Analisis dan Implementasi Basis Data Terdistribusi Horizontal pada MongoDB untuk ClikKB BKKBN Regional Jawa Barat

Analysis and Implementation of Horizontal Distributed Database on MongoDB for ClikKB BKKBN West Java Region

Nunit Prihatoni Siregar[1], Kemas Rahmat S. W., S.T., M.Eng [2], Alfian Akbar G., S.T., M.T.[3]

[1,2]Prodi S1 Teknik Informatika, Fakultas Informatika, Universitas Telkom
[3] Prodi D3 Teknik Informatika, Fakultas Ilmu Terapan, Universitas Telkom
[1]nunitsiregar@telkomuniversity.ac.id, [2]bagindok3m45@gmail.com, [3]panggil.aku.ian@gmail.com

Abstrak

ClikKB merupakan sistem yang dimiliki oleh dinas BKKBN regional Jawa Barat. Format data ClikKB yang disimpan dalam file excel menyebabkan pengisian banyak bernilai *null*. Nilai *null* pada ClikKB menyebabkan struktur data menjadi semi terstruktur. Data semi terstruktur dapat disebut dengan *nonuniform* data yaitu dokumen yang memiliki atribut yang berbeda-beda. Dari berbagai varian NoSQL, *document oriented* digunakan untuk mengatasi masalah *nonuniform* data.

Karena mencakup wilayah Jawa Barat, maka data dari ClikKB cukup besar. Sistem ini memiliki proses *read* dan *write* yang tinggi karena diakses oleh seluruh dinas BKKBN Jawa Barat. Hal ini akan mempengaruhi performansi sistem. Untuk meningkatkan performansi sistem, dilakukan proses distribusi yaitu teknik penyimpanan basis data yang dipecah kebeberapa lokasi penyimpanan yang terhubung dengan jaringan. Salah satu DBMS dari NoSQL adalah MongoDB. MongoDB memiliki kemampuan untuk distribusi data. Yang diimplementasikan ada 3 arsitektur yaitu penerapan *Document Oriented Database single server*, penerapan *sharding* dengan 2 node, dan *sharding* 3 node. Jenis fragmentasi atau pembagian data yang digunakan adalah horizontal specification karena MongoDB memiliki fitur tersebut. Secara umum, diimplementasikannya distribusi pada *document oriented database* ini menghasilkan peningkatan *response time* hingga 167,55% jika dibandingkan tanpa proses distribusi.

Kata Kunci: null value , NoSQL, *Document Oriented Database*, MongoDB, *sharding*.

Abstract

ClikKB is a system owned by the department of regional BKKBN West Java. ClikKB data format stored in an excel file causes many null filling. Null value in ClikKB lead into semi-structured data structure. Semi-structured data can be called with nonuniform data that documents have different attributes. The various of NoSQL variants, document oriented used to overcome the problem of nonuniform data.

Because it covers the West Java area, so ClikKB data is quite large. This system has high accessed about read and write by all departments BKKBN West Java. It will affect system performance. To improve system performance, implemented the distribution process namely of database storage techniques were split to several storage locations that are connected to the network. One of NoSQL DBMS is MongoDB. MongoDB has the capability for data distribution. 3 architecture that is implemented here is a single application of document oriented database server, application sharding with 2 nodes, and sharding 3 nodes. Type of specification or deviding of the data is horizontal specification because MongoDB have it features. In general, the implementation of the distribution the document oriented database response time resulted in an increase of up to 167,55% when compared with no distribution process .

Keywords: null value , NoSQL, Document Oriented Database, MongoDB, sharding.

1 Pendahuluan

ClikKB atau Cepat, Langsung, Informasi Kependudukan dan Keluarga Berencana adalah sistem yang dimiliki oleh dinas BKKBN regional Jawa Barat. Sistem ini digunakan untuk pendataan dan pelaporan data peserta KB [1]. Data ClikKB bersumber dari *survey* lapangan dan disimpan pada *file* excel. Format data ClikKB memungkinkan pengisian menyebabkan banyak *null value*.

Tren basis data yang berkembang saat ini yaitu NoSQL didesain khusus untuk memecahkan permasalahan *scalability* dan *reliability* [2]. NoSQL yang akan digunakan adalah *document oriented* karena memiliki kemampuan mangatasi masalah *nonuniform* data [3]. *Nonuniform* data yaitu data semi terstruktur atau dokumen

yang memiliki atribut yang berbeda-beda. *Null value* pada ClikKB menyebabkan struktur data menjadi semi terstruktur.

Cakupan wilayah dari sistem ClikKB adalah seluruh Jawa Barat. Dari cakupan wilayah yang luas ini, maka ClikKB memiliki data yang cukup besar. Dengan data yang besar ini, maka diperlukan sistem yang handal untuk menanganinya. Salah satu cara yang dapat dilakukan adalah distribusi data. Salah satu DBMS dari NoSQL adalah MongoDB. MongoDB memiliki kemampuan untuk distribusi basis data. Hal ini digunakan untuk menanggulangi masalah data yang cukup besar dan tingginya proses *read* dan *write* pada sistem karena sistem diakses oleh seluruh dinas BKKBN Jawa Barat. *Distributed database* adalah sebuah teknik menyimpan basis data yang dipecah menjadi beberapa lokasi penyimpanan yang terhubung dengan jaringan [4]. Teknik ini bertujuan untuk meningkatkan performansi sistem. Pada tugas akhir ini akan diimplementasikan 3 jenis arsitektur sistem. Arsitektur pertama menggunakan *single server*. Pada arsitektur kedua, mengimplementasikan *sharding* dengan 2 *node* yaitu membagi *server* basis data mejadi 2 kelompok data dan disimpan pada lokasi yang berbeda. Arsitektur ketiga diimplementasikan *sharding* dengan 3 *node*. Adapun jenis fragmentasi atau pembagian data yang digunakan adalah *horizontal fragmentation* dalam hal ini disebut *sharding* karena MongoDB memiliki fitur tersebut. Selain distribusi, mongoDB juga bersifat *schemaless* sehingga mudah dalam pendefinisian model data. MongoDB memiliki model data yang dapat dibentuk sesuai kebutuhan misalnya model non *embedded document*, model *embedded document* dan *normalized* model.

