERP yaitu ERP yang dikembangan pada tahun 2016 dengan menggunakan *database* MySQL. Dalam pengembangannya ERP yang sedang dikembangakan saat ini memberikan sebuah inovasi terbaru dengan menggunakan *database* MongoDB yang bertipe NoSQL. Diharapkan dengan pembaruan ini ERP akan lebih baik dalam memenuhi kebutuhan yang selalu berkembang. Fokus dalam pengembangan inovasi baru ini diharapkan database MongoDB yang diterapkan mampu memberikan kontribusi dalam hal performa, fleksibiliti, dan skalabiliti.

Permasalahan ERP sebelumnya dengan menggunakan *dabatase* MySQL jika setiap *tenant* melakuakan *store* data dengan skema yang berbeda maka akan terjadi pemborosan *field* yang ada. Dalam satu tabel akan ada beberapa *field* yang kosong karena perbedaan kebutuhan dalam setiap skema. Permasalahan ini akan coba diatasi dengan menggunakan *database* MongoDB yang mana bentuk penyimpanan datanya yang berupa BSON (Binary JSON) akan fleksibel menerima setiap data.

Dalam ERP yang sedang dikembangkan saat ini juga menawarkan performa kecepatan penyimpanan data yang lebih optimal. Mengingat kemampuan database NoSQL yang lebih cepat dalam pemrosesan data. Dikarenakan sifat NoSQL yang *schemaless* sehingga saat melakukan transaksi data tidak memerlukan pengecekan seperti *primary key*, *foreign key*, dan lain sebagainya.

2.2 ERP *(Enterprise Resource Planning)*

ERP adalah sebuah sistem diperuntukan bagi perusahaan manufaktur maupun jasa yang berperan mengintegrasikan dan mengotomasikan proses bisnis yang berhubungan dengan aspek operasi, produksi, maupun distribusi di perusahaan bersangkutan[1].

Sistem ERP berfokus pada integrasi semua organisasi departemen, fungsi dan proses dalam satu komputer sistem informasi, mampu mendukung dalam semua bidang, dengan setiap kebutuhan individu dan spesifik mereka. Setiap departemen memiliki subsistem informasi komputer sendiri, dioptimalkan sesuai dengan karakteristik yang lebih spesifik dari aktivitas pengembangan, namun sistem ERP menggabungkan semua karakteristik dalam program perangkat lunak terpadu yang berjalan pada database yang unik sehingga semua organisasi bisa berbagi informasi dengan lebih baik dan dengan demikian mampu mengkomunikasikan informasi dengan baik [2].

2.3 ERP Retail

Sistem Enterprise Resource Planning (ERP) mengacu pada paket perangkat lunak yang mengintegrasikan semua data dan proses organisasi yang terkait. Dirancang untuk memudahkan optimalisasi proses bisnis internal di seluruh perusahaan, paket ERP telah menjadi alat yang kompetitif bagi organisasi ritel besar.

Perangkat lunak ERP menggunakan database tunggal yang memungkinkan departemen berbeda berkomunikasi satu sama lain. Sistem ERP terdiri dari fitur fungsi khusus yang dirancang untuk berinteraksi dengan modul lain seperti Order Entry, Accounts Payable, Accounts Receivable, Purchasing, dll. Paket perangkat lunak ERP Ritel memiliki jangkauan keseluruhan perusahaan yang menawarkan kemampuan lintas fungsional kepada organisasi.

2.4 MongoDB *(Database Management System)*

Pada tahun 1998 pertama kalinya dikembangkan sebuah konsep penyimpanan basis data yaitu NoSQL oleh Carlo Strozzi, yang kemudian pada tahun 2009 Eric Evans memperkenalkan kembali teknologi NoSQL. Kehadiran NoSQL bukan berarti untuk menggantikan model RDBMS yang sudah ada. Awal kemunculannya dilatarbelakangi oleh beberapa masalah yang muncul dari RDBMS. NoSQL dan RDBMS memiliki kelebihan dan tempat masing-masing sehingga diharapkan dapat saling melengkapi teknologi penyimpanan basis data.[3]

MongoDB adalah salah satu produk database NoSQL open source yang menggunakan struktur data JSON untuk menyimpan datanya. MongoDB adalah sebuah database yang menggunakan konsep manajemen database berorientasi dokumen (document-oriented) yang dibuat menggunakan pemograman C++. Orientasi dokumen ini adalah sebuah program komputer yang dirancang untuk menyimpan, mengambil, dan mengelola data yang berorientasi dokumen..MongoDB adalah salah satu database NoSQL yang paling populer di internet. MongoDB sering dipakai untuk aplikasi berbasis Cloud, Grid Computing, atau Big Data.

Performa pada MongoDB sudah mencapai 4 kali lebih cepat dibandingkan dengn MySQL serta mudah diaplikasikan juga. Karena MongoDB ini tergabung sebagai modul PHP.

Untuk pengguna RDBMS yang mungkin agak membingungkan, karena dalam MongoDB ini tidak terdapat namanya tabel akan tetapi yang digunakan hanyalah koleksi dan dokumen. Koleksi ini dianggap sebagai sebuah directory (folder) sedangkan dokumen sendiri dianggap sebagai file (berkas) dalam directory (folder) tersebut. Perbandingan pada RDBMS seperti MySQL, pada koleksi diibaratkan dengan tabel, sedangkan dokumen diibaratkan dengan baris dalam tabel tersebut. Baris pada MongoDB ini tidak sama dengan yang ada pada RDBMS, dokumen pada MongoDB dapat memiliki beda atribut dengan dokumen yang lainnya walaupun ada pada satu koleksi.

Pada MongoDB dokumen disimpan dalam bentuk BSON (Binary JSON) [4]. Dengan struktur yang mirip dengan JSON membuatnyanya cukup mudah untuk dibaca. Dengan konsep key-value pada MongoDB, setiap dokumen akan otomatis memiliki index id yang unik. Sehingga dapat membantu mempercepat proses pencarian data secara global. *Database* jenis NoSQL dan SQL juga memiliki banyak perbedaan. Adapun perbedaan keduanya dijelaskan pada **Tabel 2.1**.

Tabel 2.1 Perbedaan *Database* NoSQL dan SQL

No.	SQL	NoSQL
Model	Relasional	Non-relasional
Penyimpanan data	Menyimpan data dalam tabel yang memiliki baris dan kolom	Menyimpan data dalam berbagai bentuk seperti dokumen, *column stores*, grafik
Skema	Data SQL memiliki skema tetap dan sangat sulit untuk dirubah	NoSQL adalah database yang bebas skema dan memungkinkan pengguna untuk mengubah skema data dengan mudah
Struktur data	Cocok untuk data yang terstruktur	Lebih cocok untuk data yang semi terstruktur atau yang tidak terstruktur
Kolom/fields	Menambahkan kolom/fields baru dalam tabel mungkin memerlukan perubahan skema	Kolom/fields baru bisa ditambah dengan lebih mudah
Skalabilitas	Baik untuk penskalaan vertikal	Lebih cocok untuk penskalaan horizontal
Konsistensi	Konsistensi yang kuat	Konsistensi yang eventual

2.4.1 *Performance*

Dalam kinerja dan eksekusi berbagai jenis operasi, database NoSQL terbagi menjadi dua kategori. Yaitu *read* dan *write* yang dioptimasi.[5]

Performa yang lebih cepat dikarenakan NoSQL database saat melakukan penyimpanan data tidak memerlukan pengecakan struktur data maupun *primary key* dan *foreign key.*

2.4.2 *Flexibility*

MongoDB menyimpan data dalam format dokumen menggunakan JSON. Namun dalam MongoDB memiliki format yang lebih spesifik yaitu BSON (Binary JSON). Ini adalah skema dokumen dan pemetaan untuk jenis bahasa pemrograman asli [6].

Format JSON dalam 1 record ditandai dengan kurung kurawal, selanjutnya ditulis nama field dan diikuti dengan titik dua , baru dikuti dengan nilai dari kolom tersebut. Secara umum formatnya sebagai berikut:

{[nama Field] : [Nilai Field] , [nama Field2] : [Nilai Field]…}

Contoh : {NIM : '123456', Nama : 'Supriyanto' , Isaktif : true}

Tipe data yang disupport oleh BSON/JSON antara lain:

- String
- Integer
- Boolean
- Arrays

Gambar 2.1 *Document* BSON

2.4.3 *Scalability*

Skalabilitas menyangkut kemampuan sistem untuk mengatasi beban kerja yang meningkat. Dalam konteks database, ini dapat didefinisikan sebagai perubahan kinerja saat *node* baru ditambahkan, atau perangkat keras ditingkatkan. Database NoSQL telah dikembangkan secara khusus untuk menargetkan skenario dimana skalabilitas sangat penting. Sistem ini bergantung pada skalabilitas horizontal dan "elastis", dengan menambahkan lebih banyak *node* ke sistem daripada mengupgrade perangkat keras. Istilah "elastis" mengacu pada elastisitas, yang merupakan karakterisasi dari cara cluster bereaksi terhadap penambahan atau penghilangan *node*[5].

Ada dua metode untuk menangani pertumbuhan sistem, yaitu:

- *Vertical Scaling*

Vertical Scaling melibatkan peningkatan kapasitas pada server dengan cara seperti meningkatkan CPU, menambah RAM, dan memperbesar memori.

- *Horizontal Scaling*

Horizontal Scaling bekerja dengan cara menggunakan lebih dari satu server, dimana jika server utama mulai terasa berat maka data akan di distribusikan ke server yang lainnya.

2.5 *Multitenant Data Architecture* atau *Multitenancy*

Arsitekur data pada sebuah sistem perangkat lunak yang mana setiap tenant memungkinkan untuk mengakses data miliknya dalam sebuah *database* yang sama. Arsitektur seperti demikian dirancang untuk *multitenancy* yang mana setiap tenant berhak atas *sharing* data dari database, manajemen konfigurasi, pengguna, proses bisinis, dan lain sebagainya[7][8].

Arsitektur data seperti ini sangat cocok digunakan dan cukup aman dalam mengatasi permasalahan kurangnya kepercayaan *tenant* untuk menyerahkan kontrol data pada pihak ketiga, dan juga cukup efektif dan efisien dalam pengelolaan dan pemeliharaan. Adapun arsitektur data *multitenant* terbagi menjadi tiga, yaitu:

2.5.1 *Separated Database, Separated Schema*

Pada arsitektur ini setiap *tenant* berhak atas satu database yang dia kelola sendiri dan tidak dapat diakses oleh tenant lain. Namun kekurangan arsitektur ini yaitu memerlukan biaya yang lebih untuk membeli server, juga untuk perawatan kedepannya. Juga biaya tambahan untuk *back-up* data atau pendistribusian data.

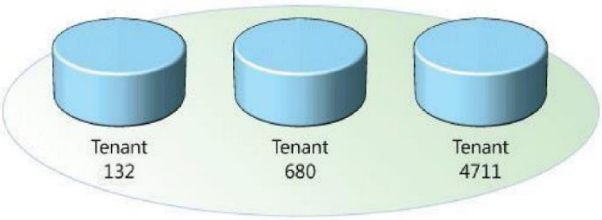
2.5.2 *Shared Database, Separated Schema*

Pada arsitektur ini, tiap tenant berhak mengakses *database* yang sama. Namun setiap tanant dibatasi dengan *schema* masing-masing. Sehingga dalam satu *database* berisi banyak *schema* tiap tenant. Dalam kasus ini jika menggunakan SQL database maka akan terjadi pemborosan *field* dikarenakan sifat *database* SQL yang konsisten.

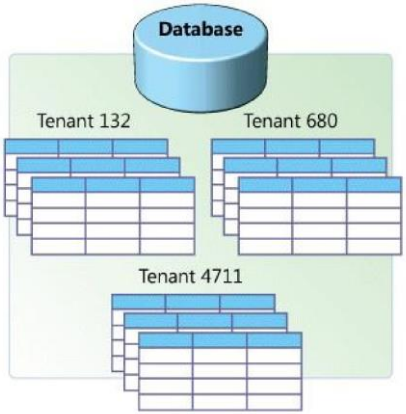
2.5.3 *Shared Database, Shared Schema*

Pada arsitektur ini setiap *tenant* berhak mengakses satu *database* yang mana setiap record akan ditandai dengan *id tenant* yang melakukan transaksi data. Arsitektur semacam ini cukup menghemat biaya. Namun akan cukup rumit untuk mengeksplor data dikarenakan data yang rumit hasil penyimpanan setiap *schema tenant* yang berbeda-beda.

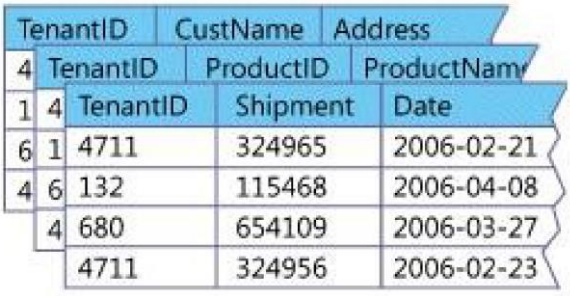
.



Gambar 2.2 Separated Schema, Separated Database [9]



Gambar 2.3 Shared Database, Separated Schema [9]



Gambar 2.4 Shared Schema, Shared Database [9]

2.6 *MySQL*

MySQL adalah sistem manajemen basis data relasional open-source (RDBMS) yang sangat popular. Yang mana didistribusikan, dikembangkan, dan didukung oleh Oracle Corporation. Sistem relasional seperti MySQL menyimpan data dalam bentuk tabel dan menggunakan bahasa query terstruktur (SQL) untuk mengakses data. Di MySQL, kita harus mendefinisikan skema berdasarkan persyaratan dan mengatur peraturan untuk mengendalikan hubungan antar bidang dalam catatan. Di MySQL, informasi terkait dapat disimpan dalam tabel yang berbeda, namun terkait dengan penggunaan bergabung. Dengan demikian, duplikasi data bisa diminimalisir [10].

2.7 *Robomongo*

Robomongo adalah tool manajemen MongoDB. Tidak seperti kebanyakan alat admin UI MongoDB lainnya, Robomongo menanamkan shell mongo sebenarnya di antarmuka dengan akses ke *command line shell* dan juga interaksi GUI.

2.8 *Phpmyadmin*

Phpmyadmin adalah perangkat lunak bebas yang ditulis dalam bahasa pemrograman PHP yang digunakan untuk menangani administrasi MySQL melalui Jejaring Jagat Jembar (World Wide Web). Phpmyadmin mendukung berbagai operasi MySQL, diantaranya (mengelola basis data, tabel-tabel, bidang (fields), relasi (relations), indeks, pengguna (users), perizinan (permissions), dan lain-lain)[1].

2.9 *Sharding*

Sharding adalah metode untuk mendistribusikan data ke beberapa mesin. MongoDB menggunakan sharding untuk

[1] https://id.wikipedia.org/wiki/PhpMyAdmin

mendukung penyebaran dengan kumpulan data yang sangat besar dan melalui operasi yang tinggi [2]

2.10 PHP (*Hypertext Preprocessor*)

PHP adalah singkatan dari *Hypertext Preprocessor* yakni salah satu bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML. PHP diciptakan oleh Rasmus Lerdorf pertama kali tahun 1994. Pada awalnya PHP adalah singakatan dari Personal Home Page Tools. Selanjutnya diganti FI (Forms Interpreter). Sejak versi 3.0, nama bahasa ini diubah menjadi PHP (Hypertxt Preprocessor) dengan singkatan PHP. PHP versi terbaru adalah versi ke-7[3].

2.11 Kerangka Kerja Yii2

Sebuah *framework* untuk bahasa pemrograman PHP dimana *framework* ini telah didukung oleh aspek *modularity* dan *eloquent database*. Arsitektur perangkat lunak ini menggunakan *Model-View-Controller*, namun dengan aspek *modularity*, maka dapat dibuat modul dimana setiap modul memiliki MVC tersendiri. Sedangkan *eloquent database* adalah basisdata berupa objek di dalam sebuah bahasa pemrograman PHP[4].

[2] https://docs.mongodb.com/manual/sharding/
[3] https://id.wikipedia.org/wiki/PHP
[4] http://www.yiiframework.com/doc-2.0/guide-intro-yii.html

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis dan perancangan sistem yang akan dibangun. Analisis membahas semua persiapan yang akan menjadi pokok pikiran pembuatan aplikasi ini. Mulai dari masalah yang melatarbelakangi, hingga analisis gambaran awal sistem yang akan dibuat. Perancangan sistem membahas hal-hal yang berkaitan dengan pondasi atau dasar pembuatan aplikasi, yang meliputi perancangan basis data, tampilan antar muka halaman aplikasi, hingga perancangan alur proses yang akan diimplementasikan di dalam aplikasi.

3.1 Analisis

Tahap analisis meliputi analisis masalah, analisis kebutuhan, deskripsi umum sistem, dan kasus penggunaan sistem yang dibuat.

3.1.1 Analisis Permasalahan

Dewasa ini perkembangan teknologi berkembang sangat pesat. Khususnya kebutuhan akan sistem informasi yang seakan-akan terus bersaing memberikan inovasi-inovasi baru agar bisa bersaing satu sama lain. Salah satu aspek yang paling penting dari sebuah sistem informasi adalah pengelolaan data. Semua developer berusaha untuk selalu mengoptimasi jenis database apa maupun *query* seperti apa yang harus digunakan agar aplikasi berjalan secara optimal.

Pada penelitian kali ini akan menganalisa mana database yang lebih *powerfull* untuk digunakan dalam kasus ERP Retail. Jika beberapa ERP sebelumnya yang telah dibuat kebanyakan menggunakan database SQL, maka tantangan kedepan adalah bagaimana menutupi kelemahan SQL agar aplikasi ERP bisa lebih baik dari sebelumnya. Khususnya dalam hal performa, fleksibilitas, dan skalabilitas. Maka dalam kesempatan kali ini inovasi yang akan dibuat adalah bagaimana jika ERP Retail yang

sekarang menggunakan *database non-relational* (NoSQL). Seperti yang telah kita ketahui, *database* NoSQL memiliki keunggulan dalam hal performa dan fleksibilitas yang tidak dimiliki oleh database SQL.

3.1.2 Spesifikasi Kebutuhan Analisa

Pada proses penelitian untuk membandingkan kemampuan *database* SQL dengan *database* NoSQL maka dibutuhkan beberapa

3.1.2.1 MySQL

Dalam penelitian ini digunakan database MySQL untuk mewakili *database* tipe SQL. Versi yang digunakan yaitu MySQL v5.0.12-dev - 20150407

3.1.2.2 MongoDB

Dalam penelitian ini digunakan database MongoDB untuk mewakili *database* tipe NoSQL. Adapun spesifikasi MongoDB yang digunakan yaitu:

- Versi : 1.2.8
- SSL Library : OpenSSL
- Crypto Library : libcrypto

3.1.2.3 *Dataset*

Dalam penelitian ini akan digunakan dua jenis *dataset* yaitu JSON untuk MongoDB dan SQL untuk MySQL.

Dataset akan dibagi secara bertahap. Untuk uji coba *insert* data dataset dengan jumlah yang sama akan di *insert* secara kontinyu. Misal, *insert* pertama menggunakan 1000 data. Kemudian dilakukan secara berulang untuk *insert* beritutnya. Dari data 1, 2, 3, 4, 5, ..8. Sedangkan untuk uji coba *select*, *update*, dan *delete* dilakukan testing mulai dari data 1000 naik bertahap dengan 2000, 3000, 4000, 5000, ...8000.

Berikut adalah contoh tipe dataset yang akan digunakan untuk percobaan:

- JSON

[{"id":1,"nama_supplier":"Dian Wahyu","email_supplier":"dianwahyu@gmail.com","alamat_supplier":"Jl. Sudimoro 35 Malang","telp_supplier":"081927638372","id_retail":1},{"id":2,"nama_supplier":"Johan Akbar","email_supplier":"johanakbar@gmail.com","alamat_supplier":"Jl Menur 87 RT 11 RW 07 Surabaya","telp_supplier":"085658368283","id_retail":2},{"id":3,"nama_supplier":"Donny Alam","email_supplier":"donnyalam@gmail.com","alamat_supplier":"Jl. Martapura 98 Mojokerto","telp_supplier":"08226374356","id_retail":3},{"id":4,"nama_supplier":"Rudi Hardi","email_supplier":"rudihardi@gmail.com","alamat_supplier":"Jl. Antasari 65 Surabaya","telp_supplier":"081726372830","id_retail":4},{"id":5,"nama_supplier":"Joko Lesmono","email_supplier":"jokolesmono@gmail.com","alamat_supplier":"Jl. Menur 35 Surabaya","telp_supplier":"081738287329","id_retail":5},{"id":6,"nama_supplier":"Lusi Maharani","email_supplier":"luusi maharanii@yahoo.com","alamat_supplier":"Jl.Kendalisada 89Surabaya","telp_supplier":"0852686382238","id_retail":6}]

Gambar 3.1 Contoh format *dataset* JSON universitas

- SQL

```
insert into supplier (id, nama_supplier, email_supplier, alamat_supplier, telp_supplier, id_retail) values (1, ' Dian Wahyu, ' dianwahyu@gmail.com ', ' Jl. Sudimoro 35 Malang ', '081927638372', 1);
insert into supplier (id, nama_supplier, email_supplier, alamat_supplier, telp_supplier, id_retail) values (2, 'Johan Akbar', ' johanakbar@gmail.com ', ' Jl Menur 87 RT 11 RW 07 Surabaya ', '085658368283', 2);
insert into supplier (id, nama_supplier, email_supplier, alamat_supplier, telp_supplier, id_retail) values (3, 'Donny Alam', 'donnyalam@gmail', ' Jl. Martapura 98 Mojokerto ', '08226374356', 3);
insert into supplier (id, nama_supplier, email_supplier, alamat_supplier, telp_supplier, id_retail) values (4, 'Rudi Hardi', 'rudihardi@gmail.com', ' Jl. Antasari 65 Surabaya ', '081726372830', 4);
insert into supplier (id, nama_supplier, email_supplier, alamat_supplier, telp_supplier, id_retail) values (5, 'Joko Lesmono', 'jokolesmono@gmail.com', ' Jl. Menur 35 Surabaya ', '081738287329', 5);
insert into supplier (id, nama_supplier, email_supplier, alamat_supplier, telp_supplier, id_retail) values (6, 'Lusi Maharani', ' luusi maharanii@yahoo.com ', ' Jl.Kendalisada89Surabaya ', '0852686382238', 6);
insert into supplier (id, nama_supplier, email_supplier,
```

Gambar 3.2 Contoh format *dataset* SQL universitas

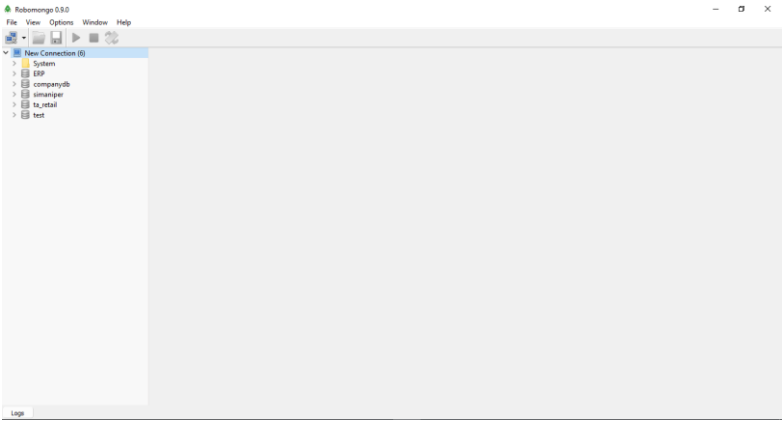
3.1.2.4 GUI (*Graphical User Interface*) *database*

Fungsi GUI *database* dalam penelitian ini adalah agar mempermudah *user* untuk memantau segala transaksi perubahan data dalam database. Selain itu GUI membantu *user* mengetahui *runtime* dari setiap transaksi. *Runtime* dibutuhkan sebagai data

yang akan diolah untuk membandingkan kemampuan antar *database*.