Pada penelitian ini berfokus pada analisa model data *collection Document Oriented Database* untuk ClikKB. Selain itu juga mengimplementasikan model data *collection* ClikKB yang telah dianalisa pada MongoDB. Dan menganalisa performansi sistem distributed *document oriented database* berdasarkan parameter uji *response time* dan *throughput*. Dalam penelitian ini, dimulai dengan studi literature, dilanjutkan dengan pengumpulan data, perancangan sistem, implementasi, pengujian dan analisis hasil dan ditutup dengan penyusunan laporan akhir.

2 Dasar Teori dan Perancangan

2.1 NoSQL

NoSQL adalah sebuah tren yang berkembang dewasa ini sebagai wujud adanya kebutuhan untuk menangani data yang besar dan kebutuhan akan *cluster*. Pramod menyatakan bahwa NoSQL *is very ill-defined* meliputi *schemaless* data, jalan pada *cluster*, dan memiliki kemampuan menjual *traditional consistency* untuk *property* berguna yang lain. Mereka mengklaim NoSQL dapat membangun sistem yang lebih performatif, *scalability* yang lebih baik dan lebih mudah untuk diprogram [3]. Selain itu, NoSQL memiliki beberapa jenis model data yang dapat dipilih sesuai dengan kebutuhan sistem. Beberapa kelebihan NoSQL jika dibandingkan dengan SQL adalah schemaless yaitu NoSQL tidak terikat dengan *schema*. Jadi dalam mendefinikan struktur data tidak harus diawal dan antar *record* tidak harus memiliki *field* yang sama. Hal ini cocok untuk data-data yang *nonuniform* seperti data ClikKB. Dan *large-scale* data Sebuah yaitu dapat menangkap lebih banyak data dan memprosesnya dengan lebih cepat. NoSQL sengaja didesain untuk berjalan pada sistem *cluster* sehingga ini akan lebih baik untuk skenario big data [3].

2.2 *Document Oriented Database*

Menurut Oliver Schmitt ide dasar dari basis data berbasis dokumen adalah bahwa unit terkecil untuk penyimpanan adalah dokumen. Struktur data dari dokumen dapat terdiri atas pasangan *key-value* [7]. Beberapa jenis *database* yang mendukung *Document Oriented Database* adalah CouchDB, MongoDB, OrientDB, RavenDB, dan Terrastore.

2.3 MongoDB

MongoDB adalah DBMS *open source* yang mendukung *Document Oriented Database* yang menyediakan performansi tinggi, *availability* tinggi dan *automatic scaling* [8]. Performansi tinggi karena MongoDB didukung dengan *indexing* sehingga mendukung *query* menjadi lebih cepat dan mencakup *key* dari *document* dan *array*. Ketersediaan tinggi karena MongoDB didukung oleh replikasi. Satu set replika adalah sekelompok *server* MongoDB yang menyimpan sekumpulan data yang sama, menyediakan redudansi untuk meningkatkan ketersediaan data. *Automatic scaling* karena MongoDB didukung oleh *horizontal specification* sebagai bagian dari fungsi inti. *Sharding* otomatis ini mendistribusikan data ke sekelompok *node* dalam cluster. Namun set replika ini mengakibatkan "*eventual-consistency*" jadi data yang dimiliki bisa jadi saat yang sama antara user A dan B melihat *value* yang berbeda padahal dokumennya sama.

Format penulisan pada MongoDB adalah BSON(Binary JSON). Format BSON mirip dengan format JSON namun BSON memiliki beberapa kelebihan yaitu mampu menyimpan tipe data *date* dan tipe BinData [9]. MongoDB memiliki *query* dasar yang mirip dengan *query* SQL sehingga memudahkan kita yang terbiasa menggunakan SQL untuk mempelajari sintaknya. *Query* pada MongoDB dibagi menjadi dua kelompok besar yaitu *read operation* dan *write operation*.

2.4 Distributed *Database*

Distributed *database* adalah koleksi beberapa basis data yang secara logis saling didistribusikan melalui jaringan komputer [4]. Pada sistem ini memiliki beberapa jenis teknik pendistribusian data (distribution model) dan beberapa jenis teknik pembagian data (data specification).

Umumnya ada dua jenis distribusi data yaitu *replication* dan *sharding*. *Replication* meletakkan data yang sama dan mengkopinya kebanyak node. *Sharding* meletakkan data yang berbeda di node yang berbeda [3]. Kita dapat menggunakan salah satu ataupun kedua metode ini karena replikasi dan *sharding* adalah 2 teknik yang ortogonal. Jenis – jenis distribusi pada NoSQL antara lain: *single server*, *sharding*, *master-slave replication*, *peer-to-peer replication*, dan *combining sharding* dan *replication*. Akan dijelaskan 2 jenis yaitu *single server* dan *sharding*.

Single server adalah teknik penyimpanan data paling simple dimana tidak ada distribusi sama sekali. Basis data dijalankan pada mesin tunggal yang menangani keseluruhan proses baca dan tulis. Arsitektur ini mudah untuk dioperasikan [3]. *Sharding* merupakan teknik pendistribusian data dengan menempatkan data yang berbeda ditiap nodenya. pada NoSQL, *sharding* adalah *horizontal specification*. *Horizontal specification* adalah teknik pembagian data dengan berdasarkan recordnya. Misalkan data tentang project dan dipisahkan berdasarkan lokasinya. Pada MongoDB, *horizontal specification* disebut *sharding*.