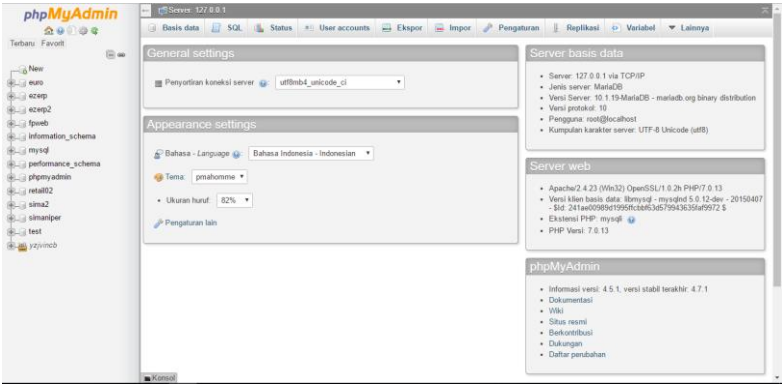
Dalam penelitian ini akan digunakan *tools* Robomongo sebagai GUI MongoDB dan phpMyAdmin sebagai GUI MySQL.

- Robomongo



Gambar 3.3 Penampilan *Interface* GUI Robomongo

- phpMyAdmin



Gambar 3.4 Penampilan *Interface* GUI phpMyAdmin

3.2 Perancangan Sistem

Perancangan sistem dilakukan untuk menggambarkan rancangan dan arsitektur yang akan digunakan untuk penelitian. Juga menggambarkan bagaimana proses-proses yang harus dijalani untuk mendapatkan hasil. Prose-proses ini akan dipresentasikan dengan diagram alir (*flowchart)* pada setiap uji coba yang dilakukan.

Perangkat keras

Lingkungan implementasi perangkat keras untuk pengerjaan tugas akhir ini sebagai berikut:

- Komputer:
  - Prosesor : Intel® Core™ i5-323M CPU (2.60GHz)
  - RAM : 4 GB
  - System Type : 64-bit Operating System

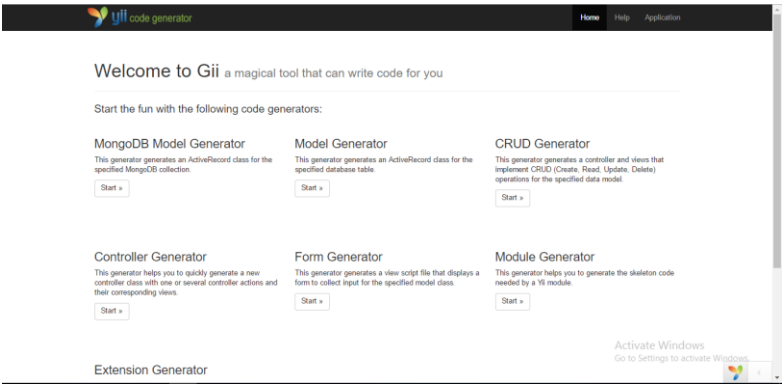
Perangkat lunak

Lingkungan implementasi perangkat lunak untuk pengerjaan tugas akhir ini sebagai berikut:

- Sistem operasi : Ubuntu 16.04 LTS, Windows 10 Pro
- IDE : Sublime text Editor, Robomongo, Phpmyadmin

3.2.1 Perancangan *Backend* Web

Pada tahap ini akan dijelaskan perancangan *backend* untuk membangun aplikasi ERP Retail. Dalam membangun aplikasi akan digunakan *Framework* Yii2. Dalam penggunaannya *framework* ini memiliki generator Gii, yaitu fitur untuk *generate* proses CRUD secara otomatis. Jadi dengan fitur ini user sudah disediakan *template* mulai dari controller, view, dan model setiap modul.



Gambar 3.5 Penampilan *Interface* Gii Yii2 Framework

3.2.2 Spesifikasi Kebutuhan Perngkat Lunak

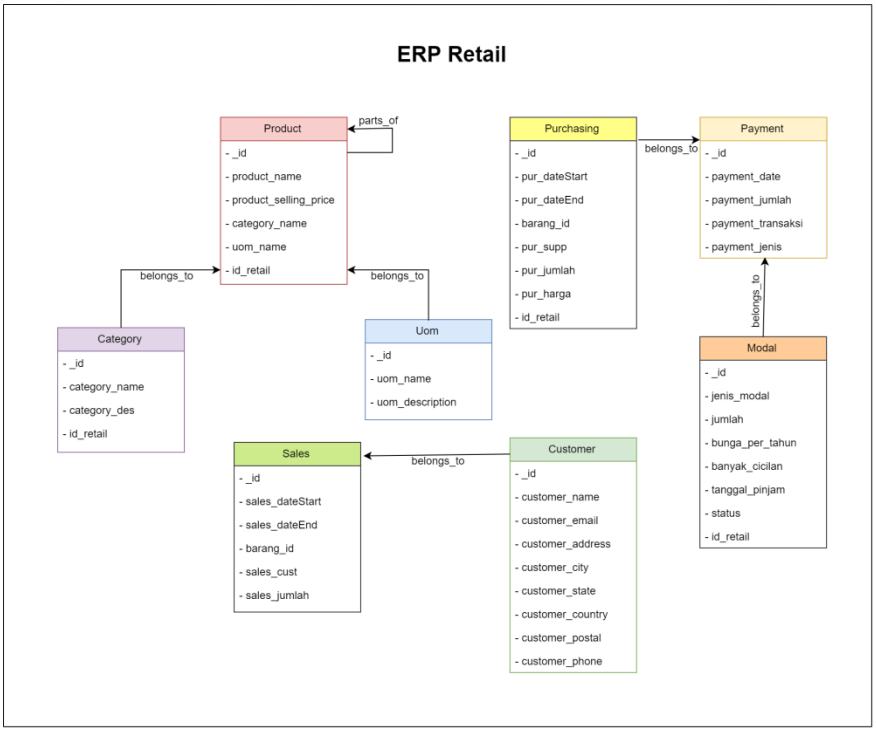
Spesifikasi kebutuhan dalam sistem ini mencakup kebutuhan fungsional. Kebutuhan fungsional berisikan proses-proses yang dibutuhkan dalam sistem dan harus dijalankan. Kebutuhan fungsional sistem dideskripsikan dalam **Tabel 3.2**

Tabel 3.1 Daftar Kebutuhan Fungsional Sistem

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-001	*Manage Category*	Pengguna dapat mengelola *category* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-002	*Manage Product*	Pengguna dapat mengelola *product* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-003	*Manage Uom*	Pengguna dapat mengelola *uom* yang meliputi proses: *create*,

		read, *update*, dan *delete*
F-004	*Manage Supplier*	Pengguna dapat mengelola *supplier* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-005	*Manage Customer*	Pengguna dapat mengelola *customer* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-006	*Manage Delivery*	Pengguna dapat mengelola *delivery* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-007	*Manage Modal*	Pengguna dapat mengelola *modal* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-008	*Manage Staff*	Pengguna dapat mengelola *staff* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-009	*Manage Inventory*	Pengguna dapat mengelola *inventory* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-010	*Manage Sales*	Pengguna dapat mengelola *sales* yang meliputi proses: *create*, *read*, *update*, dan *delete*
F-011	*Manage Purchasing*	Pengguna dapat mengelola *purchasing*

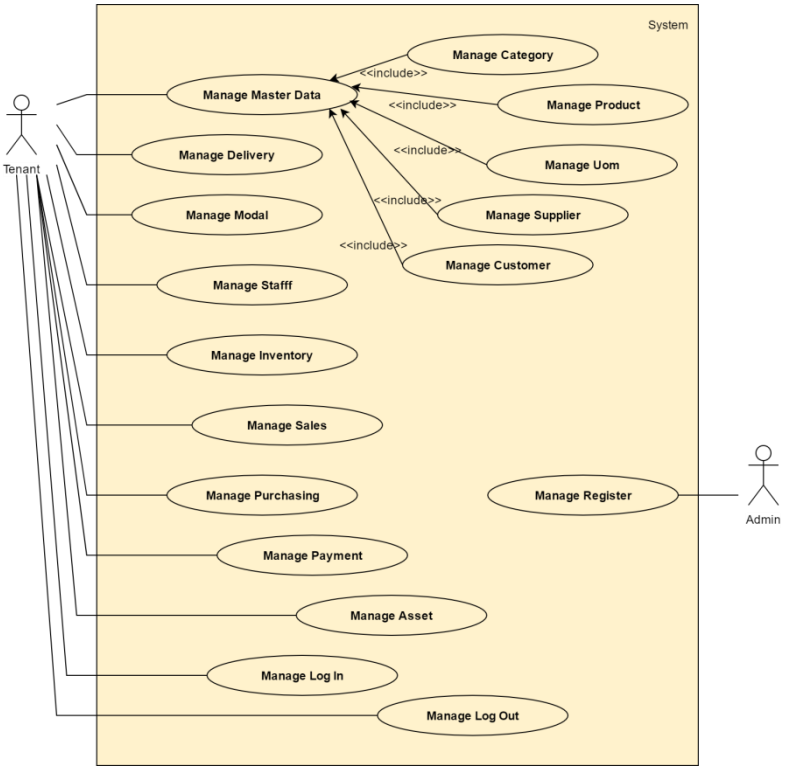
		yang meliputi proses: *create, read, update,* dan *delete*
F-012	*Manage Payment*	Pengguna dapat mengelola *payment* yang meliputi proses: *create, read, update,* dan *delete*
F-013	*Manage Asset*	Pengguna dapat mengelola *asset* yang meliputi proses: *create, read, update,* dan *delete*
F-014	*Register*	Pengguna dapat melakukan pendaftaran user baru.
F-015	*Log In*	Pengguna dapat melakukan *log in*
F-016	*Log Out*	Pengguna dapat melakukan Log Out.



Gambar 3.6 Skema MongoDB ERP Retail

3.2.3 Kasus Penggunaan

Kasus penggunaan yang dibutuhkan pada sistem sesuai dengan analisa yang telah dilakukan. Diagram kasus penggunaan dapat dilihat pada **Gambar 3.7** dan kode kasus penggunaan ada pada **Tabel 3.3**.



Gambar 3.7 Diagram Kasus Penggunaan

Tabel 3.2 Keterangan Kode Kasus Penggunaan

Kode Kasus Penggunaan	Kasus Penggunaan	Aktor
UC-001	*Manage Category*	Tenant
UC-002	*Manage Product*	Tenant
UC-003	*Manage Uom*	Tenant
UC-004	*Manage Supplier*	Tenant
UC-005	*Manage Customer*	Tenant
UC-006	*Manage Delivery*	Tenant
UC-007	*Manage Modal*	Tenant

UC-008	*Manage Staff*	Tenant
UC-009	*Manage Inventory*	Tenant
UC-010	*Manage Sales*	Tenant
UC-011	*Manage Purchasing*	Tenant
UC-012	*Manage Payment*	Tenant
UC-013	*Manage Asset*	Tenant
UC-014	*Register*	Admin
UC-015	*Log In*	Tenant
UC-016	*Log Out*	Tenant

3.2.3.1 UC-001 *Manage Category*

Tabel 3.3 Spesifikasi Kasus Penggunaan *Manage Category*

Nama	*Manage Category*
Nomor	UC-001
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *category*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Category*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* memilih submenu *category* yang terdapat pada menu *Master Data* 4. Sistem menampilkan halaman utama pada submenu *category* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru

	2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.2 UC-002 *Manage Product*

Tabel 3.4 Spesifikasi Kasus Penggunaan *Manage Product*

Nama	*Manage Product*
Nomor	UC-002
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *product*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Product*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* memilih submenu *product* yang terdapat pada menu *Master Data* 4. Sistem menampilkan halaman utama pada submenu *product* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru

	A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.3 UC-003 *Manage Uom*

Tabel 3.5 Spesifikasi Kasus Penggunaan *Manage Uom*

Nama	*Manage Uom*
Nomor	UC-003
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *uom*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Uom*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi

	3. Pemilik *retail* memilih submenu *uom* yang terdapat pada menu *Master Data* 4. Sistem menampilkan halaman utama pada submenu *uom* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.4 UC-004 *Manage Supplier*

Tabel 3.6 Spesifikasi Kasus Penggunaan *Manage Supplier*

Nama	*Manage Supplier*
Nomor	UC-004
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *supplier*

Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Supplier*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* memilih submenu *supplier* yang terdapat pada menu *Master Data* 4. Sistem menampilkan halaman utama pada submenu *supplier* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.5 UC-005 *Manage Customer*

Tabel 3.7 Spesifikasi Kasus Penggunaan *Manage Customer*

Nama	*Manage Customer*
Nomor	UC-005
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *customer*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Customer*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* memilih submenu *customer* yang terdapat pada menu *Master Data* 4. Sistem menampilkan halaman utama pada submenu *customer* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3

	A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.6 UC-006 *Manage Delivery*

Tabel 3.8 Spesifikasi Kasus Penggunaan *Manage Delivery*

Nama	*Manage Delivery*
Nomor	UC-006
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *delivery*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Delivery*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* memilih menu *delivery* 4. Sistem menampilkan halaman utama pada submenu *delivery* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3

	A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.7 UC-007 *Manage Modal*

Tabel 3.9 Spesifikasi Kasus Penggunaan *Manage Modal*

Nama	*Manage Modal*
Nomor	UC-007
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *modal*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Modal*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* menu *modal* 4. Sistem menampilkan halaman utama pada submenu *modal* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3

	A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.8 UC-008 *Manage Staff*

Tabel 3.10 Spesifikasi Kasus Penggunaan *Manage Staff*

Nama	*Manage Staff*
Nomor	UC-008
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *staff*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Staff*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* menu *staff* 4. Sistem menampilkan halaman utama pada submenu *staff* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir

Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.9 UC-009 *Manage Inventory*

Tabel 3.11 Spesifikasi Kasus Penggunaan *Manage Inventory*

Nama	*Manage Inventory*
Nomor	UC-009
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *inventory*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Inventory*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* menu *inventory* 4. Sistem menampilkan halaman utama pada submenu *inventory* 5. Pemilik retail akan memilih kegiatan yang dapat

	dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.10 UC-010 *Manage Sales*

Tabel 3.12 Spesifikasi Kasus Penggunaan *Manage Sales*

Nama	*Manage Sales*
Nomor	UC-010
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *sales*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Sales*

Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* menu *sales* 4. Sistem menampilkan halaman utama pada submenu *sales* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.11 UC-011 *Manage Purchasing*

Tabel 3.13 Spesifikasi Kasus Penggunaan *Manage Purchasing*

Nama	*Manage Purchasing*
Nomor	UC-011
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data

	purchasing
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Purchasing*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* menu *purchasing* 4. Sistem menampilkan halaman utama pada submenu *purchasing* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.12 UC-012 *Manage Payment*

Tabel 3.14 Spesifikasi Kasus Penggunaan *Manage Payment*

Nama	*Manage Payment*
Nomor	UC-012
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *payment*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Payment*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* menu *payment* 4. Sistem menampilkan halaman utama pada submenu *payment* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu

	1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.13 UC-013 *Manage Asset*

Tabel 3.15 Spesifikasi Kasus Penggunaan *Manage Asset*

Nama	*Manage Asset*
Nomor	UC-013
Deskripsi	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *asset*
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman salah satu submodul dari *Asset*
Alur Normal	1. Pemilik *retail* membuka halaman utama aplikasi 2. Sistem menampilkan halaman utama aplikasi 3. Pemilik *retail* menu *asset* 4. Sistem menampilkan halaman utama pada submenu *asset* 5. Pemilik retail akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih melihat data tertentu 6. Kasus penggunaan berkahir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke langkah normal alur 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tertentu 2. Pengguna menyunting data tersebut 3. Sistem membaharui data yang telah disunting 4. Berlanjut ke langkah normal alur 3 A3. Pengguna memilih menghapus data tertentu

	1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke langkah normal alur 3 A4. Pengguna memilih melihat data tertentu 1. Sistem menampilkan data tertentu 2. Berlanjut ke langkah normal alur 3

3.2.3.14 UC-014 *Manage Log In*

Tabel 3.16 Spesifikasi Kasus Penggunaan *Manage Log In*

Nama	*Manage Log In*
Nomor	UC-014
Deskripsi	Kasus penggunaan ini digunakan untuk *tenant* melakukan *log in* pada aplikasi retail
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna belum masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman utama dari ERP Retail
Alur Normal	1. Pemilik *retail* membuka halaman *log in* aplikasi 2. Sistem menampilkan *form* untuk *log in* 3. Pemilik mengisikan id dan *password* 4. Sistem menampilkan halaman utama aplikasi retail 5. Kasus penggunaan berkahir
Alur Alternatif	-

3.2.3.15 UC-015 *Manage Log Out*

Tabel 3.17 Spesifikasi Kasus Penggunaan *Manage Log Out*

Nama	*Manage Log Out*
Nomor	UC-015
Deskripsi	Kasus penggunaan ini digunakan untuk *tenant* melakukan *log out* pada aplikasi retail
Tipe	Fungsional
Aktor	Tenant
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman *log in* dari ERP Retail

Alur Normal	1. Pemilik *retail* mengeklik tombol *log out* aplikasi 2. Sistem menampilkan halaman *log in* 3. Kasus penggunaan berkahir
Alur Alternatif	-

3.2.3.16 UC-016 *Manage Register*

Tabel 3.18 Spesifikasi Kasus Penggunaan *Manage Register*

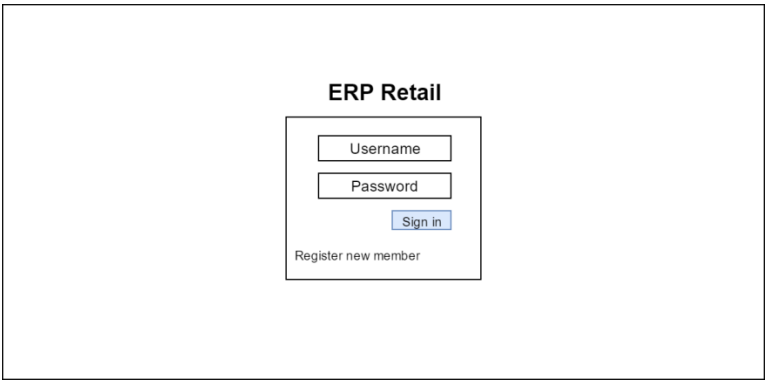
Nama	*Manage Register*
Nomor	UC-016
Deskripsi	Kasus penggunaan ini digunakan untuk *admin* saat mendaftarkan *tenant* baru
Tipe	Fungsional
Aktor	Admin
Kondisi Awal	Admin belum masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman utama dari ERP Retail
Alur Normal	1. Admin mengeklik tombol *Register* aplikasi 2. Sistem menampilkan *form* identitas *tenant* 3. Admin mengisikan identitas *tenant* baru 4. Sistem menampilkan halaman utama aplikasi 5. Kasus penggunaan berkahir
Alur Alternatif	-

3.2.4 Perancangan Tampilan Antarmuka

Pada tahap ini akan dijelaskan desain perancangan aplikasi ERP Retail. Perincian desain dijabarkan sebagai berikut:

3.2.4.1 Perancangan Antarmuka Login

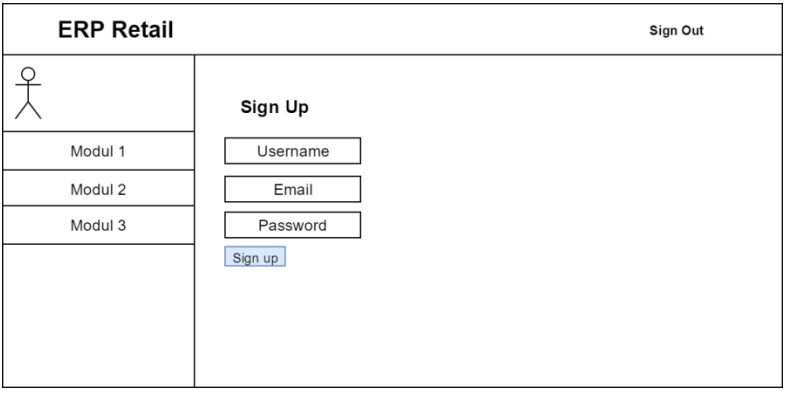
Perancangan antarmuka login terpusat pada form di tengah halaman untuk login bagi *user* yang sudah registrasi. Di bagian atas terdapat judul aplikasi dan di bawah form tersedia fitur untuk registrasi bagi *user* baru. Perancangan antarmuka login ditunjukkan pada **Gambar 3.7**



Gambar 3.8 Rancangan antarmuka login

3.2.4.2 Perancangan Antarmuka Menambahkan User Baru

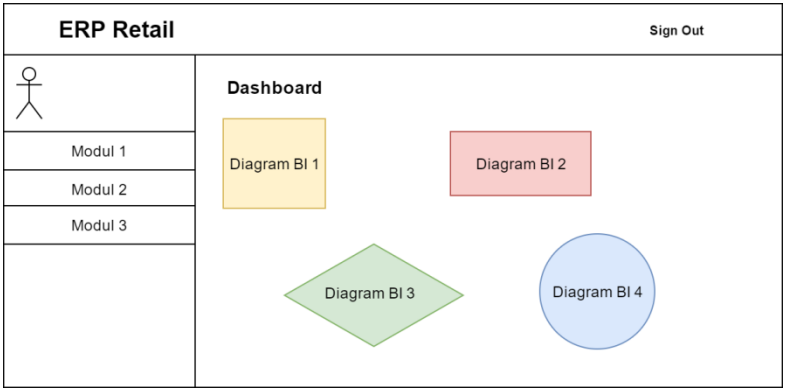
Perancangan antarmuka menambahkan user baru sudah dilengkapi dengan UI adminLTE. Di bagian atas terdapat nama aplikasi, fitur *log out*, dan notifikasi. Di bagian samping terdapat navigasi utama yang berisi sub-sub modul dari ERP Retail. Dan di bagian tengah terdapat form untuk registrasi yang berisi username, email, dan password. Perancangan antarmuka menambahkan user baru ditunjukkan pada **Gambar 3.8**



Gambar 3.9 Rancangan antarmuka menambahkan user baru

3.2.4.3 Perancangan Antarmuka Dashboard

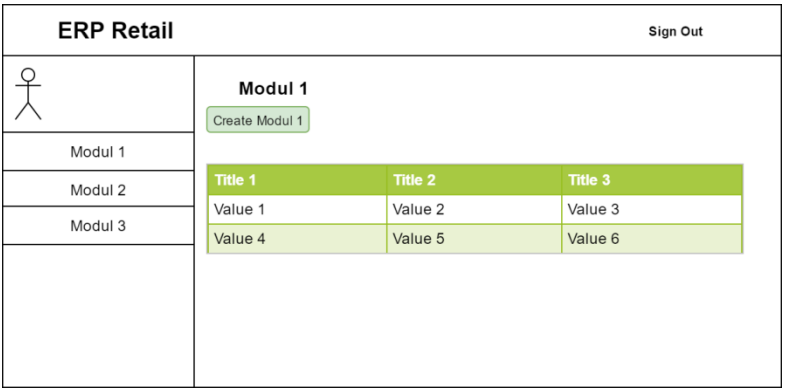
Perancangan antarmuka dashboard menggunakan UI adminLTE. Di bagian atas terdapat nama aplikasi, fitur *log out*, dan notifikasi. Di bagian samping terdapat navigasi utama yang berisi sub-sub modul dari ERP Retail. Dan di bagian tengah tampilan *business intelegent* (BI) berbentuk diagram-diagram. Perancangan dashboard ditunjukkan pada **Gambar 3.9**



Gambar 3.10 Rancangan antarmuka dashboard

3.2.4.4 Perancangan Antarmuka Submodul

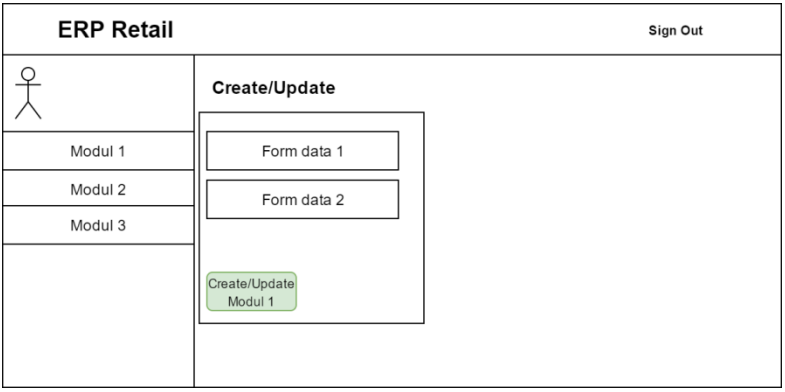
Perancangan antarmuka submodul menggunakan UI adminLTE. Di bagian atas terdapat nama aplikasi, fitur *log out*, dan notifikasi. Di bagian samping terdapat navigasi utama yang berisi sub-sub modul dari ERP Retail. Dan di bagian tengah menampilkan judul modul, *create* modul, dan list data modul. Perancangan antarmuka submodul ditunjukkan pada **Gambar 3.10**



Gambar 3.11 Rancangan antarmuka submodul

3.2.4.5 Perancangan Antarmuka *Create* dan *Update*

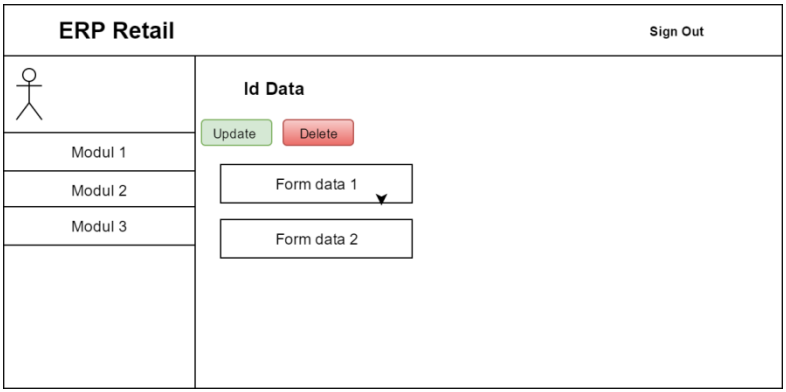
Perancangan antarmuka *create* dan *update* menggunakan UI adminLTE. Di bagian atas terdapat nama aplikasi, fitur *log out*, dan notifikasi. Di bagian samping terdapat navigasi utama yang berisi sub-sub modul dari ERP Retail. Dan di bagian tengah menampilkan judul *action* (*create* atau *update*) dan dibawahnya terdapat form data. Perancangan antarmuka *create* dan *update* ditunjukkan pada **Gambar 3.11**



Gambar 3.12 Rancangan antarmuka *create* dan *update*

3.2.4.6 Perancangan Antarmuka *View*

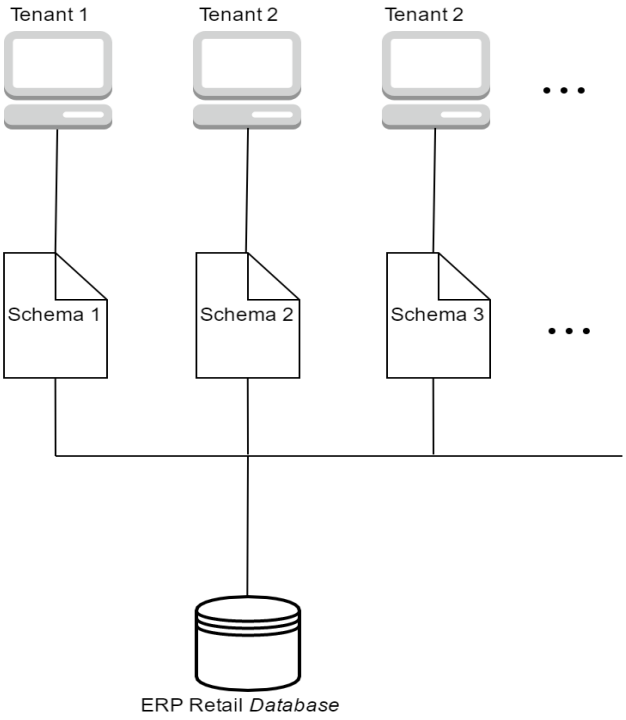
Perancangan antarmuka *view* menggunakan UI adminLTE. Di bagian atas terdapat nama aplikasi, fitur *log out*, dan notifikasi. Di bagian samping terdapat navigasi utama yang berisi sub-sub modul dari ERP Retail. Dan di bagian tengah menampilkan id data yang akan di eksekusi dibawahnya ada tombol *update* atau *delete* dan dibawahnya berisi informasi data. Perancangan antarmuka *view* ditunjukkan pada **Gambar 3.12**



Gambar 3.13 Rancangan antarmuka *view*

3.2.5 Perancangan *Multitenancy*

Pada tahap ini akan dijelaskan bagaimana desain *multitenant* pada aplikasi ERP Retail. Metode *multitenant* yang akan digunakan dengan cara menempelkan id tenant pada setiap transaksi data yang dilakukan oleh tenant. Sehingga dalam *interface* aplikasi data akan di filter berdasarkan id tenant yang melakukan transaksi.



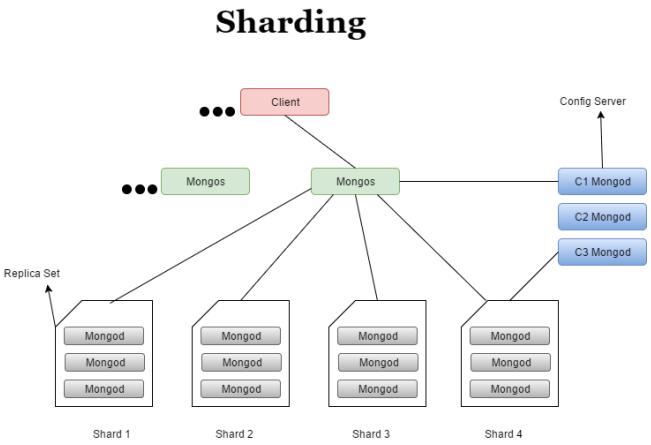
Gambar 3.14 Desain Perancangan *Multitenantcy*

3.2.5.1 Proses Distribusi *Database*

Dari perancangan *multitenancy* dengan metode *shared database shared schema*, maka data yang banyak dan terus berkembang akan menimbulkan beberapa permasalahan termasuk kapasitas *server* yang akan penuh. Oleh karena itu dalam tugas akhir ini diberikan inovasi untuk mendistribusikan data ke server lain agar *database* menjadi *scalable*.

Untuk *database* MongoDB sendiri mensupport ditribusi data dengan menggunakan metode sharding. Keunggulan metode sharding dari yang lain adalah metode ini memiliki *load balancer* yang otomatis. Dimana fungsinya untuk membagi data baru ke

server(*shard*) lain secara otomatis. Sehingga data dengan otomatis bisa terdistribusi.



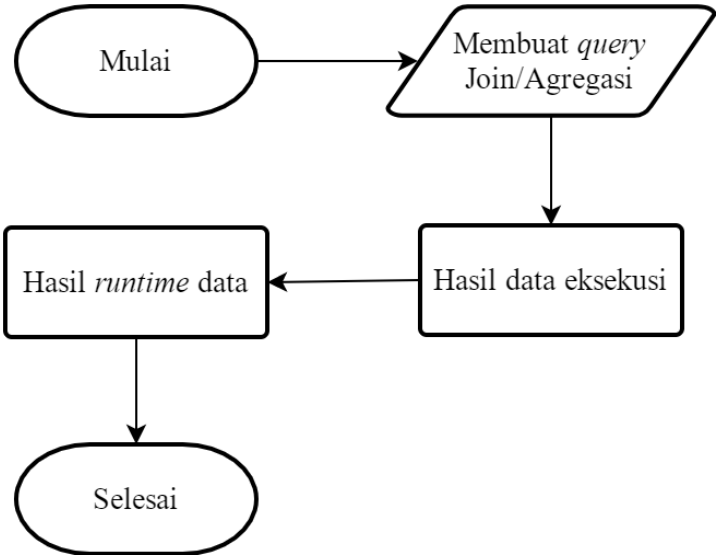
Gambar 3.15 Desain Perancangan *Sharding*

3.2.6 Perancangan Proses

Perancangan proses dilakukan untuk memberikan gambaran bagaimana proses-proses yang harus dilalui untuk melakukan penelitian. Dalam bagian ini akan dipresentasikan uji coba untuk membandingkan database jenis SQL dengan NoSQL dalam hal kemampuan *join*, *insert*, *update*, *select*, dan *delete*. Juga uji coba *flexibility* untuk membuktikan bahwa *database* NoSQL merupakan *database* yang *flexible.*

3.2.6.1 Proses Uji Coba *Join*/Agregasi

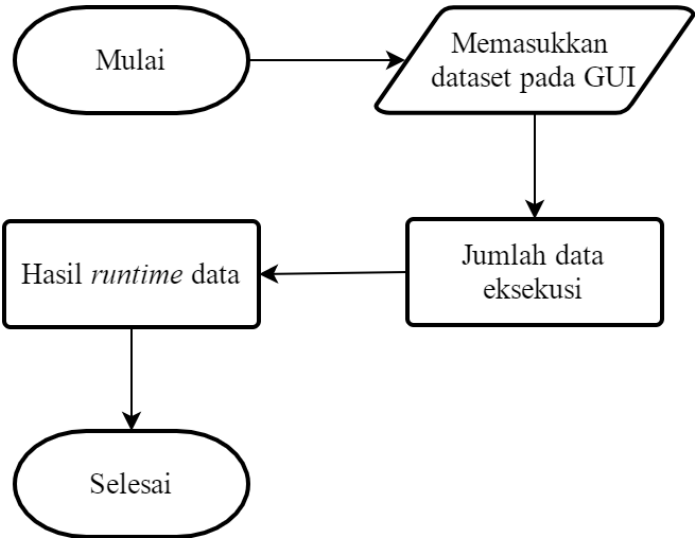
Proses ini dilakukan untuk membandingkan kemampuan "*Join*" atau "Agregasi" dari database SQL dengan database NoSQL. Tahap yang dilakukan antara lain membuat *query* join/agregasi, menampilkan hasil data transaksi, dan menampilkan *runtime* data. Diagram alir dari proses ini ditunjukkan pada **Gambar 3.15**



Gambar 3.16 Diagram Alir Tahap *Join*/Agregasi

3.2.6.2 Proses Uji Coba *Insert*

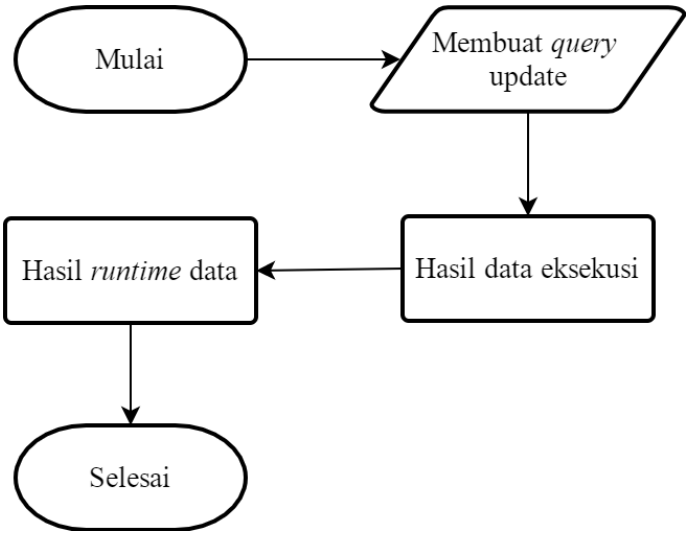
Proses ini dilakukan untuk membandingkan kemampuan *insert* data dari database SQL dengan database NoSQL. Tahap yang dilakukan antara lain memasukkan *dataset* pada GUI, menampilkan jumlah data hasil transaksi, dan menampilkan *runtime* data. Diagram alir dari proses ini ditunjukkan pada **Gambar 3.16**



Gambar 3.17 Diagram Alir Tahap *Insert*

3.2.6.3 Proses Uji Coba *Update*

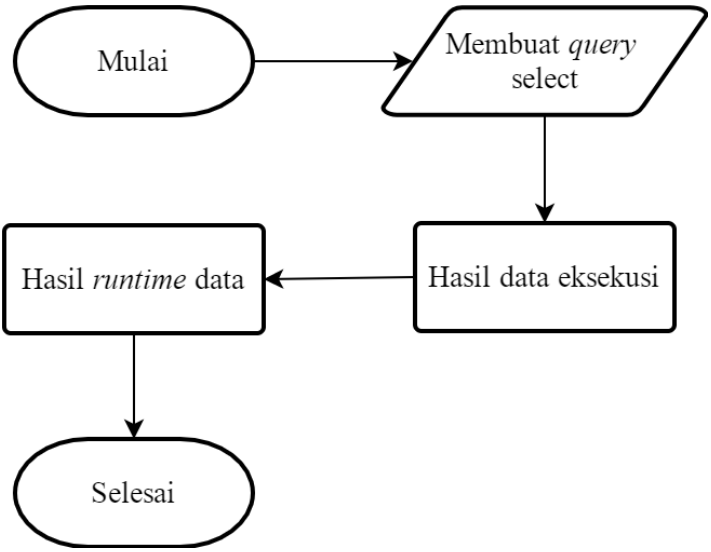
Proses ini dilakukan untuk membandingkan kemampuan *insert* data dari database SQL dengan database NoSQL. Tahap yang dilakukan antara lain membuat *query update*, menampilkan hasil data transaksi, dan menampilkan *runtime* data. Diagram alir dari proses ini ditunjukkan pada **Gambar 3.17**



Gambar 3.18 Diagram Alir Tahap *Update*

3.2.6.4 Proses Uji Coba *Select*

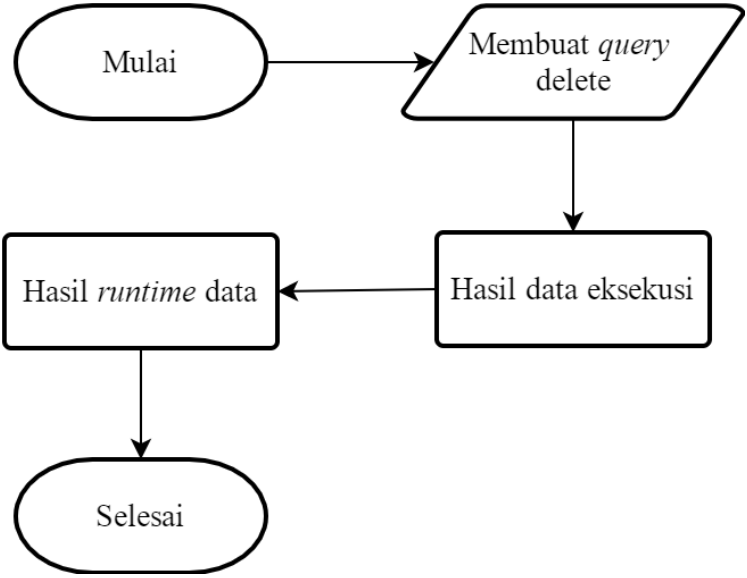
Proses ini dilakukan untuk membandingkan kemampuan *insert* data dari database SQL dengan database NoSQL. Tahap yang dilakukan antara lain membuat *query select*, menampilkan hasil data transaksi, dan menampilkan *runtime* data. Diagram alir dari proses ini ditunjukkan pada **Gambar 3.18**



Gambar 3.19 Diagram Alir Tahap *Select*

3.2.6.5 Proses Uji Coba *Delete*

Proses ini dilakukan untuk membandingkan kemampuan *insert* data dari database SQL dengan database NoSQL. Tahap yang dilakukan antara lain membuat *query delete*, menampilkan hasil data transaksi, dan menampilkan *runtime* data. Diagram alir dari proses ini ditunjukkan pada **Gambar 3.19**



Gambar 3.20 Diagram Alir Tahap *Delete*

3.2.6.6 Proses Uji Coba *Flexibility*

Pada tahap ini akan dijelaskan proses-proses yang dilakukan untuk uji coba *flexibility* pada database MongoDB. Pada proses ini akan membuktikan bahwa *database* jenis NoSQL bersifat *schemaless* yang mana dalam satu tabel/*collection* mampu menampung data yang berbeda-beda strukturnya.

Adapun tahap uji coba dilakukan dengan cara insert data dengan struktur yang berbeda-beda. Adapun desain uji coba sebagai berikut:

- Insert 1

  {

  data A;
  data B;
  data C;
  data D;

}

- Insert 2
  {
  data A;
  data B;
  data D;
  }

- Insert 3
  {
  data A;
  data B;
  data C;
  data D;
  data E;

  }

BAB IV
IMPLEMENTASI SISTEM

Bab ini membahas implementasi dari perancangan sistem ERP Retail yang di dalamnya akan dibahas lingkungan pengembangan system dan implementasi dari perancangan *backend* web, antarmuka pengguna, *multitenancy*, dan pendistribusian data.

4.1 Lingkungan Pengembangan Sistem

Lingkungan pengembangan sistem yang digunakan untuk mengembangkan Tugas Akhir ini dilakukan pada lingkungan dan kakas sebagai berikut.

1. Basis data yang digunakan pada *server* adalah MongoDB.
2. PC untuk *server database* menggunakan Sistem Operasi Windows 10 Pro x64 dan Ubuntu 16.04.
3. PC untuk *server* menggunakan Intel® Core™ i5-3230 @2.60GHz , RAM 4GB dengan Sistem Operasi Windows 10 Pro x64 dan Ubuntu 16.04
4. Mozilla Firefox 51.0.1 dan Chrome 58.0 sebagai antarmuka untuk pengujian aplikasi klien
5. Sublime 1.0.0.1 sebagai text editor untuk membangun aplikasi

4.2 Implementasi *Backend* Web

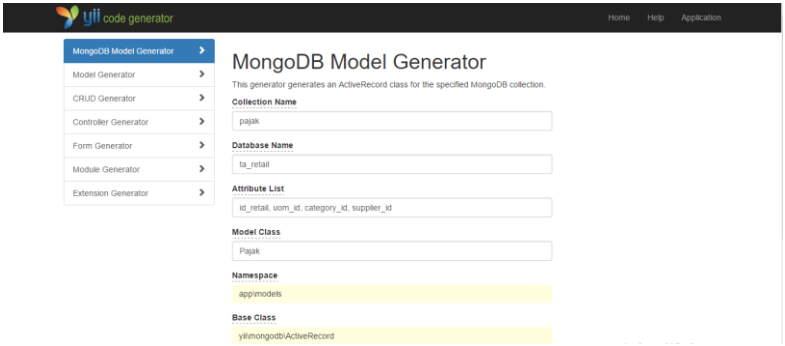
Pada bagian ini akan dijelaskan secara detail mengenai implementasi *backend* web pada aplikasi ERP Retail dengan menggunakan *framework* yii2.

4.2.1 *Generate* Kode Menggunakan Gii Generator

Pada tahap implementasi ini akan dijabarkan bagaimana proses-proses menggunakan gii generator untuk menghasilkan *ActiveRecord class* dalam bentuk file *model*, *controller*, dan *view* sebagai kode *basic* CRUD yii2 MongoDB[11].

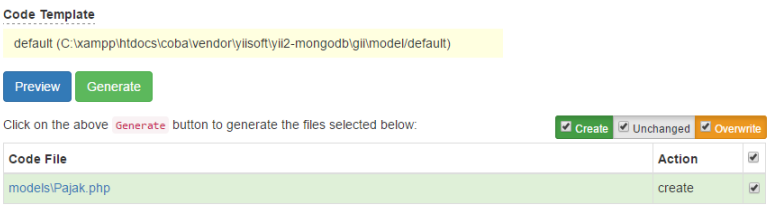
4.2.1.1 *Generate* MongoDB Kode *Model*

Langkah awal menggunakan gii generator dimulai dengan *generate* kode *model* dengan cara mengisikan nama *collection*, nama *database*, list atribut, dan *model class.*

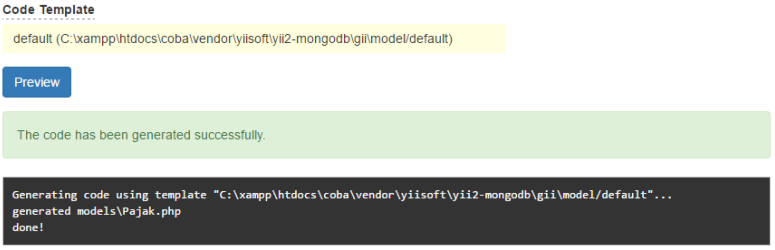


Gambar 4.1 Proses generate kode *model*

Setelah form terisi maka klik tombol preview, maka akan menampilkan hasil kode kemudian klik tombol generate untuk mendapatkan kode.



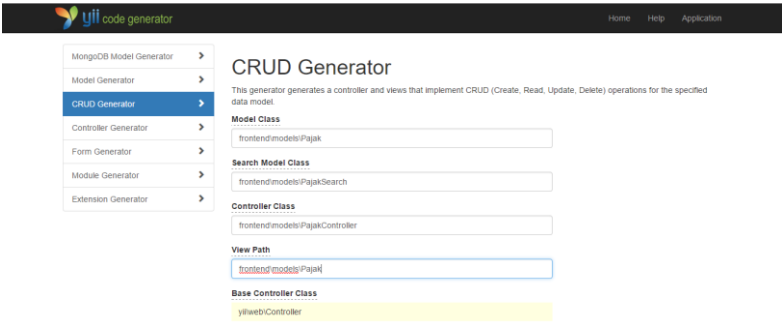
Gambar 4.2 Hasil preview kode



Gambar 4.3 Hasil generate kode

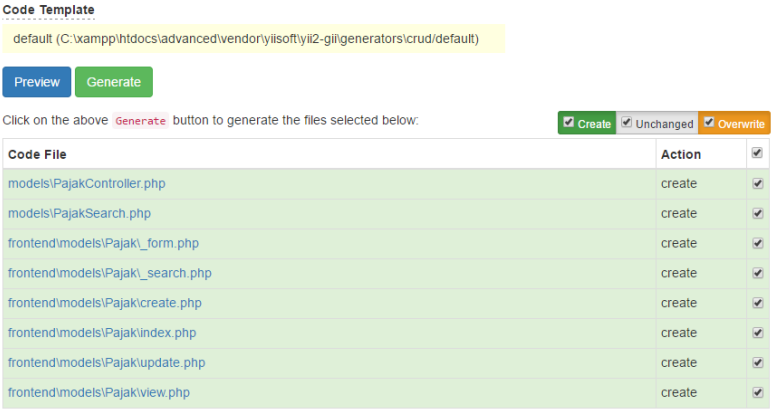
4.2.1.2 *Generate* MongoDB CRUD Kode

Pada tahap ini akan di generate kode untuk CRUD pada aplikasi berdasarkan kode *model* yang di *generate* dari tahap sebelumnya. Untuk melakukan *generate* CRUD kode harus mengisi form *model class*, *search model class*, *controller class*, dan *view path.*



Gambar 4.4 Proses generate CRUD kode

Setelah form terisi maka klik tombol preview, maka akan menampilkan hasil kode kemudian klik tombol generate untuk mendapatkan kode.



Gambar 4.5 Hasil preview kode

Gambar 4.6 Hasil generate kode

4.2.1.3 Hasil MongoDB CRUD dan *Model* Kode

Setelah melakukan *generate* kode untuk aplikasi CRUD maka akan dihasilkan beberapa file .php yang akan dijabarkan sebagai berikut:

- Category.php

```
class Category extends \yii\mongodb\ActiveRecord
{
    public static function collectionName()
    {
        return ['ta_retail', 'category'];
    }

    public function attributes()
    {
        return [
            '_id',
            'category_name',
            'category_des',
            'id_retail',
        ];
    }

    public function rules()
    {
        return [
            [['category_name', 'category_des', 'id_retail'], 'safe']
        ];
    }

    public function attributeLabels()
    {
        return [
            '_id' => 'ID',
            'category_name' => 'Category Name',
            'category_des' => 'Category Des',
            'id_retail' => 'Id Retail',
        ];
    }
}
```

Kode Sumber 4.1 Model

- CategoryController.php

```
class CategoryController extends Controller
{
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    public function actionIndex()
    {
        $searchModel = new CategorySearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }
```

```
public function actionCreate()
    {
        $model = new Pajak();

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => (string)$model->_id]);
        } else {
            return $this->render('create', [
                'model' => $model,
            ]);
        }
    }

    public function actionUpdate($id)
    {
        $model = $this->findModel($id);

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => (string)$model->_id]);
        } else {
            return $this->render('update', [
                'model' => $model,
            ]);
        }
    }

    public function actionDelete($id)
    {
        $this->findModel($id)->delete();
```

```
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

     protected function findModel($id)
    {
        if (($model = Category::findOne($id)) !== null) {
            return $model;
        } else {
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
}
```

Kode Sumber 4.2 *Controller*

- CategorySearch.php

```
class CategorySearch extends Category
{
    public function rules()
    {
        return [
            [['_id', 'category_name', 'category_des', 'id_retail'], 'safe'],
        ];
    }

    public function scenarios()
    {
        return Model::scenarios();
```

```
    }

    public function search($params)
    {
        $query = Category::find();

        // add conditions that should always apply here

        $dataProvider = new ActiveDataProvider([
            'query' => $query,
        ]);

        $this->load($params);

        if (!$this->validate()) {
            // uncomment the following line if you do not want to
return any records when validation fails
            // $query->where('0=1');
            return $dataProvider;
        }

        // grid filtering conditions
        $query->andFilterWhere(['like', '_id', $this->_id])
            ->andFilterWhere(['like',     'category_name',     $this-
>category_name])
            ->andFilterWhere(['like',      'category_des',      $this-
>category_des])
            ->andFilterWhere(['like',  'id_retail',  Yii::$app->user-
>identity->id]);

        return $dataProvider;
    }
}
```

Kode Sumber 4.3 *Controller Search*

- _form.php

```
<div class="row">
        <div class="col-lg-6">
                <div class="box box-primary">
                        <div class="box-body">
                                <div class="factory-form">

                                <?php $form = ActiveForm::begin(); ?>

                                <?= $form->field($model, 'category_name') ?>

                                <?= $form->field($model, 'category_des') ?>

                                <?= $form->field($model, 'id_retail')->hiddenInput(['value'=> Yii::$app->user->identity->id])->label(false) ?>

                                <div class="form-group">
                                   <?= Html::submitButton($model->isNewRecord ? 'Create' : 'Update', ['class' => $model->isNewRecord ? 'btn btn-success' : 'btn btn-primary']) ?>
                                </div>

                                <?php ActiveForm::end(); ?>

            </div>
          </div>
       </div>
</div>
```

Kode Sumber 4.4 *Form*

- _search.php

```
<div class="category-search">

    <?php $form = ActiveForm::begin([
        'action' => ['index'],
        'method' => 'get',
    ]); ?>

    <?= $form->field($model, '_id') ?>

    <?= $form->field($model, 'category_name') ?>

    <?= $form->field($model, 'category_des') ?>

    <?= $form->field($model, 'id_retail') ?>

    <div class="form-group">
        <?= Html::submitButton('Search', ['class' => 'btn btn-primary']) ?>
        <?= Html::resetButton('Reset', ['class' => 'btn btn-default']) ?>
    </div>

    <?php ActiveForm::end(); ?>

</div>
```

Kode Sumber 4.5 *Search*

- create.php

```
$this->title = 'Create Category';
$this->params['breadcrumbs'][] = ['label' => 'Categories', 'url'
=> ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="category-create">

    <h1><?= Html::encode($this->title) ?></h1>

    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>

</div>
```

Kode Sumber 4.6 *Create*

- index.php

```
$this->title = 'Categories';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="category-index">

    <h1><?= Html::encode($this->title) ?></h1>
    <?php // echo $this->render('_search', ['model' =>
$searchModel]); ?>

    <p>
        <?= Html::a('Create Category', ['create'], ['class' => 'btn
btn-success']) ?>
    </p>
    <?= DataTables::widget([
        'dataProvider' => $dataProvider,
```

```
        'filterModel' => $searchModel,
        'columns' => [
            ['class' => 'yii\grid\SerialColumn'],

            '_id',
            'category_name',
            'category_des',
            //'id_retail',

            ['class' => 'yii\grid\ActionColumn'],
        ],
    ]); ?>
</div>
```

Kode Sumber 4.7 Index

- update.php

```
$this->title = 'Update Category: ' . $model->_id;
$this->params['breadcrumbs'][] = ['label' => 'Categories', 'url'
=> ['index']];
$this->params['breadcrumbs'][] = ['label' => $model->_id,
'url' => ['view', 'id' => (string)$model->_id]];
$this->params['breadcrumbs'][] = 'Update';
?>
<div class="category-update">

    <h1><?= Html::encode($this->title) ?></h1>

    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
</div>
```

Kode Sumber 4.8 *Update*

- view.php

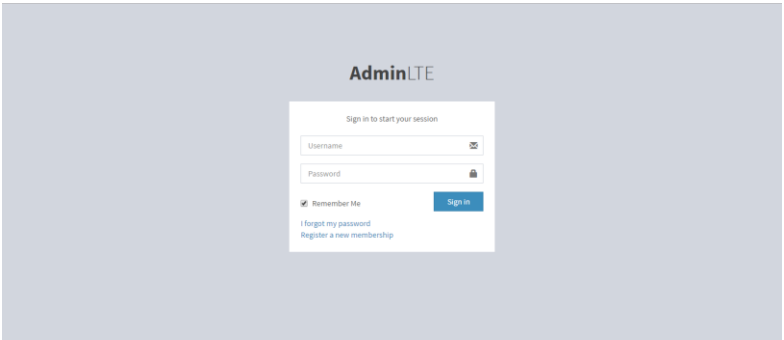
```
$this->title = $model->_id;
$this->params['breadcrumbs'][] = ['label' => 'Categories', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="category-view">
    <h1><?= Html::encode($this->title) ?></h1>
    <p>
        <?= Html::a('Update', ['update', 'id' => (string)$model->_id], ['class' => 'btn btn-primary']) ?>
        <?= Html::a('Delete', ['delete', 'id' => (string)$model->_id], [
            'class' => 'btn btn-danger',
            'data' => [
                'confirm' => 'Are you sure you want to delete this item?',
                'method' => 'post',
            ],
        ]) ?>
    </p>

    <?= DetailView::widget([
        'model' => $model,
        'attributes' => [
            '_id',
            'category_name',
            'category_des',
            'id_retail',
        ],
    ]) ?>
</div>
```

Kode Sumber 4.9 *View*

4.3 Implementasi Antarmuka Pengguna

Implementasi antarmuka pengguna berbasis web ini menggunakan bahasa HTML dan *framework* yii2 dengan template *interface* AdminLTE. Pada subbab akan dijelaskan dan ditampilkan tampilan halaman ERP Retail sesuai rancangan antarmuka yang terdapa pada bab III.

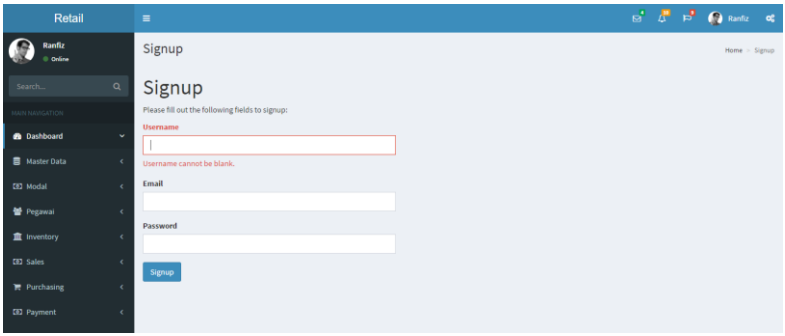
4.3.1 Implementasi Halaman Antarmuka Login



Gambar 4.7 Antarmuka Halaman Login

Pada antarmuka ini pengguna dapat melihat form berisi *username* dan *password* serta tombol *sign in* untuk login ke aplikasi. Di bawah form login juga terdapat fitur untuk registrasi pengguna baru

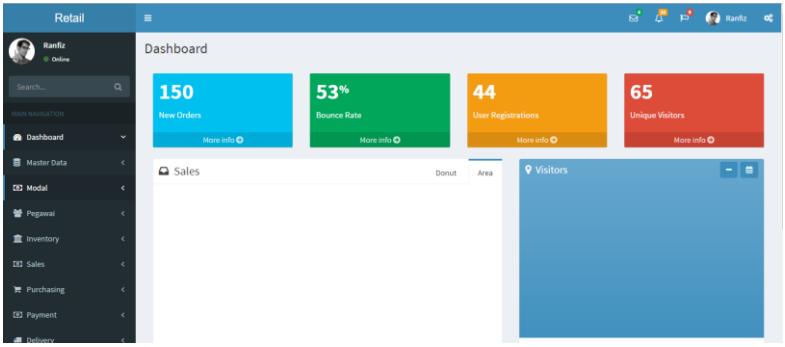
4.3.2 Implementasi Halaman Antarmuka Menambahkan User Baru



Gambar 4.8 Antarmuka Halaman Menambahkan User baru

Pada antarmuka ini pengguna dapat melihat dari atas berisi judul aplikasi, notifikasi, dan fitu untuk *sign out*. Lalu di sebelah kiri terdapat menu navigasi yang berisi status *user* dan modul-modul dari ERP Retail. pada halaman tengah terdapat form yang berisi *username*, *email*, dan *password* serta tombol *sign up* untuk menambahkan *user* ke aplikasi.

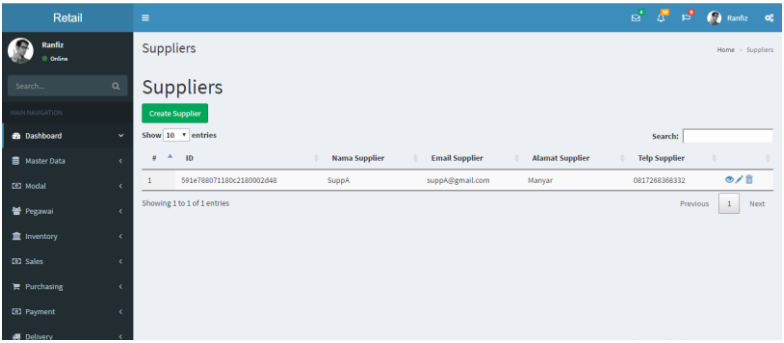
4.3.3 Implementasi Halaman Antarmuka Dashboard



Gambar 4.9 Antarmuka Halaman Dashboard

Pada antarmuka ini berisi halam utama dari aplikasi ERP Retail. Pengguna dapat melihat dari atas berisi judul aplikasi, notifikasi, dan fitu untuk *sign out*. Lalu di sebelah kiri terdapat menu navigasi yang berisi status *user* dan modul-modul dari ERP Retail. pada halaman tengah terdapat beberapa grafik atau data-data dari *business intelegent* ERP Retail.

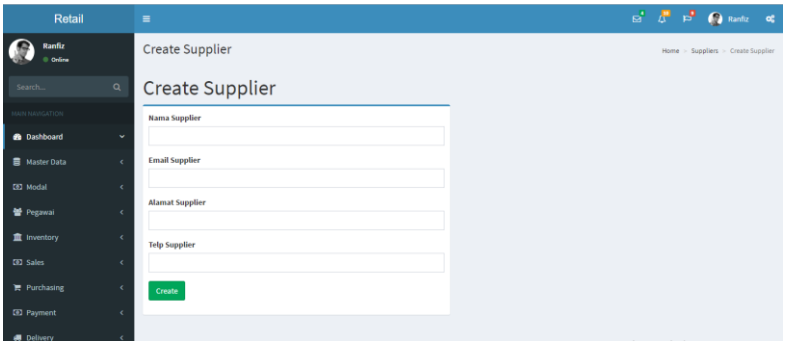
4.3.4 Implementasi Halaman Antarmuka Submodul



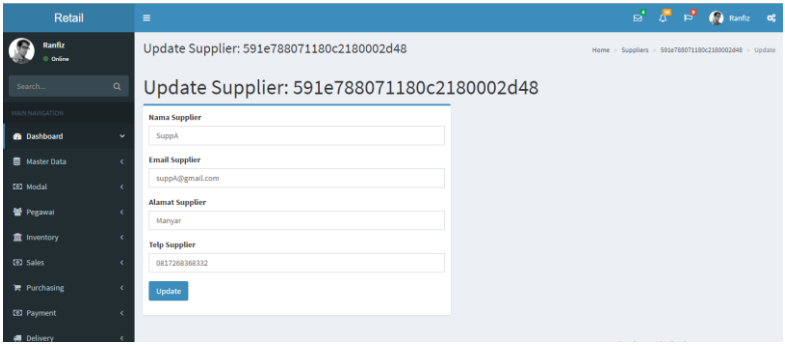
Gambar 4.10 Antarmuka Halaman Submodul

Pada antarmuka ini pengguna dapat melihat dari atas berisi judul aplikasi, notifikasi, dan fitu untuk *sign out*. Lalu di sebelah kiri terdapat menu navigasi yang berisi status *user* dan modul-modul dari ERP Retail. pada halaman tengah terdapat judul modul dan dibawahnya terdapat tombol untuk create data modul. Juga hamalan ini menampilkan data modul berdasarkan id *tenant* yang melakukan transaksi.

4.3.5 Implementasi Halaman Antarmuka *Create* dan *Update*



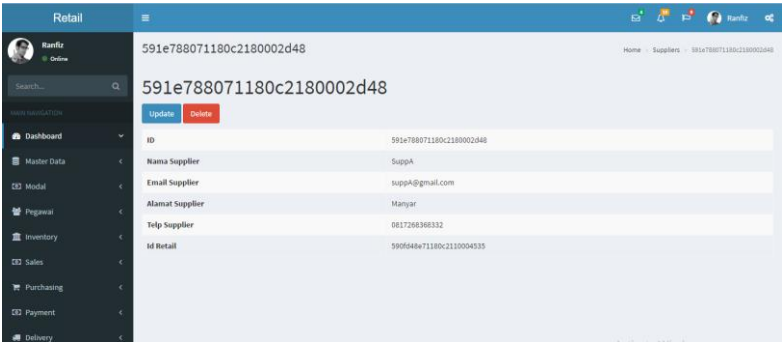
Gambar 4.11 Antarmuka Halaman *Create*



Gambar 4.12 Antarmuka Halaman *Update*

Pada antarmuka ini pengguna dapat melihat dari atas berisi judul aplikasi, notifikasi, dan fitu untuk *sign out*. Lalu di sebelah kiri terdapat menu navigasi yang berisi status *user* dan modul-modul dari ERP Retail. Pada halaman tengah terdapat judul transaksi apa yang akan dilakukan (*create* atau *update*) lalu dibawahnya berisi form kosong jika digunakan untuk transaksi *create* atau form yang berisi data untuk transaksi *update*. Lalu di bawah form terdapat tombol *create* atau *update* sesuai transaksi yang akan dilakukan.