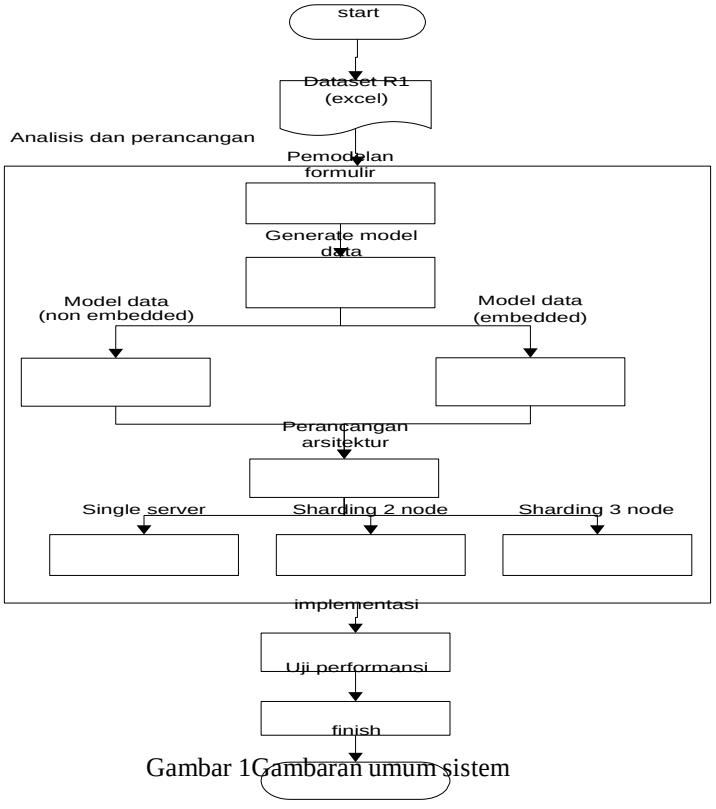
Pembagian data pada *sharding* dilakukan berdasarkan *shard key*. *Shard* key adalah *indexed field* ataupun *indexed compound field* yang ada pada setiap dokumen dikoleksi. MongoDB membagi nilai *shard* key kedalam *chunk* dan mendistribusikan *chunk* ke *shard*. Untuk membagi nilai *shard* key ke dalam *chunks*, MongoDB menggunakan *range based sharding* atau *hash based sharding*. Pada range based *sharding*, MongoDB membagi data set kedalam range yang ditentukan oleh nilai *shard* key. Dokumen dengan *key* yang saling berdekatan akan menjadi satu *chunks* dan satu *shard*. Pada *hash based partitioning*, MongoDB menghitung sebuah nilai *field hash* dan menggunakan *hash* ini untuk membentuk *chunk*. Dengan teknik ini, 2 dokumen dengan *shard* key yang dekat mungkin jadi bagian di chunk yang sama. Distribusi jenis ini lebih random untuk koleksi pada cluster.

2.5 Performansi basis data

Performansi basis data adalah kemampuan basis data dalam memproses suatu *query* . Untuk mengukur performansi basis data ada 2 parameter yang digunakan yaitu *response time* dan *throughput*. *Response time* adalah jumlah waktu yang dibutuhkan untuk menyelesaikan sebuah instruksi semenjak instruksi tersebut dijalankan [11]. *Throughput* adalah jumlah instruksi yang dapat diselesaikan dalam suatu interval waktu yang telah diberikan [11].

2.6 Perancangan

Pada bagian perancangan akan dibahas mengenai gambaran umum sistem yang diimplementasikan.



Gambar 1Gambaran umum sistem

Gambar 1 di atas merupakan diagram alir yang menunjukkan alur sistem ClikKB secara umum. Sistem ClikKB memiliki sumber data atau data set berupa file excel berformat R1. Format R1 terdapat pada lampiran A.

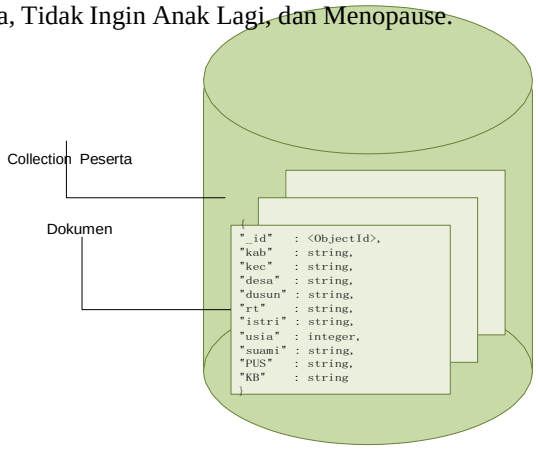
Pada bagian analisis dan perancangan, dilakukan analisis terhadap data yang termuat dalam file R1 untuk mengetahui model pengisian data pada dataset tersebut atau dapat disebut pemodelan formulir. Setelah diketahui model pengisiannya, dirancang struktur bsonnya. Langkah selanjutnya adalah merancang model data atau jika pada relasional disebut skema. Model data yang dirancang ada 2 alternatif yaitu model *embedded* dan non *embedded*. Setelah langkah perancangan model data, dilakukan perancangan arsitektur yang akan diterapkan. Arsitektur yang diterapkan ada 3 yaitu *single server*, distribusi *horizontal* dengan 2 *shard server*, dan distribusi *horizontal* dengan 3 *shard server*.

Setelah analisis dan perancangan, masuk keproses implementasi. Proses implementasi adalah proses menentukan bagaimana model disimpan berdasarkan perancangan yang telah dilakukan. Pertama menentukan arsitektur mana yang akan diimplementasi. Selanjutanya model data apa yang akan diimplementasi. Selanjutnya mengolah data set sesuai model data yang dipilih dan menyesuaikan data set dengan model pengisian yang telah ditentukan.