4.3.6 Implementasi Halaman Antarmuka *View*



Gambar 4.13 Antarmuka Halaman *View*

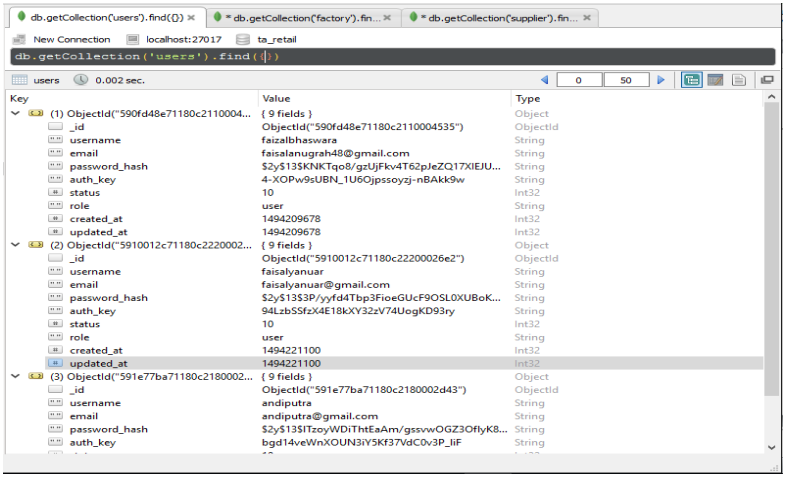
Pada antarmuka ini pengguna dapat melihat dari atas berisi judul aplikasi, notifikasi, dan fitu untuk *sign out*. Lalu di sebelah kiri terdapat menu navigasi yang berisi status *user* dan modul-modul dari ERP Retail. Pada halaman tengah terdapat id data. Kemudian diikuti dua tombol untuk *update* dan *delete* dan dibawah tombol berisi informasi data yang di *view*.

4.4 Implementasi *Multitenancy*

Pada bagian ini akan dijelaskan mengenai implementasi *multitenancy* pada aplikasi ERP Retail. Implementasi *multitenancy* pada tugas akhir ini diaplikasikan pada level aplikasi. Mengingat *database* MongoDB tidak support *multitenant* level *database engine*. Proses implementasi *multitenancy* secara rinci dijabarkan sebagai berikut:

4.4.1 Registrasi Setiap *Tenant*

Pada proses ini *tenant* baru melakukan registrasi dengan tujuan untuk bisa *login* pada aplikasi ERP Retail. Setiap tenant akan memilik _id *unique* yang memang di *generate* langsung oleh *database* MongoDB. Data semua *tenant* disimpan dalam satu *collection*.



Gambar 4.14 Data *Tenant*

4.4.2 Menempelkan Id *Tenant* Setiap *Store* Data

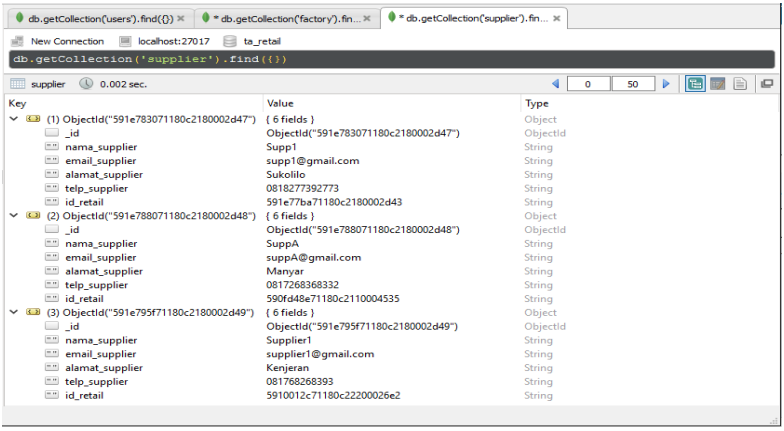
Pada proses ini aplikasi dirancang agar setiap data transaksi yang dilakukan *tenant* ikut ditempel id *tenant* melalui *session* saat *tenant* login.

```
<?php $form = ActiveForm::begin(); ?>
<?= $form->field($model, 'nama_supplier') ?>
 <?= $form->field($model, 'email_supplier') ?>
<?= $form->field($model, 'alamat_supplier') ?>
<?= $form->field($model, 'telp_supplier') ?>
<?=$form->field($model, 'id_retail')->hiddenInput(['value'=>
Yii::$app->user->identity->id])->label(false)   ?>
```

Kode Sumber 4.10 Menempel Id *Tenant*

Dengan cara seperti ini maka setiap *tenant* saat melakukan transaksi data setelah login, maka data transaksi akan ditempel id *tenant* yang bersangkutan. Id ini berfungsi untuk

melakukan *filter* data pada level aplikasi agar data yang muncul hanya dari data milik *tenant* yang melakukan login.



Gambar 4.15 Data Tansaksi Ditempel Id *Tenant*

4.4.3 *Filter* Data Berdasarkan Id *Tenant*

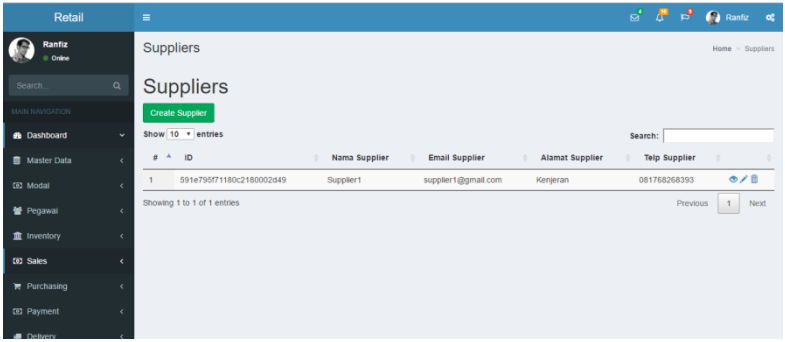
Pada proses ini aplikasi dirancang untuk melakukan *filter* data dari database berdasarkan id *tenant* yang sedang login.

```
$query
->andFilterWhere(['like', '_id', $this->_id])
->andFilterWhere(['like','nama_supplier',$this-
>nama_supplier])
->andFilterWhere(['like','email_supplier',$this-
>email_supplier])
->andFilterWhere(['like','alamat_supplier',$this-
>alamat_supplier])
->andFilterWhere(['like','telp_supplier',$this->telp_supplier])
->andFilterWhere(['like','id_retail',Yii::$app->user->identity-
>id]);
```

Kode Sumber 4.11 *Filtering* Berdasarkan Id *Tenant*

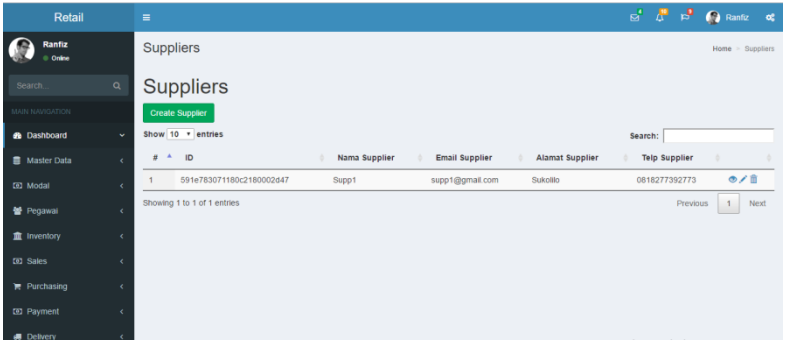
Dengan cara seperti ini maka aplikasi hanya menampilkan data berdasarkan id *tenant* yang login. Dibawah ini akan dijabarkan hasil data dari *filtering* pada level aplikasi berdasarkan id *tenant* yang login.

- Tenant A



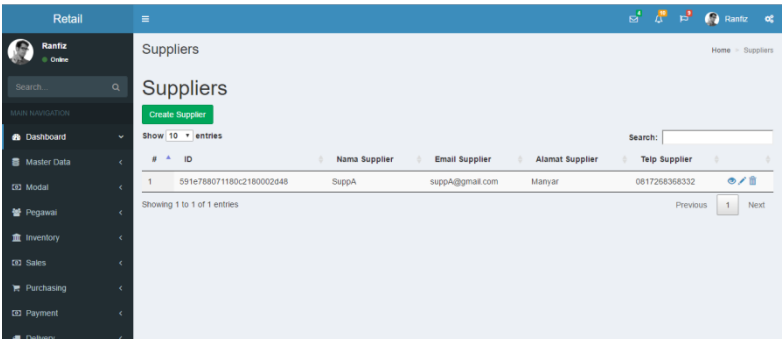
Gambar 4.16 Data *Tenant* A dari Hasil *Filtering*

- Tenant B



Gambar 4.17 Data *Tenant* B dari Hasil *Filtering*

- Tenant C

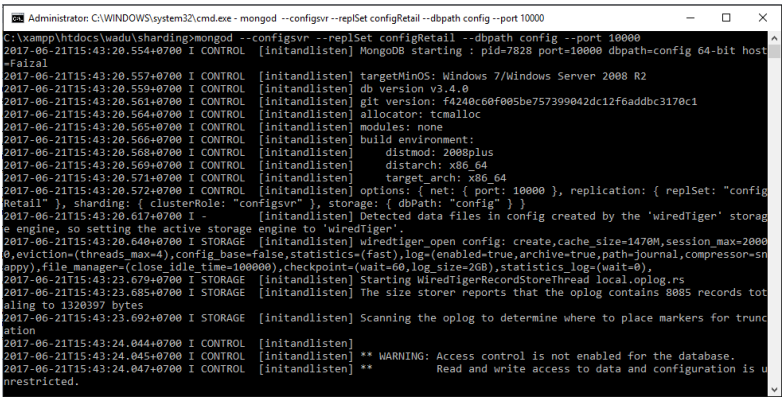


Gambar 4.18 Data *Tenant* C dari Hasil *Filtering*

4.5 Implementasi Penditribusian Data

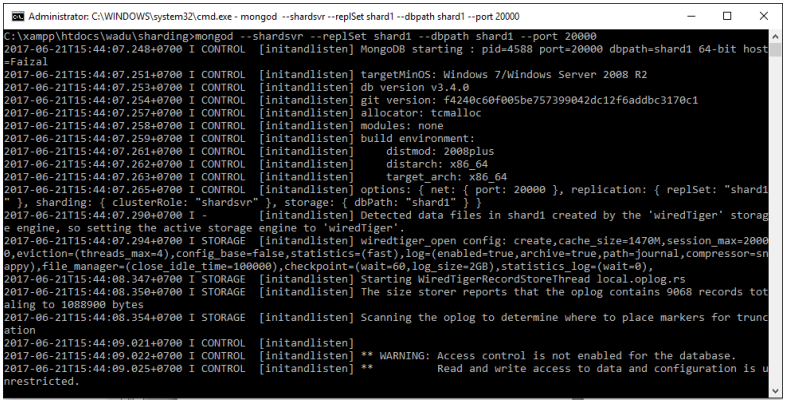
Dengan Implementasi pendistribusian data pada tugas akhir ini akan menggunakan metode *sharding* milik MongoDB. Dalam uji coba nya akan menggunakan server local, sehingga inisialilasi antar server menggunakan *folder* sebagai penyimpanan data yang terdistribusi[12]. Adapun penjabaran proses *sharding* akan dijelaskan dibawah ini:

1. Konfigurasi *server* sebagai *config* untuk sharding. Dalam uji coba ini digunakan folder “sharding” sebagai lokasi pendistribusian data. Pembagian *server* untuk *config* dan *shard* dibedakan pada inisialisasi port. Untuk *config* digunakan port 10000. Hasil proses inisialisasi *config* ditunjukkan pada **Gambar 4.19**

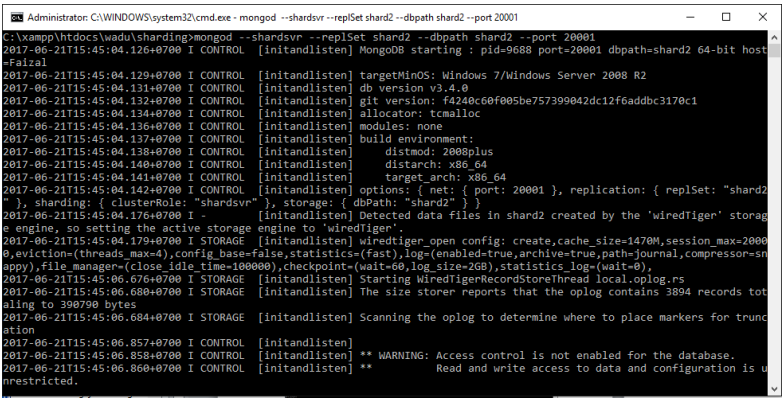


Gambar 4.19 Konfigurasi *Config Server* pada Port 10000

2. Konfigurasi *shard* sebagai tempat menampung data yang terdistribusi. Pada uji coba tugas akhir ini digunakan dua *shard server* yaitu pada port 20000 dan 20001. Hasil proses inisialisasi *shard* ditunjukkan pada **Gambar 4.20 dan 4.21**

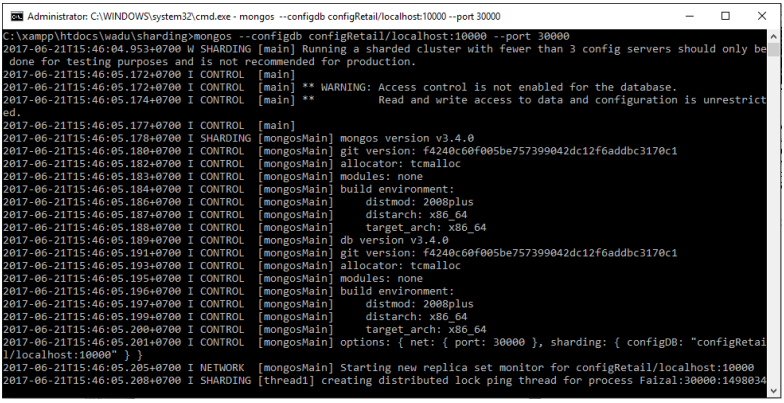


Gambar 4.20 Konfigurasi *Shard Server* pada Port 20000

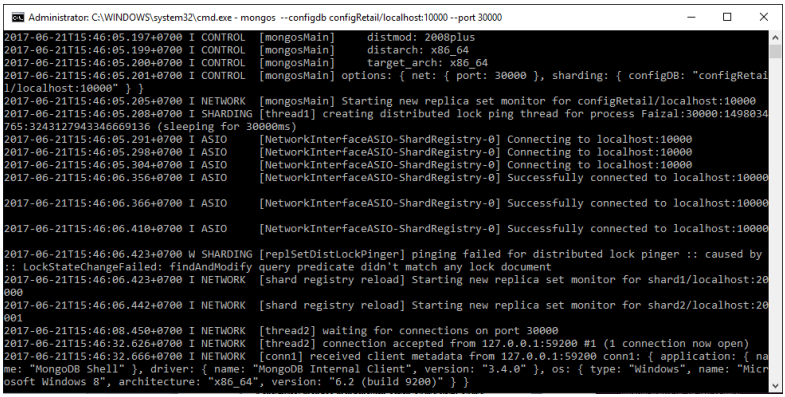


Gambar 4.21 Konfigurasi *Shard Server* pada Port 20001

3. Konfigurasi mongos (MongoDB *server*) untuk mengenalkan database dengan lokasi *sharding* yaitu pada *config server* di port 10000. Mongos diinisialisasikan pada port 30000. Setelah melakukan proses maka akan menjukkan hasil bahwa sudah connect dengan *config server* pada port 10000. Hasil proses inisialisasi mongos dan hasil *connect* ditunjukkan pada **Gambar 4.22** dan **Gambar 4.23**

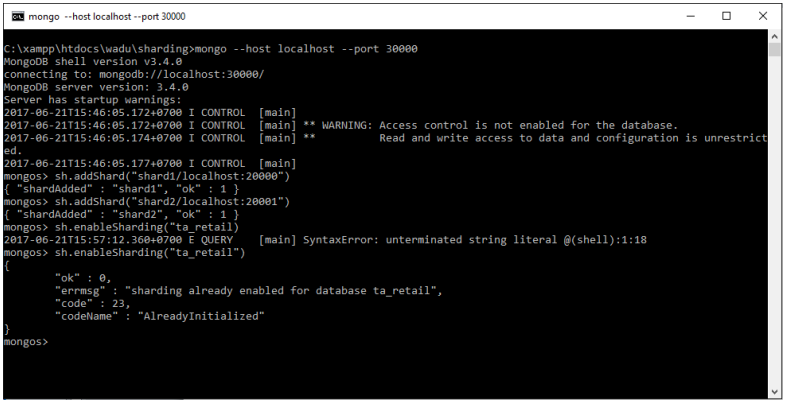


Gambar 4.22 Konfigurasi Mongos pada Database



Gambar 4.23 Hasil Mongos *Connect Config Server*

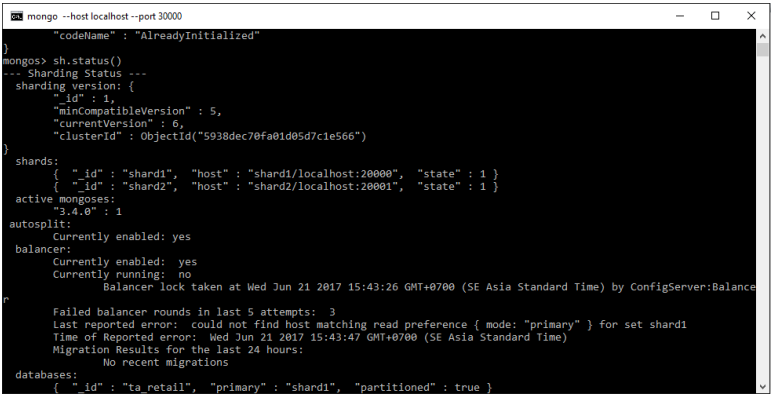
4. Konfigurasi mongos mengenalkan setiap *shard* yaitu *shard* pada port 20000 dan 20001. Lalu diikuti penngaktifan sharding dan pengecekan status sharding. Hasil proses inisialisasi mongos ditunjukkan pada **Gambar 4.24**



Gambar 4.24 Konfigurasi Mongos pada *Shard*

5. Hasil status *sharding* menunjukkan setiap *shard* yang terkoneksi, status autosplit, status load balancer, dan nama

dari *database* yang digunakan. Hasil status ditunjukkan pada **Gambar 4.25**



Gambar 4.25 Status *Sharding*

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan analisa sistem yang telah dijabarkan pada Bab III dan terhadap tujuan dibuatnya aplikasi ini, yakni agar menerapkan database yang *good performance*, flexible, dan *scalable*.

5.1 Lingkungan Pengujian

Lingkungan pengembangan sistem yang digunakan untuk mengembangkan Tugas Akhir ini dilakukan pada lingkungan dan kakas sebagai berikut.

1. Basis data yang digunakan pada *server* adalah MongoDB dan MySQL.
2. PC untuk *server database* menggunakan Sistem Operasi Windows 10 Pro x64 dan Ubuntu 16.04.
3. PC untuk *server* menggunakan Intel® Core™ i5-3230 @2.60GHz , RAM 4GB dengan Sistem Operasi Windows 10 Pro x64 dan Ubuntu 16.04.
4. Robomongo dan phpMyAdmin sebagai GUI *database.*
5. *Dataset* format JSON dan SQL.

5.2 Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pengujian dibagi menjadi dua yaitu pengujian *performa*nce dan *flexibility*. Pengujian *performance* dilakukan dengan cara memasukkan *dataset* untuk uji coba testing *insert* dan mengelola data dalam *database* dengan *query* untuk uji coba *join*/agregasi, *select*, *update*, dan *delete*. Untuk pengujian flexibility dilakukan dengan cara memasukkan beberapa data berbeda yang membentuk skema ke dalam database.

5.3 Pengujian Fungsionalitas Sistem

Pengujian fungsionalitas aplikasi dilakukan dengan melakukan skenario yang sama dengan rancangan alur proses aplikasi sebagai tolok ukur keberhasilan pengujian, dan mengacu pada kasus penggunaan yang sebelumnya telah dijelaskan pada Bab III. Pengujian pada kebutuhan fungsionalitas dapat dijabarkan pada subbab berikut:

5.3.1.1 Pengujian *Manage Category*

Pengujian ini terdiri dari pengujian *manage category*. Pengujian *manage category* yaitu menambah, menyunting, dan menghapus data *category* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.1**.

Tabel 5.1 Pengujian *Manage Category*

Nomor	**UC-001**
Nama	*Manage Category*
Tujuan Pengujian	Menguji kemampuan sistem dalam *manage* data *category*
Skenario 1	**Pengguna menambah data *category***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Category* di dalam menu *Master Data*
Data Uji	Data *category*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *category*
Skenario 2	**Pengguna menyunting data *category***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Category* di dalam menu

	Master Data
Data Uji	Data *category*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *category* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *category***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Category* di dalam menu *Master Data*
Data Uji	Data *category*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *category*

5.3.1.2 Pengujian *Manage Product*

Pengujian ini terdiri dari pengujian *manage product*. Pengujian *manage product* yaitu menambah, menyunting, dan menghapus data *product* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.2**.

Tabel 5.2 Pengujian *Manage Product*

Nomor	**UC-002**
Nama	*Manage Product*
Tujuan Pengujian	Menguji kemampuan sistem dalam *manage* data *product*
Skenario 1	**Pengguna menambah data *product***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Product* di dalam menu *Master Data*
Data Uji	Data *product*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *product*
Skenario 2	**Pengguna menyunting data *product***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Product* di dalam menu *Master Data*
Data Uji	Data *product*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *product* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *product***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Product* di dalam menu *Master Data*

Data Uji	Data *product*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *product*

5.3.1.3 Pengujian *Manage Uom*

Pengujian ini terdiri dari pengujian *manage uom*. Pengujian *manage uom* yaitu menambah, menyunting, dan menghapus data *uom* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.3**.