Setelah proses implementasi, selanjutnya dilakukan pengujian terhadap performansi sistem yang dibangun dengan parameter uji *response time* dan *throughput*.

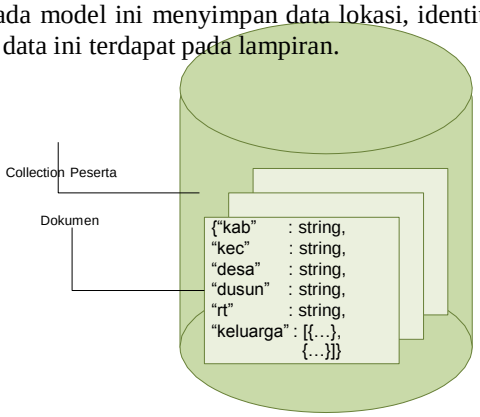
2.6.1 Perancangan basis data

Analisis data dilakukan terhadap file excel berformat R1. Adapun atribut yang ada pada format R1 adalah kabupaten, kecamatan, desan, dusun, dan rt bertipe data string. Atribut nomor bertipe data integer, nama istri bertipe data string, usia istri bertipe data integer, dan nama suami bertipe data string. Atribut status PUS(Pasangan Usia Subur) yaitu usia 15-49 tahun bertipe data string. Status PUS memiliki kemungkinan pengisian berupa PUS tidak beresiko, PUS beresiko yang memiliki beberapa alternatif isian seperti T1, T2 dan selanjutnya yang akan dijelaskan kemudian, dan Bukan PUS. Atribut PUS berfungsi untuk mencatat keikutsertaan KB memiliki sub atribut Peserta KB dengan tipe data string dan bukan peserta KB dengan tipe data string. Sub atribut peserta KB memiliki kemungkinan pengisian adalah IUD, MOW, MOP, IMP, SUNTIK, PIL, atau KDM. Sub atribut Bukan Peserta KB memiliki kemungkinan pengisian Hamil Beresiko, Hamil Tidak beresiko, Ingin Anak Segera, Ingin Anak Ditunda, Tidak Ingin Anak Lagi, dan Menopause.



Gambar 2 Model data non *embedded document*

Gambar 2 di atas menunjukkan bahwa *collection* peserta dapat menampung dokumen keluarga sesuai dengan data inputan. 1 dokumen pada model ini menyimpan data lokasi, identitas, status PUS dan PUS. Secara detail, bentuk dokumen pada model data ini terdapat pada lampiran.



Gambar 3 Model data *embedded document*

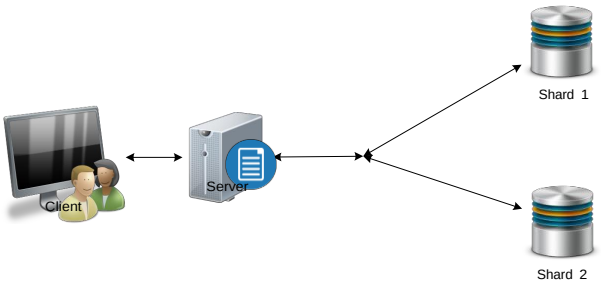
Gambar 3 menunjukkan bahwa pada model data ini, 1 dokumen akan menyimpan data lokasi yaitu data kabupaten, kecamatan, desa, dusun, dan rt serta mengembedd beberapa dokumen peserta yang berisi data identitas, status pus, dan keterangan KB. Dan dokumen – dokumen pada model ini disimpan dalam 1 *collection* yaitu *collection* peserta. Lebih detail tentang 1 *document* pada model ini ditunjukkan pada lampiran.

2.6.2 Perancangan arsitektur

Arsitektur yang dirancang ada 3 jenis arsitektur yaitu arsitektur *single server*, *sharding* 2 *node*, dan *sharding* 3 *node*.

Gambar 4 Arsitektur *single server*

Gambar 4 adalah arsitektur dimana sistem basis data yang digunakan menggunakan 1 *server* basis data. *Query* dikirimkan ke *server*, *server* mengambil atau menyimpan data ke *database*. Setelah itu *database* mengolah data dan dikembalikan ke *server*, dan *server* memberikan respon ke *user*/*query* .

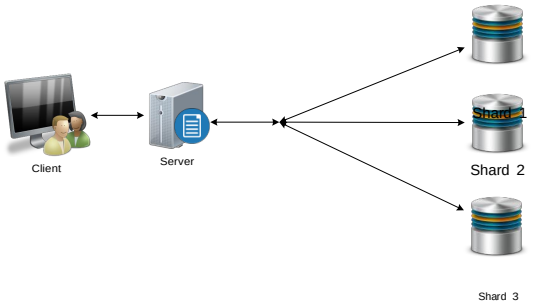


Gambar 5 Arsitektur *sharding* 2 node

Pada gambar 5 , dilakukan mekanisme *sharding* pada basis data. Arsitektur ini memiliki beberapa komponen yang berbeda dengan arsitektur *single server*. Komponen tersebut adalah *mongos server*, *config server*, dan *shard server*.

Mongos dapat dikatakan sebagai *router* atau *controller* yang mengatur atau mengarahkan *request* menuju *shard* mana dengan bantuan metadata dari *config server*. Jadi *request* dari *client* tidak langsung tertuju pada *shard server*. *Config server* berfungsi untuk menyimpan metadata dari *shard*. *Shard server* atau dapat disebut *mongod* adalah komputer yang menyimpan bagian dari koleksi data.

Sharding pada arsitektur ini membagi data menjadi 2 bagian dan didistribusikan pada 2 *node* yang berbeda. *Query* dikirimkan ke-*server*, *server* menentukan *request* diteruskan ke node mana dengan memanfaatkan config *server* yang berisi metadata data *shard*. Setelah ditentukan *shard* tujuan, *server* mengirim *request* ke *shard* tujuan.



Gambar 6 Arsitektur *sharding* 3 node

Pada arsitektur ketiga yang ditunjukkan oleh gambar 6, dilakukan mekanisme *sharding* pada 3 node yang berbeda. Arsitektur ini memiliki komponen dan cara kerja yang sama dengan arsitektur *sharding* 2 *node*. Yang berbeda adalah, jika pada 2 node masing-masing *shard server* menyimpan setengah data, maka pada arsitektur ini masing-masing *shard server* menyimpan sepertiga data.

3 Pembahasan

3.1 Pengujian model data

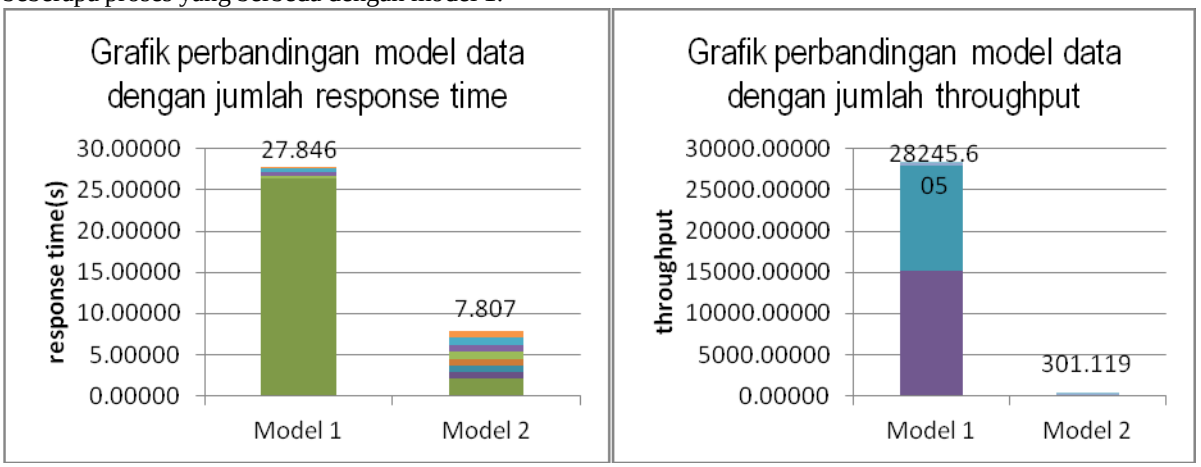
Pengujian model data akan menguji 2 model data yang telah dirancang sebelumnya pada perancangan basis data yaitu model non *embedded document*, dan model *embedded document*. Pengujian yang dilakukan akan diimplementasikan pada arsitektur *sharding* 2 *node*. Pengujian model data ini tidak dikaitkan dengan arsitektur sistem. Adapun skenario untuk pengujian pada model data adalah dengan menggunakan *query* – *query* yang akan dideskripsikan pada bagian selanjutnya. Masing – masing model akan diuji dengan *query* uji dan akan direkam hasil *response time* serta *throughtput*-nya.

Tabel 1 Hasil pengujian model data

Query	Model 1		Model 3	
	R(s)	T	R(s)	T
insert 1 excel	26.419	0.038	2.054	0.488
insert 1 *document*	0.007	149.134	0.766	1.315
view 1 desa	0.00007	15051.330	0.886	1.132
view 1 peserta	0.00008	12710.012	0.711	1.406
update	0.020	50.739	0.020	62.837
delete	0.004	270.833	0.005	229.157
subquery	0.212	4.727	0.896	1.125
aggregate *query*	0.567	1.857	0.864	1.158
subquery +aggregate	0.389	2.568	0.844	1.185
subquery +aggregate	0.229	4.366	0.760	1.316
total	27.846	28245.605	7.807	301.119

Tabel 1 di atas menunjukkan hasil pengujian pada 2 model data. Didapatkan hasil bahwa pada model 1, jika dijumlahkan untuk keseluruhan *query* , maka total *response time*nya adalah 27,846 detik. Sedangkan model 2 memiliki total *response time* 7.807 detik. Secara umum dapat dikatakan bahwa *response time* model yang mengimplementasikan *embedded document* lebih baik 356.70% jika dibanding dengan non *embedded* model.

Dari segi *throughput*, didapatkan hasil yang kurang sebanding dengan *response time*. Pada umumnya, *response time* berbanding terbalik dengan *throughput*. Pada pengujian ini, *throughput* model 1 adalah 28245.605 transaksi. Dan model 2 memiliki *throughput* 301.119 transaksi. Nilai yang tidak sebanding ini dikarenakan *query* view 1 desa dan view 1 peserta pada model 2 memiliki *response time* yang jauh lebih tinggi daripada model 1 karena perbedaan *query* yang digunakan untuk menghasilkan data yang sama. Pada model 2 membutuhkan beberapa proses yang berbeda dengan model 1.



Gambar 7 Grafik Pengujian model data

Gambar 7 bagian kiri di atas merupakan grafik hasil pengujian 2 model data terhadap jumlah *response time* dari keseluruhan *query* uji. Secara umum, diketahui bahwa model 1 memiliki jumlah *response time* yang tinggi yaitu 27.846 detik. Ini dikarenakan proses insert 1 excel pada model 1 memiliki *response time* sebesar 26.419 detik. Lamanya proses insert karena setiap 1 baris data, model ini akan menyimpan pada basis data sehingga adanya waktu akses yang berulang kali menyebabkan *response time*nya menjadi lama. Sedangkan waktu respon lebih rendah untuk keseluruhan *query* yang diujikan adalah model 2 dengan waktu 7.807 detik yaitu 356.70% lebih cepat dibanding model 1.