Tabel 5.3 Pengujian *Manage Uom*

Nomor	**UC-003**
Nama	*Manage Uom*
Tujuan Pengujian	Menguji kemampuan sistem dalam *manage* data *uom*
Skenario 1	**Pengguna menambah data *uom***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Uom* di dalam menu *Master Data*
Data Uji	Data *uom*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *uom*

Skenario 2	**Pengguna menyunting data *uom***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Uom* di dalam menu *Master Data*
Data Uji	Data *uom*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *uom* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *uom***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Uom* di dalam menu *Master Data*
Data Uji	Data *uom*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *uom*

5.3.1.4 Pengujian *Manage Supplier*

Pengujian ini terdiri dari pengujian *manage supplier*. Pengujian *manage supplier* yaitu menambah, menyunting, dan menghapus data *supplier* yang sudah ada pada sistem. Rincian

91skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.4**.

Tabel 5.4 Pengujian *Manage Supplier*

Nomor	**UC-004**
Nama	*Manage Supplier*
Tujuan Pengujian	Menguji kemampuan 91sistem dalam *manage* data *supplier*
Skenario 1	**Pengguna menambah data *supplier***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Supplier* di dalam menu *Master Data*
Data Uji	Data *supplier*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *supplier*
Skenario 2	**Pengguna menyunting data *supplier***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Supplier* di dalam menu *Master Data*
Data Uji	Data *supplier*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *supplier* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *supplier***

Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *supplier* di dalam menu *Master Data*
Data Uji	Data *supplier*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *supplier*

5.3.1.5 Pengujian *Manage Customer*

Pengujian ini terdiri dari pengujian *manage customer*. Pengujian *manage customer* yaitu menambah, menyunting, dan menghapus data *customer* yang sudah ada pada sistem. Rincian 92skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.5**.

Tabel 5.5 Pengujian *Manage Customer*

Nomor	**UC-005**
Nama	*Manage Customer*
Tujuan Pengujian	Menguji kemampuan 92sistem dalam *manage* data *customer*
Skenario 1	**Pengguna menambah data *customer***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Customer* di dalam menu *Master Data*
Data Uji	Data *customer*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data

Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *customer*
Skenario 2	**Pengguna menyunting data *customer***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *Customer* di dalam menu *Master Data*
Data Uji	Data *customer*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *customer* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *customer***
Kondisi Awal	Pengguna berada pada menu utama dan memilih submenu *customer* di dalam menu *Master Data*
Data Uji	Data *customer*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *customer*

5.3.1.6 Pengujian *Manage Delivery*

Pengujian ini terdiri dari pengujian *manage delivery*. Pengujian *manage delivery* yaitu menambah, menyunting, dan menghapus data *delivery* yang sudah ada pada sistem. Rincian 94skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.6**.

Tabel 5.6 Pengujian *Manage Delivery*

Nomor	**UC-006**
Nama	*Manage Delivery*
Tujuan Pengujian	Menguji kemampuan 94sistem dalam *manage* data *delivery*
Skenario 1	**Pengguna menambah data *delivery***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *delivery*
Data Uji	Data *delivery*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *delivery*
Skenario 2	**Pengguna menyunting data *delivery***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *delivery*
Data Uji	Data *delivery*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *delivery* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail

	update
Skenario 3	**Pengguna menghapus data *delivery***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *delivery*
Data Uji	Data *delivery*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *delivery*

5.3.1.7 Pengujian *Manage Modal*

Pengujian ini terdiri dari pengujian *manage modal*. Pengujian *manage modal* yaitu menambah, menyunting, dan menghapus data *modal* yang sudah ada pada sistem. Rincian 95skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.7**.

Tabel 5.7 Pengujian *Manage Modal*

Nomor	**UC-007**
Nama	*Manage Modal*
Tujuan Pengujian	Menguji kemampuan 95sistem dalam *manage* data *modal*
Skenario 1	**Pengguna menambah data *modal***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *modal*
Data Uji	Data *modal*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data

Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *modal*
Skenario 2	**Pengguna menyunting data *modal***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *modal*
Data Uji	Data *modal*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *modal* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *modal***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *modal*
Data Uji	Data *modal*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *modal*

5.3.1.8 Pengujian *Manage Staff*

Pengujian ini terdiri dari pengujian *manage staff*. Pengujian *manage staff* yaitu menambah, menyunting, dan

menghapus data *staff* yang sudah ada pada sistem. Rincian 97skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.8**.

Tabel 5.8 Pengujian *Manage Staff*

Nomor	**UC-008**
Nama	*Manage Staff*
Tujuan Pengujian	Menguji kemampuan 97sistem dalam *manage* data *staff*
Skenario 1	**Pengguna menambah data *staff***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *staff*
Data Uji	Data *staff*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *staff*
Skenario 2	**Pengguna menyunting data *staff***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *staff*
Data Uji	Data *staff*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *staff* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *staff***
Kondisi Awal	Pengguna berada pada menu utama dan

	memilih menu *staff*
Data Uji	Data *staff*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *staff*

5.3.1.9 Pengujian *Manage Inventory*

Pengujian ini terdiri dari pengujian *manage inventory*. Pengujian *manage inventory* yaitu menambah, menyunting, dan menghapus data *inventory* yang sudah ada pada sistem. Rincian 98skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.9**.

Tabel 5.9 Pengujian *Manage Inventory*

Nomor	**UC-009**
Nama	*Manage Inventory*
Tujuan Pengujian	Menguji kemampuan 98sistem dalam *manage* data *inventory*
Skenario 1	**Pengguna menambah data *inventory***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *inventory*
Data Uji	Data *inventory*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *inventory*

Skenario 2	**Pengguna menyunting data *inventory***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *inventory*
Data Uji	Data *inventory*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *inventory* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *inventory***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *inventory*
Data Uji	Data *inventory*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *inventory*

5.3.1.10 Pengujian *Manage Sales*

Pengujian ini terdiri dari pengujian *manage sales*. Pengujian *manage sales* yaitu menambah, menyunting, dan menghapus data *sales* yang sudah ada pada sistem. Rincian 99skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.10**.

Tabel 5.10 Pengujian *Manage Sales*

Nomor	**UC-010**
Nama	*Manage Sales*
Tujuan Pengujian	Menguji kemampuan 100sistem dalam *manage* data *sales*
Skenario 1	**Pengguna menambah data *sales***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *sales*
Data Uji	Data *sales*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *sales*
Skenario 2	**Pengguna menyunting data *sales***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *sales*
Data Uji	Data *sales*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *sales* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *sales***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *sales*
Data Uji	Data *sales*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah

	ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *sales*

5.3.1.11 Pengujian *Manage Purchasing*

Pengujian ini terdiri dari pengujian *manage purchasing*. Pengujian *manage purchasing* yaitu menambah, menyunting, dan menghapus data *purchasing* yang sudah ada pada sistem. Rincian 101skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.11**.

Tabel 5.11 Pengujian *Manage Purchasing*

Nomor	**UC-011**
Nama	*Manage Purchasing*
Tujuan Pengujian	Menguji kemampuan 101sistem dalam *manage* data *purchasing*
Skenario 1	**Pengguna menambah data *purchasing***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *purchasing*
Data Uji	Data *purchasing*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *purchasing*
Skenario 2	**Pengguna menyunting data *purchasing***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *purchasing*

Data Uji	Data *purchasing*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *purchasing* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *purchasing***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *purchasing*
Data Uji	Data *purchasing*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *purchasing*

5.3.1.12 Pengujian *Manage Payment*

Pengujian ini terdiri dari pengujian *manage payment*. Pengujian *manage payment* yaitu menambah, menyunting, dan menghapus data *payment* yang sudah ada pada sistem. Rincian 102skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.12**.

Tabel 5.12 Pengujian *Manage Payment*

Nomor	**UC-012**

Nama	*Manage Payment*
Tujuan Pengujian	Menguji kemampuan 103sistem dalam *manage* data *payment*
Skenario 1	**Pengguna menambah data *payment***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *payment*
Data Uji	Data *payment*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *payment*
Skenario 2	**Pengguna menyunting data *payment***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *payment*
Data Uji	Data *payment*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *payment* yang telah ada, kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *payment***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *payment*
Data Uji	Data *payment*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus

	dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *payment*

5.3.1.13 Pengujian *Manage Asset*

Pengujian ini terdiri dari pengujian *manage asset*. Pengujian *manage asset* yaitu menambah, menyunting, dan menghapus data *asset* yang sudah ada pada sistem. Rincian 104skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.13**.

Tabel 5.13 Pengujian *Manage Asset*

Nomor	**UC-013**
Nama	*Manage Asset*
Tujuan Pengujian	Menguji kemampuan 104sistem dalam *manage* data *asset*
Skenario 1	**Pengguna menambah data *asset***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *asset*
Data Uji	Data *asset*
Langkah Pengujian	Pengguna mengeklik tombol *create* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *asset*
Skenario 2	**Pengguna menyunting data *asset***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *asset*
Data Uji	Data *asset*
Langkah Pengujian	Pengguna memilih tombol *update* dari salah satu data *asset* yang telah ada,

	kemudian menyunting data tersebut dan menyimpan data yang baru ke dalam basis data
Hasil Yang Diharapkan	Data baru yang dimasukkan, menggantikan data lama yang ada di dalam basis data
Hasil Yang Didapat	Data baru tersimpan di dalam basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman detail *update*
Skenario 3	**Pengguna menghapus data *asset***
Kondisi Awal	Pengguna berada pada menu utama dan memilih menu *asset*
Data Uji	Data *asset*
Langkah Pengujian	Pengguna memilih tombol *delete* dari salah satu data struktur perusahaan yang telah ada, kemudian melakukan konfirmasi penghapusan data dan data akan dihapus dari basis data
Hasil Yang Diharapkan	Data yang dihapus akan terhapus dari basis data
Hasil Yang Didapat	Data terhapus di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama *asset*

5.3.1.14 Pengujian *Manage Register*

Pengujian ini terdiri dari pengujian *manage register*. Pengujian *manage register* yaitu menambah *tenant* baru oleh admin ke dalam aplikasi. Rincian 105skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.14**.

Tabel 5.14 Pengujian *Manage Register*

Nomor	**UC-014**
Nama	*Manage Register*
Tujuan Pengujian	Menguji kemampuan 105sistem dalam *manage register tenant* baru
Kondisi Awal	Pengguna berada pada menu *log in* dan

	mengeklik tombol *register*
Data Uji	Data *tenant*
Langkah Pengujian	Pengguna mengeklik tombol *resgister* dan memasukkan data uji ke dalam basis data
Hasil Yang Diharapkan	Data yang dimasukkan tersimpan di basis data
Hasil Yang Didapat	Data tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama aplikasi

5.3.1.15 Pengujian *Manage Log In*

Pengujian ini terdiri dari pengujian *manage log in*. Pengujian *manage log in* yaitu *tenant* masuk ke dalam aplikasi. Rincian 106skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.15**.

Tabel 5.15 Pengujian *Manage Log In*

Nomor	**UC-015**
Nama	*Manage Log In*
Tujuan Pengujian	Menguji kemampuan 106sistem dalam *manage log in tenant* untuk masuk ke aplikasi
Kondisi Awal	Pengguna berada pada menu *log in* dan memasukkan id dan *password*
Data Uji	Data *tenant*
Langkah Pengujian	Pengguna mengeklik tombol *log in*
Hasil Yang Diharapkan	Pengguna masuk ke halaman utama aplikasi
Hasil Yang Didapat	Pengguna masuk ke halaman utama aplikasi
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama aplikasi

5.3.1.16 Pengujian *Manage Log Out*

Pengujian ini terdiri dari pengujian *manage log out*. Pengujian *manage log out* yaitu *tenant* keluar dari aplikasi. Rincian 107skenario pengujian pada kasus penggunaan ini dapat dilihat pada **Tabel 5.16**.

Tabel 5.16 Pengujian *Manage Log In*

Nomor	**UC-016**
Nama	*Manage Log Out*
Tujuan Pengujian	Menguji kemampuan 107sistem dalam *manage log out tenant* untuk keluar dari aplikasi
Kondisi Awal	Pengguna berada pada halama utama aplikasi
Data Uji	Data *tenant*
Langkah Pengujian	Pengguna mengeklik tombol *log out*
Hasil Yang Diharapkan	Pengguna keluar dari aplikasi
Hasil Yang Didapat	Pengguna keluar dari aplikasi
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman *log in* aplikasi

5.4 Pengujian *Performance*

ERP Retail merupakan aplikasi yang menapung data besar dan membutuhkan kemampuan yang cepat dan optimal dalam pengolahan data. Pada tugas akhir ini akan dibuktikan *database* mana yang lebih cepat performanya antara *database* jenis SQL dan jenis NoSQL. Dalam uji coba ini digunakan dataset dengan jumlah dan konten yang sama agar setara beban yang digunakan.

5.4.1.1 Pengujian *Insert*

Pada pengujian ini akan dibuktikan kemampuan *insert* dari database MySQL dan MongoDB. Adapun prosesnya akan dijabarkan lebih detail di bawah ini.

1. Membuat *query insert* untuk testing. *Query* akan dijabarkan pada **Gambar 5.1** dan **Gambar 5.2**

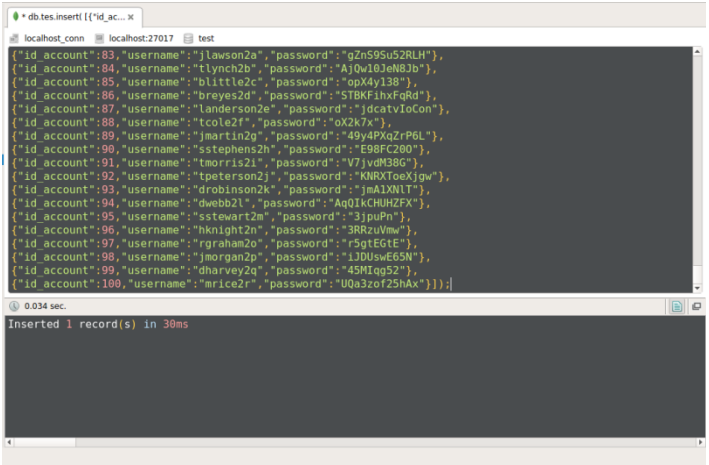
```
db.tes.insert(
{“id_account”:1,”username”:”jlawson2a”,”password”:”gZnS
9Su52RLH”})
```

Gambar 5.1 *Query* Insert MongoDB

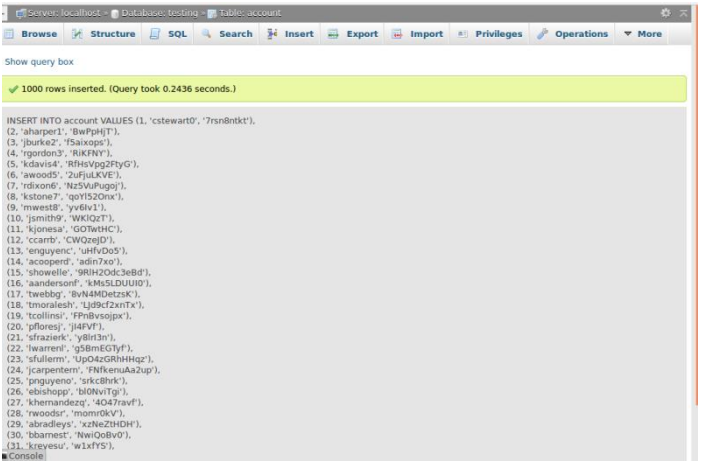
```
INSERT INTO account VALUES (1,' jlawson2a',' gZnS9Su52RLH');
```

Gambar 5.2 *Query* Insert MySQL

2. Kemudian *query* dijalankan pada *collection* tujuan. Uji coba dilakukan 8 kali dimulai dari *insert* 1000 data untuk uji coba pertama dan lanjut kelipatan 1000 pada percobaan berikutnya dengan bertahap. Contoh hasil uji coba dijabarkan di bawah ini:



Gambar 5.3 Hasil *Insert* 1000 Data MongoDB



Gambar 5.4 Hasil *Insert* 1000 Data MySQL

3. *Runtime* hasil uji coba dicatat setiap transaksi dari mulai *insert* 1000 data pertama, 2000 data kedua, 3000 data ketiga, dan seterusnya. Hasil *record* data runtime ditampilkan pada **Tabel 5.17** dan **Tabel 5.18**

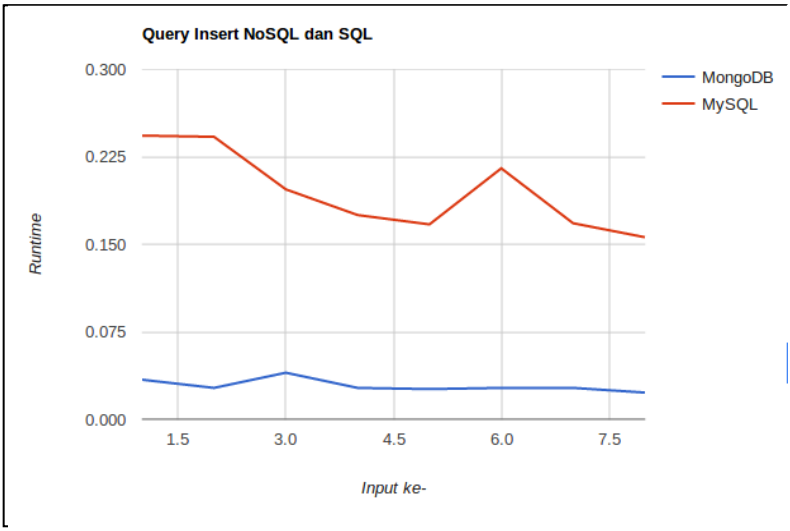
Tabel 5.17 Hasil *Runtime Insert* MongoDB

No	Jumlah Data	Runtime
1	1000	0.034
2	1000	0.027
3	1000	0.04
4	1000	0.027
5	1000	0.026
6	1000	0.027
7	1000	0.027
8	1000	0.023
Rata-Rata		0.028875

Tabel 5.18 Hasil *Runtime Insert* MySQL

No	Jumlah Data	Runtime
1	1000	0.243
2	1000	0.242
3	1000	0.197
4	1000	0.175
5	1000	0.167
6	1000	0.215
7	1000	0.168
8	1000	0.156
Rata-Rata		0.195375

4. Berdasarkan tabel diatas dibuat graik perbandingan *runtime insert* dari MySQL dan MongoDB agar lebih mudah untuk dianalisa. Grafik ditampilkan pada **Gambar 5.5**



Gambar 5.5 Grafik Perbandingan *Runtime Insert*

5.4.1.2 Pengujian *Join*/Agregasi

Pada pengujian ini akan dibuktikan kemampuan *join*/agregasi dari database MySQL dan MongoDB. Adapun prosesnya akan dijabarkan lebih detail di bawah ini.

1. Membuat *query join*/agregasi untuk testing. *Query* akan dijabarkan pada **Gambar 5.6** dan **Gambar 5.7**

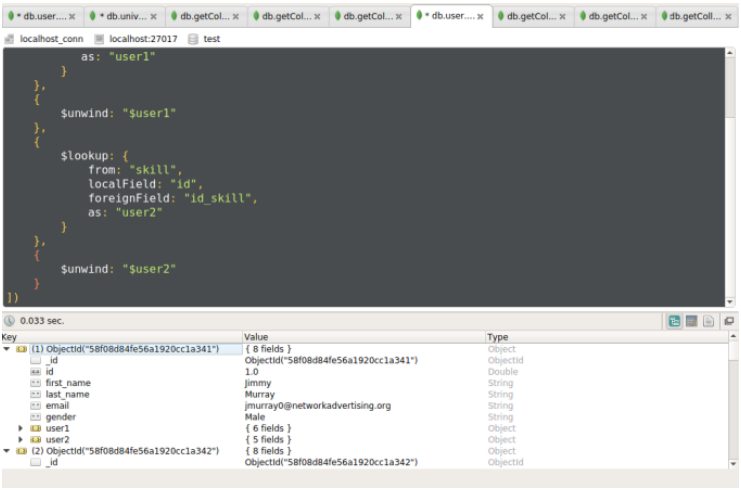
```
db.user.aggregate([
   {
      $lookup: {
         from: "dosen",
         localField: "id",
         foreignField: "id_dosen",
         as: "user1"
         }
      },
      {
         $undwind: "$user1"
      }
])
```

Gambar 5.6 *Query* Agregasi MongoDB

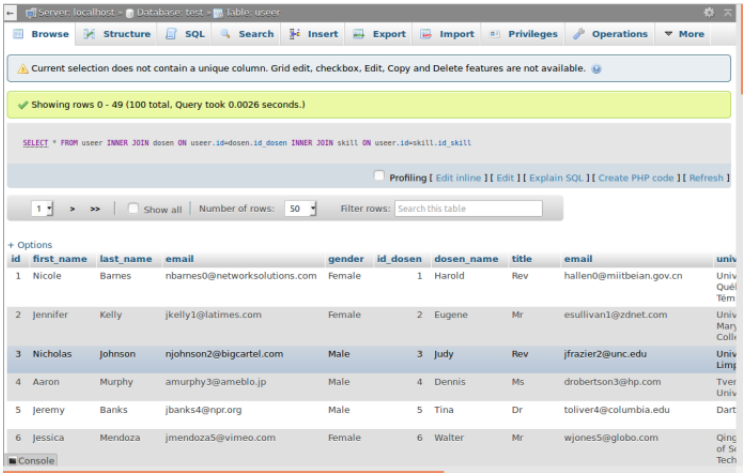
```
SELECT * FROM user INNER JOIN dosen ON user.id=dosen.id_dosen;
```

Gambar 5.7 *Query Join* MySQL

2. Kemudian *query* dijalankan pada *collection* tujuan. Uji coba dilakukan pada 8 tabel secara bertahap dimulai dengan 2 tabel, 3 tabel, 4 tabel, dan seterusnya. Contoh hasil uji coba dijabarkan di bawah ini:



Gambar 5.8 Hasil Agregasi 3 *Collection* MongoDB



Gambar 5.9 Hasil *Join* 3 Tabel MySQL

3. *Runtime* hasil uji coba dicatat setiap transaksi dari mulai *join*/agregasi 2 tabel, 3 tabel, 4 tabel, dan seterusnya. Hasil *record* data runtime ditampilkan pada **Tabel 5.19** dan **Tabel 5.20**

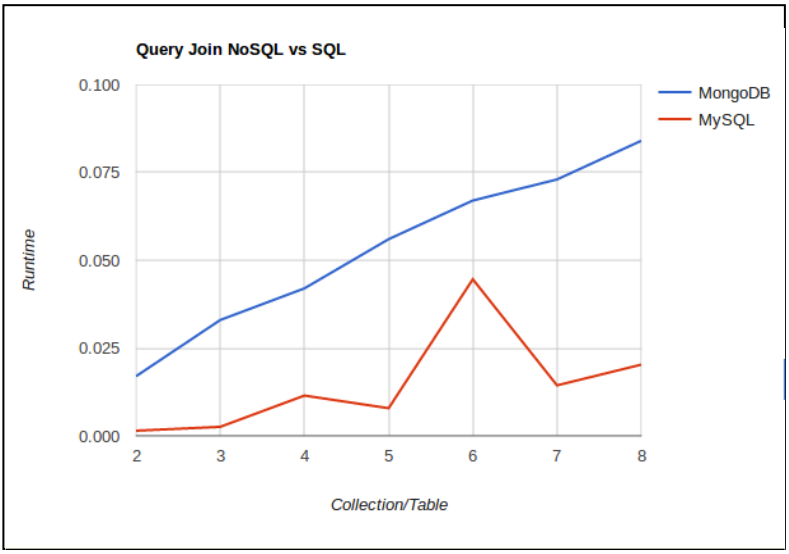
Tabel 5.19 Hasil *Runtime Join*/Agregasi MongoDB

No	Jumlah Collection	Runtime
1	2	0.017
2	3	0.033
3	4	0.042
4	5	0.056
5	6	0.067
6	7	0.073
7	8	0.084
Rata-Rata		0.05314286

Tabel 5.20 Hasil *Runtime Join*/Agregasi MySQL

No	Jumlah Table	Runtime
1	2	0.0015
2	3	0.0026
3	4	0.0115
4	5	0.0079
5	6	0.0446
6	7	0.0144
7	8	0.0203
Rata-Rata		0.01468571

4. Berdasarkan tabel diatas dibuat graik perbandingan *runtime join*/agregasi dari MySQL dan MongoDB agar lebih mudah untuk dianalisa. Grafik ditampilkan pada **Gambar 5.10**



Gambar 5.10 Grafik Perbandingan *Runtime Join*/Agregasi

5.4.1.3 Pengujian *Select*

Pada pengujian ini akan dibuktikan kemampuan *select* dari database MySQL dan MongoDB. Adapun prosesnya akan dijabarkan lebih detail di bawah ini.