Gambar 7 bagian kanan merupakan grafik hasil pengujian 2 model data terhadap *throughput*. Diketahui bahwa *throughput* umumnya berbanding terbalik dengan *response time*. Jika *response time* kecil maka *throughput* besar dan sebaliknya. Namun pada kasus ini berbeda. *Throughput* total pada pengujian ini, tertinggi dimiliki oleh model 1 yaitu 28245.605 transaksi. Model 2 memiliki *throughput* 301.119 transaksi.

Tingginya *throughput* total yang dimiliki oleh model 1 dikarenakan *query* view 1 desa dan view 1 peserta memiliki response yang sangat kecil jika dibandingkan dengan model 2. Jadi saat dilakukan penjumlahan menghasilkan *throughput* yang tinggi. Sedangkan pada model 2 memiliki *throughput* terrendah karena pada pada saat pengujian, *response time* yang dimiliki cukup konstan tidak terlalu tinggi dan tidak terlalu rendah. Selain itu perbedaan *response time* model 1 dan 2 pada *query* view 1 desa dan view 1 peserta sangat tinggi. Hal ini yang menyebabkan *throughput* model 2 menjadi lebih kecil.

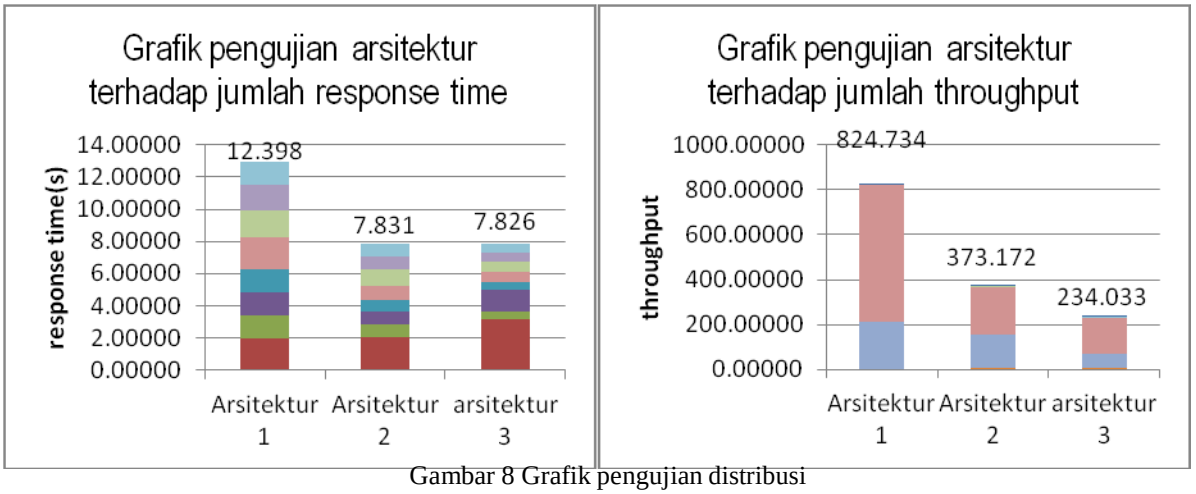
3.2 Pengujian Distribusi

Tabel 2 Hasil pengujian distribusi

Query	Arsitektur 1		Arsitektur 2		arsitektur 3	
	R(s)	T	R(s)	T	R(s)	T
insert 1 excel	1.959	0.511	2.054	0.487	3.132	0.319
insert 1 *document*	1.438	0.695	0.766	1.306	0.533	1.876
view 1 desa	1.448	0.690	0.813	1.232	1.303	0.769
view 1 peserta	1.424	0.702	0.711	1.406	0.503	1.986
update	0.005	209.206	0.007	152.207	0.017	62.955
delete	0.002	610.495	0.005	211.894	0.006	159.253
subquery	1.954	0.512	0.896	1.116	0.599	1.668
aggregate *query*	1.660	0.602	0.977	1.024	0.613	1.631
subquery +aggregate	1.638	0.610	0.844	1.185	0.582	1.719
subquery +aggregate	1.409	0.710	0.760	1.316	0.539	1.856
total	12.938	824.734	7.831	373.172	7.826	234.033

Tabel 2 di atas adalah hasil pengujian distribusi terhadap *response time* dan *throughput*. Didapatkan bahwa arsitektur pertama atau *single server* memiliki total *response time* paling tinggi yaitu 12.938 detik. Arsitektur kedua yaitu *sharding* menggunakan 2 *shard server* memiliki *response time* 7.831 detik. Arsitektur ketiga yaitu mengimplementasikan *sharding* menggunakan 3 *shard server* memiliki *response time* rata-rata 7.826 detik. Secara umum dapat dikatakan bahwa *response time* arsitektur yang mengimplementasikan distribusi data atau *sharding* lebih baik 165.20% jika dibanding arsitektur tanpa distribusi data.

Dari segi *throughput*, didapatkan hasil yang kurang sebanding dengan *response time*. Pada umumnya, *response time* berbanding terbalik dengan *throughput*. Pada pengujian ini, *throughput* tertinggi dimiliki oleh arsitektur 1 dengan *throughput* 824.734 transaksi. Model 2 dengan *throughput* 373.172 transaksi dan model 3 dengan 234.033 transaksi. Adanya nilai yang tidak sebanding ini dikarenakan *query delete* pada arsitektur 1 memiliki *response time* yang sangat rendah jika dibandingkan dengan arsitektur 2 dan 3.



Gambar 8 Grafik pengujian distribusi

Gambar 8 bagian kiri adalah grafik hasil pengujian arsitektur atau distribusi terhadap jumlah *response time* untuk keseluruhan *query* yang diujikan. Secara keseluruhuan, arsitektur 1 memiliki *response time* paling tinggi yaitu 12.937 detik. Selanjutnya arsitektur 2 7.831 detik dan arsitektur 3 7.826 detik. Jadi arsitektur dengan

sharding memiliki jumlah *response time* yang lebih baik dari pada arsitektur tanpa *sharding*. Karena rata-rata *query* yang berkaitan dengan proses *read* memiliki *response time* yang rendah pada arsitektur dengan *sharding*.

Gambar 8 bagian kanan adalah grafik hasil pengujian distribusi terhadap *throughput* ditinjau dari keseluruhan *query* . Arsitektur 1 memiliki jumlah *throughput* 824.734. arsitektur 2 memiliki *throughput* 373.172 transaksi dan arsitektur 3 memiliki *throughput* 234.033. Arsitektur tanpa *sharding* memiliki *throughput* lebih besar dibanding arsitektur dengan *sharding*. Hal ini karena *query delete* pada arsitektur 1 memiliki response yang sangat baik dibanding arsitektur 2 dan 3. Namun pada *query – query* lain khususnya *quer find*, secara umum arsitektur dengan *sharding* memiliki *throughput* yang lebih baik.

4 Kesimpulan dan Saran

4.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian, dapat disimpulkan bahwa:

1. Model data koleksi yang dapat diterapkan pada ClikKB ada 2 model yaitu *non embedded document* dan *embedded document*. Model *embedded document* dinilai lebih baik diterapkan untuk ClikKB karena *response time* yang lebih stabil.
2. Model data *embedded document* memiliki *response time* lebih cepat dari pada model yang lain. Namun untuk *throughput*, model non *embedded document* yang paling tinggi. Karena *query* view 1 desa dan 1 peserta pada model 1 memiliki *response* yang jauh lebih rendah dari model 2.
3. Secara umum, performansi sistem yang menerapkan distribusi atau *sharding* lebih baik jika dibanding pada sistem tanpa *sharding*. Sebagian besar *response time* sistem *sharding* jauh lebih cepat. Namun ada beberapa *query* yang lebih cepat jika diimplementasikan dengan sistem tanpa *sharding*.

4.2 Saran

Setelah dilakukannya implementasi, penulis memiliki beberapa saran antara lain:

1. Implementasi yang dilakukan hanya menggunakan *sharding*, tanpa replikasi. Untuk selanjutnya dapat dilakukan implementasi terhadap resplikasi maupun kombinasi *sharding* dan replikasi.
2. Dilakukan analisis dan implementasi model data yang lain.
3. Pengujian yang dilakukan hanya menggunakan parameter uji *response time* dan *throughput*. Untuk selanjutnya dapat dilakukan pengujian menggunakan parameter uji yang lain seperti *query* cost, error rate atau yang lainnya.

Daftar Pustaka:

[1] [1] "BKKBN Jawa Barat," 2014. [Online]. Available: jabar.bkkbn.go.id/ViewBerita.aspx?BeritaID=1718. [Accessed 24 October 2014].

[2] G. Brud, "*NoSQL Sysadmin*," vol. 36, October 2011.

[3] P. J. Sadalage and M. Fowler, *NoSQL Distilled*, New Jersey: Pearson Education, Inc., 2013.

[4] M. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, New York: Springer, 2011.

[5] A. Kadir, Konsep & Tuntunan Praktis Basis Data, Yogyakarta: Andi Yogyakarta, 1999.

[6] A. Silberschatz, H. F. Korth and S. Sudarshan, *Database System Concepts*, New York: McGraw-Hill Companies, Inc., 2011.

[7] O. Schmitt and T. A. Majchrzak, "*Using Document-based Database for Medical Information System in Unreliable Environments*," in International ISCRAM Conference, Vancouver, 2012.

[8] M. Stonebreaker, "*SQL Databases v. NoSQL Databases*," Vol. 53, April 2010.

[9] "BSON," creative commons, [Online]. Available: http://bsonspec.org/. [Accessed 28 October 2014].

[10] I. MongoDB, "*The MongoDB 2.6 Manual*," 2014. [Online]. Available: docs.mongodb.org/manual/. [Accessed 26 Oktober 2014].

[11] E. Fransisca S, Implementasi dan Analisis Performansi Heterogeneous Distributed *Database* (Relational *Database* dan *Document Oriented Database*) pada *Database* as a Service, Bandung: IT Telkom, 2013.

[12] M. Nicola and M. Jarke, "*Performance Modeling of Distributed and Replicated Databases*," Aachen.

Lampiran

Format R1 yang digunakan pada penelitian adalah sebagai berikut:

KABUPATEN	KECAMATAN	DESA/KELURAHAN	DUSUN/RW	RT	NO	IDENTITAS			STATUS PUS		PASANGAN USIA SUBUR (PUS)																
						ISTERI		NAMA SUAMI	PUS (USIA 15 - TIDAK BERES	BERESIK 0 *)	BUKAN PUS	PESERTA KB						BUKAN PESERTA KB									
						NAMA	USIA (TAHUN)					IUD	MOW	MOP	IMP	SUNTIK	PIL	KDM	HAMIL TIDAK BERES	HAMIL BERESI KO *)	INGIN ANAK SEGERI	INGIN ANAK DI	TIDAK INGIN ANAK	MENOPAUSE		
						2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21		
BOGOR	CIBINONG	CIRI MEKAR	001	01	1	SITIRAHMAWATI	41	M.A.HERMAN		T2							v										
					2	SANTIHANDAYANI	25	MUHTADIN	v								v										
					3	MAESAROH	37	TASIM		T2							v										
					4	METIA	34	ENCEP SUHENDAR	v								v										
					5	YUSNIDAR	23	HASBULAH	v								v										
					6	SITI	23	YUDI	v								v										
					7	UCUM	37	SARMAN		T2							v										
					8	YEYEN	31	UJANG	v								v										
					9	NGATIYEM	34	WAGIMIN	v									v										
					10	NURHAYATI	49	DADAN		T2													v				
					11	RISUYANTI	29	HERIYANA	v									v										
					12	SARYANTI	36	DARYANTO		T2												v					
					13	SURATMI	34	UNANG SUTAJI	v								v										
					14	IDAFARIDA	29	ARIF KAHMAN	v									30 M									
					15	SPURWANI	44	SAMSUDIN		T2								v									
					16	HUZAIBAH	22	M.SOLEH	v									v									
					17	WARKONAJ	45	IMANUDIN		T2							v										
					18	YANTI	30	MARDIANTO	v								v										
					19	ZUMAROH	34	MARJOKO	v								v										
					20	YUNI	32	BUDI	v								v										
					21	NIHAYA	47	-			v																