1. Membuat *query select* untuk testing. *Query* akan dijabarkan pada **Gambar 5.11** dan **Gambar 5.12**

```
db.getCollection('dosen').find()
```

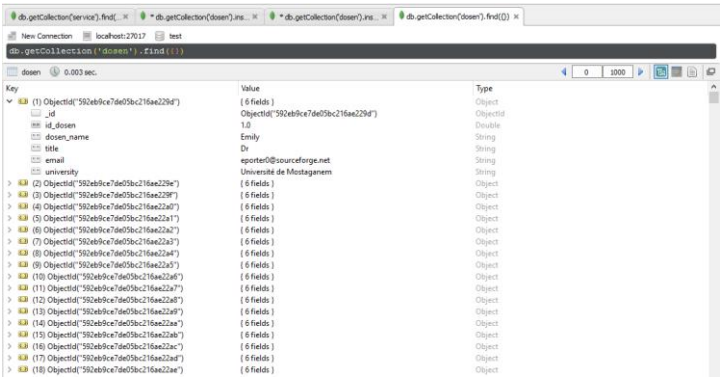
Gambar 5.11 *Query Select* MongoDB

```
SELECT * FROM 'dosen' WHERE 1;
```

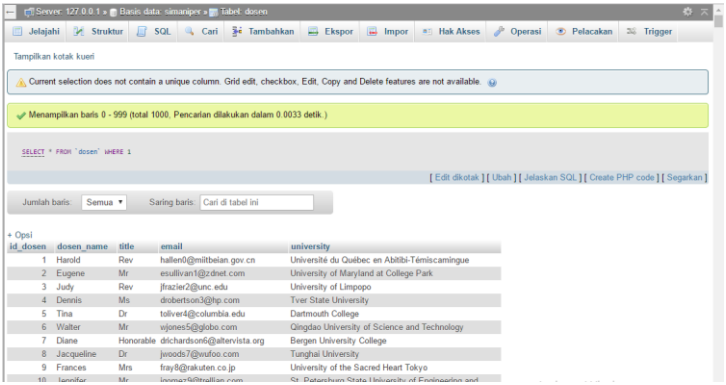
Gambar 5.12 *Query Select* MySQL

2. Kemudian *query* dijalankan pada *collection* tujuan. Uji coba dilakukan 8 kali dimulai dari *select* 1000 data untuk uji coba pertama dan lanjut kelipatan 1000 pada percobaan

berikutnya dengan bertahap. Contoh hasil uji coba dijabarkan di bawah ini:



Gambar 5.13 Hasil *Select* 1000 Data MongoDB



Gambar 5.14 Hasil *Select* 1000 Data MySQL

3. *Runtime* hasil uji coba dicatat setiap transaksi dari mulai *select* 1000 data pertama, 2000 data kedua, 3000 data ketiga, dan seterusnya. Hasil *record* data runtime ditampilkan pada **Tabel 5.21** dan **Tabel 5.22.**

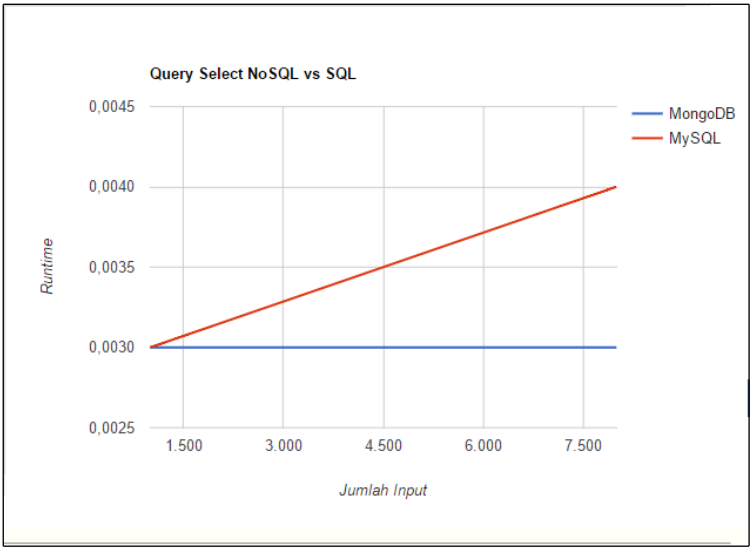
Tabel 5.21 Hasil *Runtime Select* MongoDB

No	Jumlah Data	Runtime
1	1000	0.003
2	2000	0.003
3	3000	0.003
4	4000	0.003
5	5000	0.003
6	6000	0.003
7	7000	0.003
8	8000	0.003
Rata-Rata		0.003

Tabel 5.22 Hasil *Runtime Select* MySQL

No	Jumlah Data	Runtime
1	1000	0.003
2	2000	0.004
3	3000	0.005
4	4000	0.011
5	5000	0.011
6	6000	0.016
7	7000	0.012
8	8000	0.035
Rata-Rata		0.012125

4. Berdasarkan tabel diatas dibuat graik perbandingan *runtime select* dari MySQL dan MongoDB agar lebih mudah untuk dianalisa. Grafik ditampilkan pada **Gambar 5.15**



Gambar 5.15 Grafik Perbandingan *Runtime Select*

5.4.1.4 Pengujian *Update*

Pada pengujian ini akan dibuktikan kemampuan *update* dari database MySQL dan MongoDB. Adapun prosesnya akan dijabarkan lebih detail di bawah ini.

1. Membuat *query update* untuk testing. *Query* akan dijabarkan pada **Gambar 5.16** dan **Gambar 5.17**

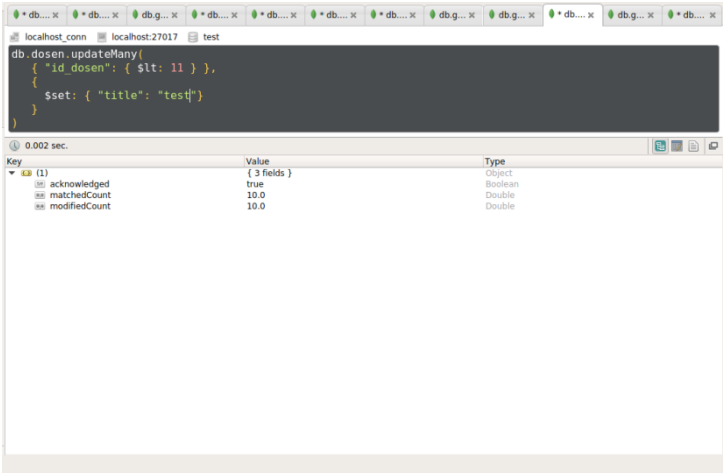
```
db.dosesn.updateMany(
        { “id_dosen”: {$lt:11}},
        {
          $set: { “title”: “test”}
        }
)
```

Gambar 5.16 *Query Update* MongoDB

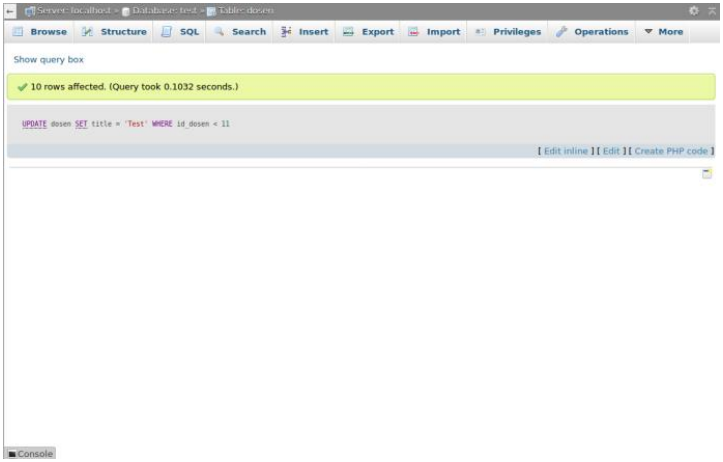
```
UPDATE dosen SET title = “Test” WHERE id_dosen < 11;
```

Gambar 5.17 *Query Update* MySQL

2. Kemudian *query* dijalankan pada *collection* tujuan. Uji coba dilakukan 8 kali dimulai dari *update* 10 data untuk uji coba pertama dan lanjut kelipatan 10 pada percobaan berikutnya dengan bertahap. Contoh hasil uji coba dijabarkan di bawah ini:



Gambar 5.18 Hasil *Update* 10 Data MongoDB



Gambar 5.19 Hasil *Update* 10 Data MySQL

3. *Runtime* hasil uji coba dicatat setiap transaksi dari mulai *update* 10 data pertama, 20 data kedua, 30 data ketiga, dan seterusnya. Hasil *record* data runtime ditampilkan pada **Tabel 5.23** dan **Tabel 5.24.**

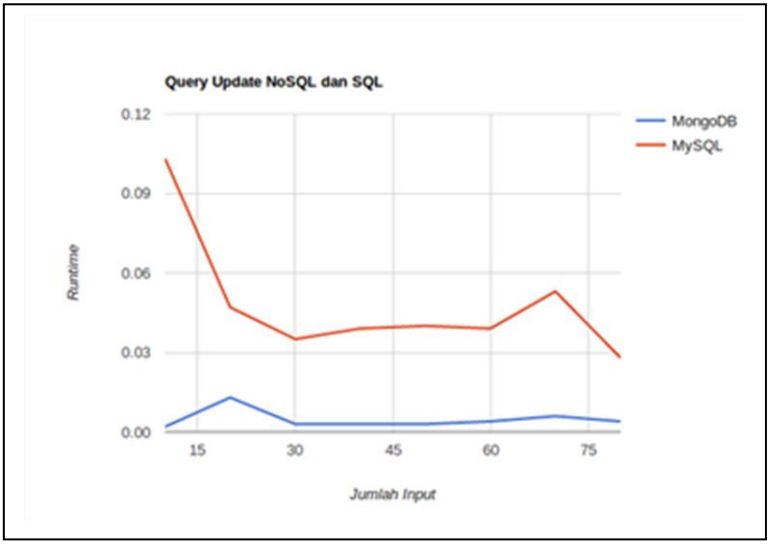
Tabel 5.23 Hasil *Runtime Update* MongoDB

No	Jumlah Data	Runtime
1	10	0.002
2	20	0.013
3	30	0.003
4	40	0.003
5	50	0.003
6	60	0.004
7	70	0.006
8	80	0.004
Rata-Rata		0.00475

Tabel 5.24 Hasil *Runtime Update* MySQL

No	Jumlah Data	Runtime
1	10	0.103
2	20	0.047
3	30	0.035
4	40	0.039
5	50	0.04
6	60	0.039
7	70	0.053
8	80	0.028
Rata-Rata		0.048

4. Berdasarkan tabel diatas dibuat graik perbandingan *runtime update* dari MySQL dan MongoDB agar lebih mudah untuk dianalisa. Grafik ditampilkan pada **Gambar 5.20**



Gambar 5.20 Grafik Perbandingan *Runtime Update*

5.4.1.5 Pengujian *Delete*

Pada pengujian ini akan dibuktikan kemampuan *delete* dari database MySQL dan MongoDB. Adapun prosesnya akan dijabarkan lebih detail di bawah ini.

1. Membuat *query delete* untuk testing. *Query* akan dijabarkan pada **Gambar 5.21** dan **Gambar 5.22**

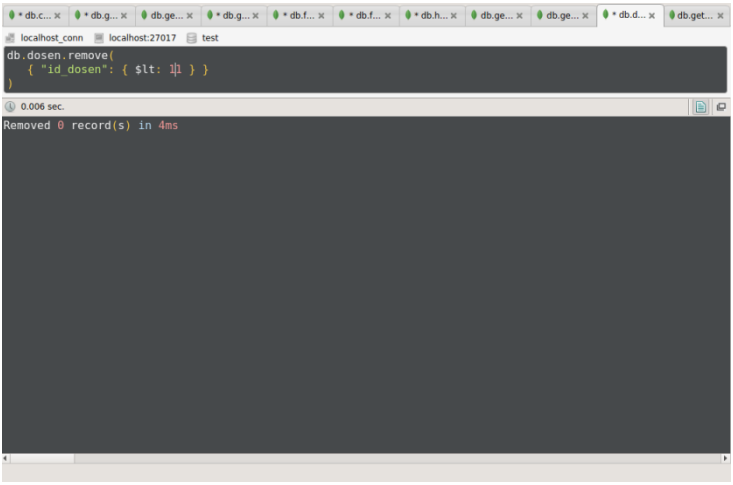
```
db.dosen.remove(
        {“id_dosen”: { $lt : 11} }
)
```

Gambar 5.21 *Query Delete* MongoDB

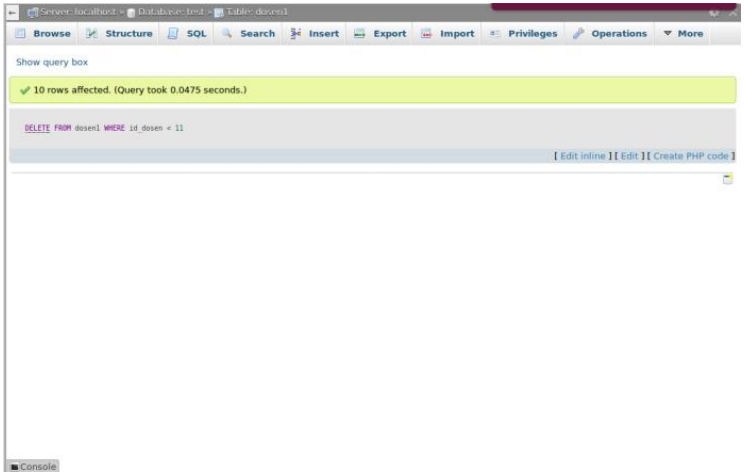
```
DELETE FROM dosen1 WHERE id_dosen < 11;
```

Gambar 5.22 *Query Delete* MySQL

2. Kemudian *query* dijalankan pada *collection* tujuan. Uji coba dilakukan 8 kali dimulai dari *delete* 10 data untuk uji coba pertama dan lanjut kelipatan 10 pada percobaan berikutnya dengan bertahap. Contoh hasil uji coba dijabarkan di bawah ini:



Gambar 5.23 Hasil *Delete* 10 Data MongoDB



Gambar 5.24 Hasil *Delete* 10 Data MySQL

3. *Runtime* hasil uji coba dicatat setiap transaksi dari mulai *delete* 10 data pertama, 20 data kedua, 30 data ketiga, dan

seterusnya. Hasil *record* data runtime ditampilkan pada **Tabel 5.25** dan **Tabel 5.26**

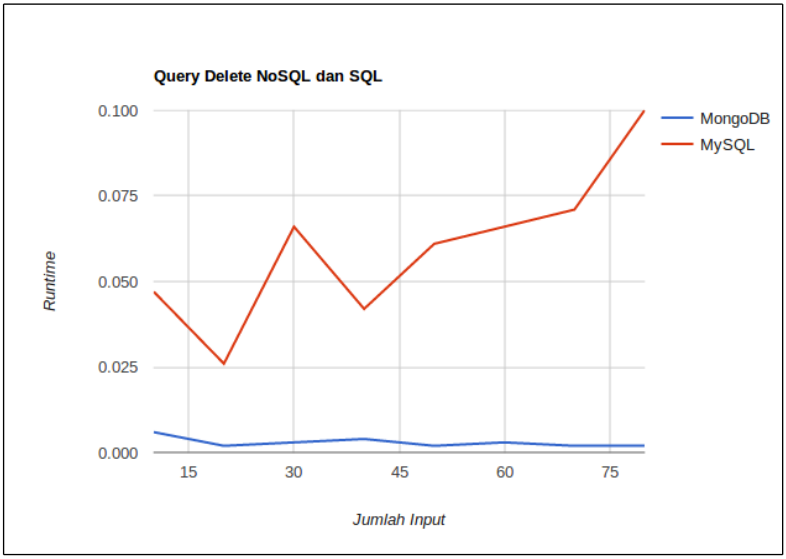
Tabel 5.25 Hasil *Runtime Delete* MongoDB

No	Jumlah Data	Runtime
1	10	0.006
2	20	0.002
3	30	0.003
4	40	0.004
5	50	0.002
6	60	0.003
7	70	0.002
8	80	0.002
Rata-Rata		0.003

Tabel 5.26 Hasil *Runtime Delete* MySQL

No	Jumlah Data	Runtime
1	10	0.047
2	20	0.026
3	30	0.066
4	40	0.042
5	50	0.061
6	60	0.066
7	70	0.071
8	80	0.1
Rata-Rata		0.059875

4. Berdasarkan tabel diatas dibuat graik perbandingan *runtime delete* dari MySQL dan MongoDB agar lebih mudah untuk dianalisa. Grafik ditampilkan pada **Gambar 5.25**

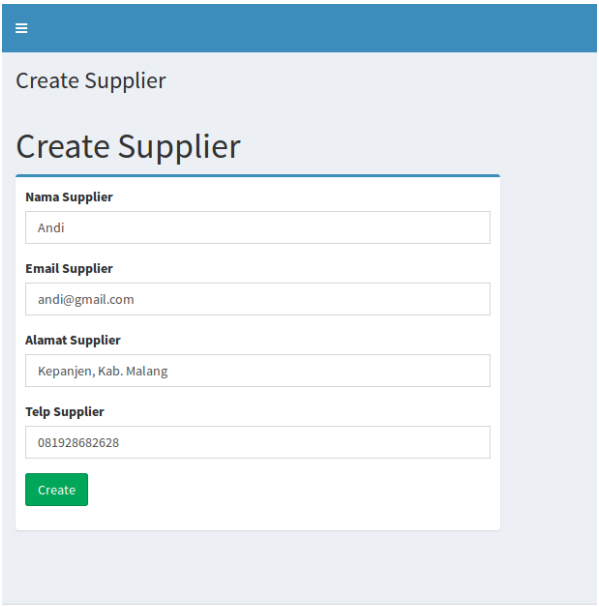


Gambar 5.25 Grafik Perbandingan *Runtime Delete*

5.5 Pengujian *Flexibility*

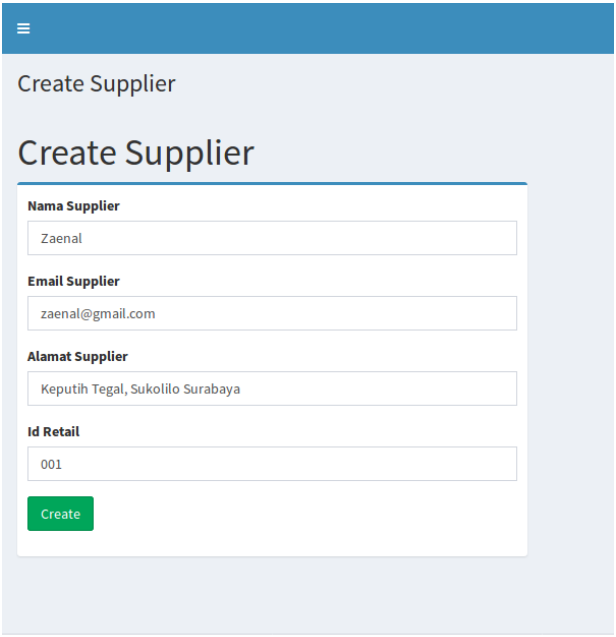
Pada tugas akhir ini aplikasi ERP Retail dirancang sebagai aplikasi yang memiliki skema yang dinamis dari level *interface*, *web service*, hingga *database*. Oleh karena itu akan dibuktikan pada tingkat *database* apakah *database* tipe NoSQL mampu menangani kebutuhan ini, yang mana *database* cukup rumit bahkan tidak bisa menanganinya. Adapun proses pembuktian akan dijabarkan di bawah ini.