Struktur data model 1

Data lengkap dan menggunakan KB

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"   : string,
"desa"          : string,
"dusun"         : string,
"rt"            : string,
"istri"         : string,
"usia"          : integer,
"suami"         : string,
"PUS"   : string,
"KB"            : string
}
```

Struktur di atas menunjukkan 1 dokumen peserta yang memiliki atribut seperti di atas dimana tipe datanya string kecuali data usia. Struktur di atas digunakan untuk menyimpan data peserta yang menggunakan KB dan memiliki data yang lengkap.

Data lengkap tidak menggunakan KB

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"   : string,
"desa"          : string,
"dusun"         : string,
"rt"            : string,
"istri"         : string,
"usia"          : integer,
"suami"         : string,
"PUS"   : string,
"noKB"          : string
}
```

Struktur di atas digunakan untuk menyimpan 1 dokumen peserta yang tidak menggunakan KB dan memiliki data yang lengkap. Tipe data yang digunakan adalah string kecuali untuk data usia

Data lengkap tanpa keterangan KB atau nonKB

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"   : string,
"desa"          : string,
"dusun"         : string,
"rt"            : string,
"istri"         : string,
"usia"          : integer,
"suami"         : string,
"PUS"   : string
}
```

Struktur di atas digunakan untuk menyimpan 1 dokumen peserta yang tidak diketahui keikutsertaannya dalam berKB. Tipe data yang digunakan adalah string kecuali untuk data usia menggunakan integer.

Tidak ada data suami dan menggunakan KB

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"   : string,
"desa"          : string,
"dusun"         : string,
"rt"            : string,
"istri"         : string,
"usia"          : integer,
"PUS"   : string,
"KB"            : string
}
```

Struktur di atas digunakan untuk menyimpan data peserta yang tidak diketahui data suaminya atau data suami bernilai *null* dan menggunakan KB. Tipe data yang digunakan adalah string kecuali data usia yang bertipe data integer.

Tidak ada data suami dan tidak berKB

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"   : string,
"desa"          : string,
"dusun"         : string,
"rt"            : string,
"istri"         : string,
"usia"          : integer,
"PUS"   : string,
"noKB"          : string
}
```

Struktur di atas digunakan untuk menyimpan data peserta yang data suami bernilai *null* dan tidak menggunakan KB.

Tidak ada data suami dan tidak ada keterangan

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"   : string,
"desa"          : string,
"dusun"         : string,
"rt"            : string,
"istri"         : string,
"usia"          : integer,
"PUS"   : string
}
```

Struktur di atas digunakan untuk menyimpan data peserta dimana data suami bernilai *null* dan data penggunaan KB dan nonKBnya juga *null.*

Tidak ada data istri dan usia

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"  : string,
"desa"          : string,
"dusun"        : string,
"rt"            : string,
"suami"        : string,
"PUS"   : string
}
```

Struktur di atas digunakan untuk menyimpan data peserta dimana nama dan usia istri bernilai *null* dan keterangan KB atau tidak berKB juga *null.*

Tidak ada usia istri

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"  : string,
"desa"          : string,
"dusun"        : string,
"rt"            : string,
"istri"        : string,
"suami"       : string,
"PUS"   : string
}
```

Struktur di atas digunakan untuk menyimpan data peserta dimana usia istri bernilai *null* dan keterangan KB atau tidak berKB juga *null*.

Tidak ada usia istri dan suami

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"  : string,
"desa"          : string,
"dusun"        : string,
"rt"            : string,
"istri"        : string,
"PUS"   : string
}
```

Struktur di atas digunakan untuk menyimpan data peserta dimana usia, suami, dan keterangan keanggotaan KB bernilai null.

Tidak ada nama istri

```
{
"_id"   : <objectid>,
"kab"   : string,
"kec"  : string,
"desa"          : string,
"dusun"        : string,
"rt"            : string,
"usia"          : integer,
"suami"        : string,
"PUS"   : string
}
```

Struktur di atas digunakan untuk menyimpan data dimana data istri dan keterangan keanggotaan KB bernilai null.

Struktur data model 2

```
{
"_id"   : <objectId>,
"kab"   : string,
"kec"  : string,
"desa"          : string,
"dusun"        : string,
"rt"            : string,
"keluarga" : [
    {"istri": string,
    "usia"     : integer,
    "suami": string,
    "PUS"      : string,
    "KB"       : string  },
        {
    "istri"   : string,
    "usia"     : integer,
    "suami"   : string,
    "PUS"      : string,
    "noKB"     : string  },
        {
    "istri"   : string,
    "usia"     : integer,
    "suami"   : string,
    "PUS"      : string  },
        {
    "istri"   : string,
    "usia"     : integer,
    "PUS"      : string,
    "KB"            : string  },
                { ... }
                    ]
}
```

Struktur di atas digunakan pada model kedua yaitu model *embedded document*. Model ini menyimpan data lokasi dan meng-embedd beberapa dokumen peserta. Adapun model data dari data peserta seperti pada model sebelumnya yaitu 10 jenis model.