1. Buat satu *collection* sebagai lokasi penyimpanan data yang berbeda dari setiap tenant.
2. Rancang aplikasi ERP Retail menjadi aplikasi yang dinamis. Dalam artian setiap tenant bebas merubah modul hingga *form* untuk membentuk skema yang menjadi kebutuhan masing-masing tenant. Contoh hasil daftar form yang berbeda untuk membentu skema *tenant* dijabarkan di bawah ini.



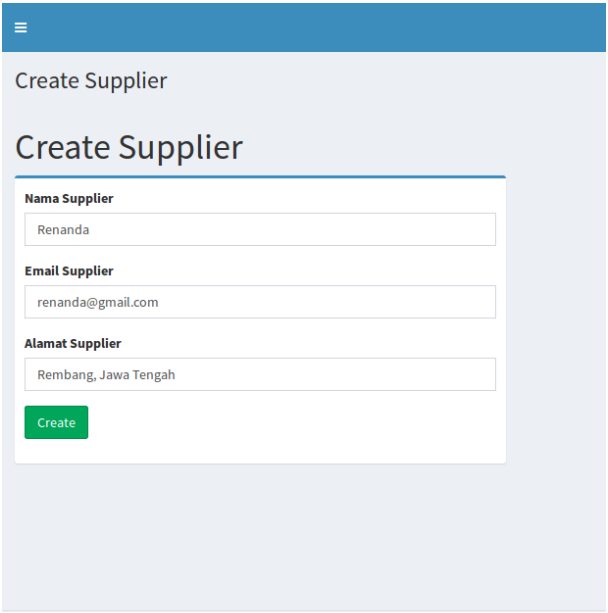
Gambar 5.26 Form 1 Membentuk Skema A

Pada **Gambar 5.26** menampilkan *form create* pada modul supplier yang membentuk skema A dengan *field* nama supplier, email supplier, alamat supplier, dan telepon supplier.



Gambar 5.27 Form 2 Membentuk Skema B

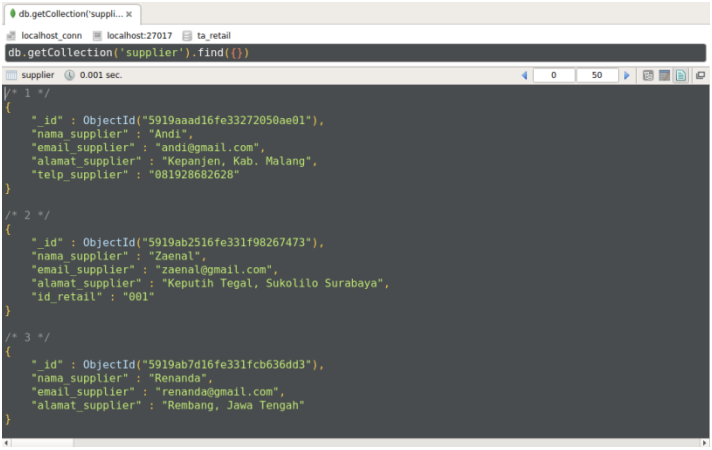
Pada **Gambar 5.27** menampilkan *form create* pada modul supplier yang membentuk skema B dengan *field* nama supplier, email supplier, alamat supplier, dan id retail.



Gambar 5.28 Form 3 Membentuk Skema C

Pada **Gambar 5.28** menampilkan *form create* pada modul supplier yang membentuk skema C dengan *field* nama supplier, email supplier, dan alamat supplier.

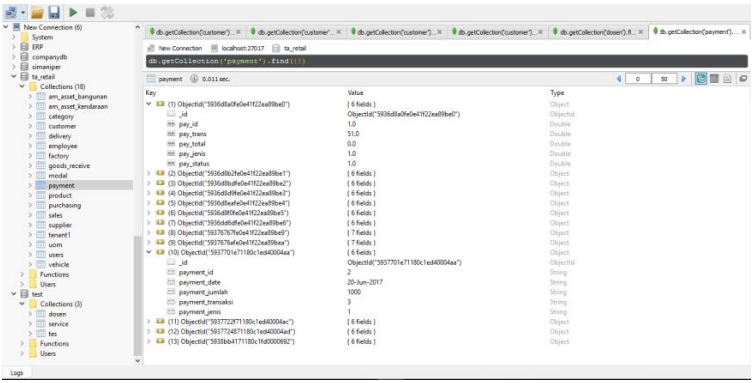
3. Rancang Dari ketiga *form* data diatas hasil *store* data ditempatkan pada satu collection. Hasil data penyimpanan pada collection ditampilkan pada **Gambar 5.29**



Gambar 5.29 *Colletion* Data Hasil Skema Tenant

5.5.2 Optimasi Flexibility

Pada tugas akhir ini aplikasi ERP Retail dioptimasi untuk mengikuti pemetaan NoSQL *database*. Optimasi ditawarkan berdasarkan analisa performa *database* NoSQL dengan tujuan mendapatkan hasil aplikasi yang optimal. Pada optimasi kali ini diterapkan dengan menggabungkan 2 Tabel (dalam MySQL) yang saling berelasi datanya dengan tujuan menghindari transaksi *join*/agregasi yang merupakan kelemahan dari *database* NoSQL. Adapun hasil implementasinya ditunjukkan pada **Gambar 5.30**.



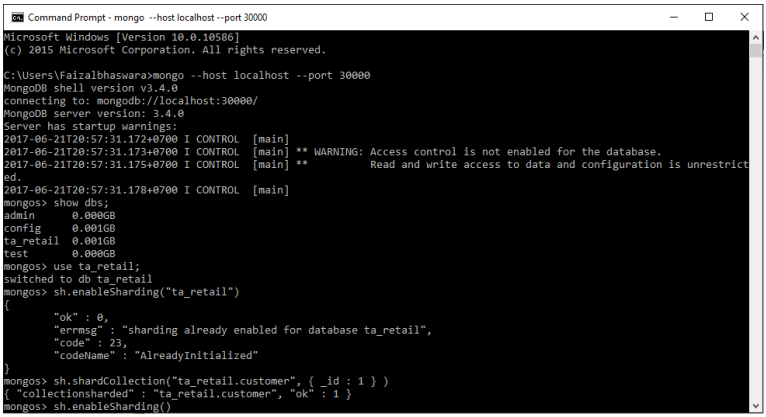
Gambar 5.30 Optimasi *Flexibility*

Pada **Gambar 5.30** ditunjukkan penggabungan data dari tabel *payment master* dan tabel *payment.* Karena data *payment master* dibutuhkan untuk memproses data baru untuk proses *payment.*

5.6 Pengujian *Scalability*

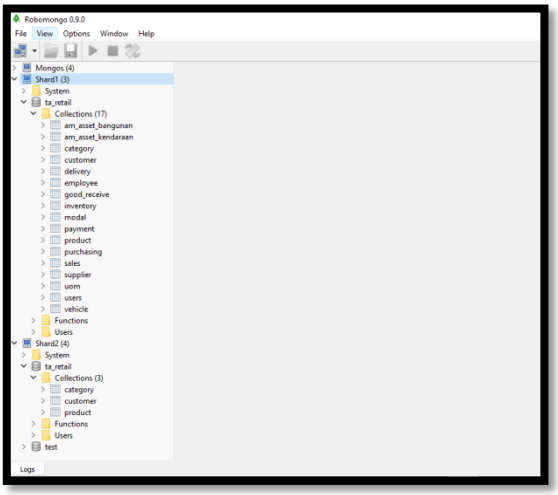
Pada tugas akhir ini aplikasi ERP Retail dirancang dengan *database* yang mampu menangani pertumbuhan data dengan cara mendistribusikannya ke *database* lain. Adapun proses uji coba pendistribusian data akan dijabarkan di bawah ini.

1. Inisialisasi *collection* yang akan didistribusikan. Dalam uji coba ini akan digunakan *collection customer*.



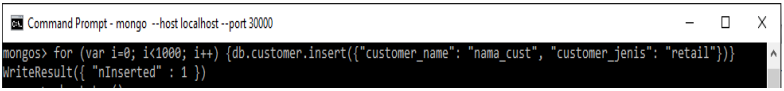
Gambar 5.31 Inisialisasi *Collection* untuk *Sharding*

2. Setelah diinisialisasikan, *collection* akan didistribusikan ke shard1 dan shard2.

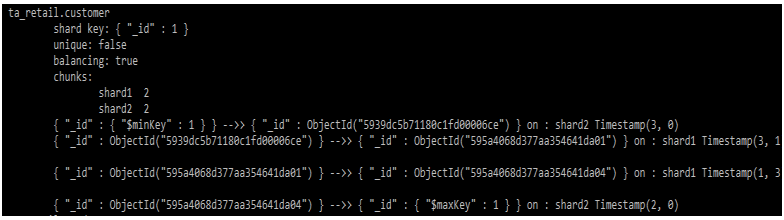


Gambar 5.32 Collection Customer pada Shard1 dan Shard2

3. Lalu dilakukan uji coba melakukan insert 1000 data *raw.*

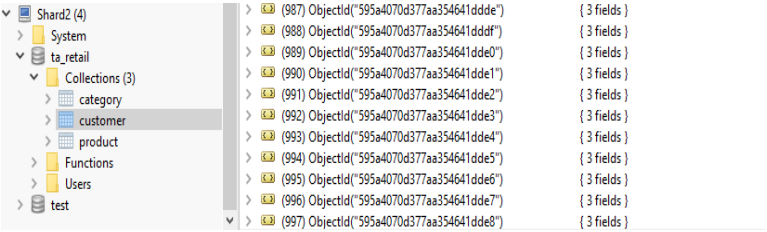


Gambar 5.33 Uji Coba Insert 1000 data *raw*

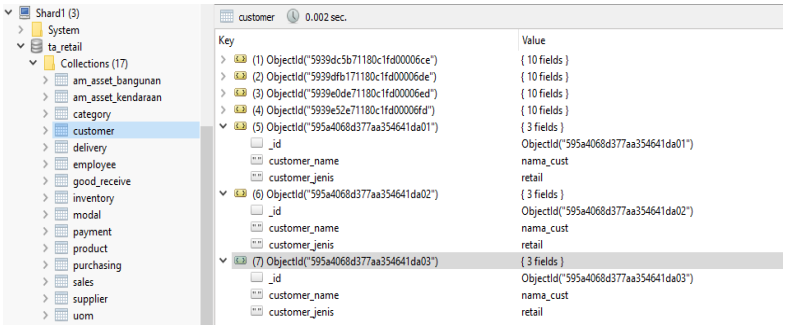


Gambar 5.34 Status Distribusi 1000 data *raw* pada Shard

4. Data *raw* secara otomatis terdistribusi ke shard1 dan shard2 dengan pembagian yang ditentukan oleh mongos. Pada kali ini 997 data terdistribusi ke shard2 dan 3 data sisa ke shard1.



Gambar 5.35 Data *Raw* pada Shard2



Gambar 5.36 Data *Raw* pada Shard1

5.7 Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian *performance* dan evaluasi pengujian *flexibility* partisipan terhadap aplikasi.

5.7.1 Evaluasi Pengujian Fungsionalitas

Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada **Tabel 5.27**. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik kesimpulan bahwa fungsionalitas dari aplikasi telah dapat bekerja sesuai dengan yang diharapkan.

Tabel 5.27 Evaluasi Pengujian Fungsionalitas

No.	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Skenario	Hasil
1	UC-001	*Manage Category*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
2	UC-002	*Manage Product*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
			Skenario 1	Berhasil

No.	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Skenario	Hasil
3	UC-003	*Manage Uom*	Skenario 2	Berhasil
			Skenario 3	Berhasil
4	UC-004	*Manage Supplier*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
5	UC-005	*Manage Customer*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
6	UC-006	*Manage Delivery*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
7	UC-007	*Manage Modal*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
8	UC-008	*Manage Staff*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
9	UC-009	*Manage Inventory*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
10	UC-010	*Manage Sales*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
11	UC-011	*Manage Purchasing*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
12	UC-012	*Manage Payment*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
13	UC-013	*Manage Asset*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
14	UC-014	*Manage Register*	Skenario 1	Berhasil
15	UC-015	*Manage Log In*	Skenario 1	Berhasil
16	UC-016	*Manage Log Out*	Skenario 1	Berhasil

5.7.2 Evaluasi Pengujian *Performance*

Rangkuman mengenai hasil pengujian *performance* dapat dilihat pada **Tabel 5.11**. Berdasarkan data pada tabel tersebut, menunjukkan bahwa *database* MongoDB dengan tipe NoSQL lebih unggul performanya dari *database* MySQL dengan tipe SQL dalam tansaksi *insert*, *select*, *update*, dan *delete*. Namun untuk tansaksi *join*/agregasi database MySQL lebih unggul daripada *database* MongoDB.

Tabel 5.28 Hasil *Performance* MongoDB dan MySQL

No.	*Action*	MongoDB	MySQL	Selisis
1	*Insert*	0.028	0.195	0.167
2	*Join*/Agregasi	0.053	0.014	0.039
3	*Select*	0.003	0.012	0.009
4	*Update*	0.004	0.048	0.044
5	*Delete*	0.003	0.059	0.056

5.7.3 Evaluasi Pengujian *Flexibility*

Hasil pengujian kemampuan *flexibility* dapat dilihat pada **Tabel 5.12**. Berdasarkan data pada tabel tersebut, menunjukkan bahwa *database* MongoDB dengan tipe NoSQL mampu mengani data dengan yang dinamis.

Tabel 5.29 Hasil *Flexibility* MongoDB dan MySQL

No.	*Action*	MongoDB	MySQL
1	*Insert* dengan data yang dinamis	✓	✘

5.7.4 Evaluasi Pengujian *Scalability*

Hasil pengujian kemampuan *scalability* dapat dilihat pada **Tabel 5.30**.

Tabel 5.30 Hasil Uji Coba *Scalability* MongoDB

No.	*Action*	Hasil
1	Distribusi 1000 data raw ke shard1 dan shard2	Berhasil

5.7.5 Evaluasi *Database* NoSQL dan SQL

Pada subbab ini akan dijelaskan evaluasi pada *database* NoSQL dan SQL dengan tujuan mendapatkan *database* yang cocok untuk diterapkan pada aplikasi ERP Retail.

5.7.5.1 Pemilihan *Database* NoSQL atau SQL

Dalam proses membangun sebuah aplikasi sangat penting untuk menggali kebutuhan aplikasi dan melakukan analisa dengan harapan aplikasi yang dibangun bisa menjadi aplikasi yang sempurna dalam segala fitur. Termasuk faktor yang sangat penting dalam analisanya adalah pemilihan jenis *database* yang tepat agar aplikasi berjalan secara optimal. Adapun analisa pemilihan jenis *database* agar aplikasi berjalan dengan baik dijelaskan pada **Tabel 5.31**.

Tabel 5.31 Analisa Pemilihan Jenis *Database*

SQL	NoSQL
- Aplikasi yang mementingkan integritas data - Aplikasi yang dibangun	- Aplikasi membutuhkan penyimpanan data yang relatif besar, namun terdapat

SQL	NoSQL
membutuhkan query yang rumit (kompleks) - Menggunakan metode *waterfall development* dalam membangun aplikasi - Aplikasi yang memiliki ketergantungan yang kuat pada transaksi lebih dari satu tabel	keterbatasan *resource* (komputer/server) - Data bersifat tidak terstruktur atau biasa disebut sebagai *Schemaless Data Representation* - Aplikasi mengutamakan faktor kecepatan dan skalabilitas - Menggunakan metode *agile development* dalam membangun aplikasi - Aplikasi membutuhkan proses *write/insert* dalam jumlah yang besar dan waktu yang singkat

5.7.5.2 Evaluasi *Database* pada Aplikasi ERP Retail

Aplikasi ERP Retail dibangun dengan beberapa fitur dan struktur sesuai *requirement* untuk menjalankan proses bisnis yang telah dibuat. Dari kebututuhan fitur maupun struktur dianalisa agar bisa diketahui *database* jenis apa yang tepat untuk diterapkan pada aplikasi. Adapun penjelasan dari aplikasi ERP Retail dijelaskan pada **Tabel 5.32**.

Tabel 5.32 Evaluasi *Database* pada Aplikasi ERP Retail

Kebutuhan	ERP Retail	Database
Transaksi, Join	4 Tabel/Collection	SQL
CRUD	9 Tabel/Collection	NoSQL
Struktur data	Dinamis	NoSQL
Performa	Cepat	NoSQL

Dari penjelasan beberapa faktor diatas, maka dapat diambil kesimpulan bahwa aplikasi ERP Retail akan lebih optimal jika menggunakan *database* jenis NoSQL.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan Tugas Akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. *Database* MongoDB jenis NoSQL terbukti memiliki performa yang lebih cepat dalam proses transaksi CRUD daripada SQL dengan jumlah selisih *runtime insert* 0,167, *select* 0,009, *update* 0,44, *delete* 0,056. Namun lemah untuk menghadapi transaksi *join*/agregasi dengan jumlah selisih *runtime* 0,039.
2. Dengan pembuktian *database* NoSQL mampu menerima data dengan skema dan struktur yang berbeda dari setiap *tenant*, maka database NoSQL mampu memenuhi kebutuhan aplikasi ERP Retail yang bisa mensupport perbedaan skema dari setiap tenant dan mengikuti perkembangan proses bisnis kedepannya. Oleh karena itu database NoSQL akan lebih tepat untuk digunakan.
3. Kebutuhan aplikasi ERP Retail dalam mengatasi data yang terus berkembang dapat diatasi oleh database NoSQL dengan sifatnya yang *scalable* dibuktikan dengan mendistribusikan 1000 data *raw* ke dua shard (shard1 dan shard2).

6.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada

hasil perancangan, implementasi dan pengujian yang telah dilakukan:

1. Pengembangan aplikasi ERP Retail kedepannya dapat mengoptimasi dalam *join*/agregasi tabel agar kemampuan dalam menangani data lebih optimal.
2. Penggunaan *database engine* yang sudah menerapkan *multitenant* agar mempermudah proses pembuatan aplikasi dalam orientasi *multitenancy*.

DAFTAR PUSTAKA

[1] "Wiley: Integrated Business Processes with ERP Systems - Simha R. Magal, Jeffrey Word." [Online]. Available: http://www.wiley.com/WileyCDA/WileyTitle/productCd-EHEP001815.html. [Accessed: 29-May-2017].

[2] "Andrejs Tambovcevs and Tatjana Tambovceva 'ERP system implementation: benefits and economic effectiveness' Faculty of Computer Science and Information Technology Riga Technical University Riga, Latvia 2013." .

[3] "Mufid Itsnaini Zain and M. Rudyanto Arief 'PERBANDINGAN STRUKTUR PENYIMPANAN DAN PERFORMANSI NOSQL MONGODB DENGAN DBMS SQL SERVER' Jurusan Teknik Informatika STMIK AMIKOM YOGYAKARTA." .

[4] "Ameya Nayak, Anil Poriya, and Dikshay Poojary 'Type of NOSQL Databases and its Comparison with Relational Databases' Dept. of Computer Engineering Thakur College of Engineering and Technology University of Mumbai, March 2013." .

[5] J. R. Lourenço, B. Cabral, P. Carreiro, M. Vieira, and J. Bernardino, "Choosing the right NoSQL database for the job: a quality attribute evaluation," *J. Big Data*, vol. 2, no. 1, p. 18, Aug. 2015.

[6] "Divya Chauhan and K. L. Bansal 'Using the Advantages of NOSQL: A Case Study on MongoDB' Himachal Pradesh University Summerhill,Shimla,India." .

[7] "T. Connolly and C. Begg, Database Systems: A Practical Approach to Design, Implementation, and Management, Harlow: Pearson Education, 2005." .

[8] "Principles Of Distributed Database Systems - M. Tamer Ozsu, Patrick Valduriez.pdf." .

[9] “Multi-Tenant Data Architecture.” [Online]. Available: https://msdn.microsoft.com/en-us/library/aa479086.aspx. [Accessed: 18-Dec-2016].
[10] “Dipina Damodaran B , Shirin Salim and Surekha Marium Vargese ‘PERFORMANCE EVALUATION OF MYSQL AND MONGODB DATABASES’ Department of Computer Engineering, M A College of Engineering, Kothamangalam, Kerala, India April 2016.” .
[11] “Generating Code with Gii - Getting Started - The Definitive Guide to Yii 2.0.” [Online]. Available: http://www.yiiframework.com/doc-2.0/guide-start-gii.html. [Accessed: 04-Jun-2017].
[12] “Deploy a Sharded Cluster — MongoDB Manual 3.4,” *https://github.com/mongodb/docs/blob/master/source/tutorial/deploy-shard-cluster.txt*. [Online]. Available: http://docs.mongodb.com/manual/tutorial/deploy-shard-cluster. [Accessed: 04-Jun-2017].

BIODATA PENULIS

Penulis, **Faizal Anugrah Bhaswara**, lahir di Malang, 25 Juni 1995. Penulis menempuh pendidikan sekolah menengah pertama di SMP Negeri 5 Malang dan melanjutkan pendidikannya di SMA Negeri 1 Malang. Selanjutnya penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika, Fakultas Teknologi dan Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif menjadi santri pondok di Pondok Pesantren Khoirul Huda II Surabaya dan aktif pada beberapa kepanitiaan seperti LKMM Pra-TD, Schematics, dan kegiatan lainnya. Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Manajemen Informasi (MI). Penulis dapat dihubungi melalui email: faisalanugrah48@gmail.com.