Transformasi Skema Basis Data Relasional Menjadi Model Data Berorientasi Dokumen pada MongoDB

I Gede Winaya*[1], Ahmad Ashari[2]
[1]Program Studi S2 Ilmu Komputer, FMIPA UGM, Yogyakarta
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: ***[1]igedewinaya@gmail.com**, [2]ashari@ugm.ac.id

Abstrak

MongoDB merupakan basis data yang menggunakan model penyimpanan data berorientasi dokumen. Pada kenyataanya, untuk beralih dari basis data relasional ke suatu basis data NoSQL seperti MongoDB bukanlah perkara yang mudah terlebih lagi jika data yang dimiliki sangat kompleks. Berdasarkan dokumentasi yang telah dilakukan oleh beberapa perusahaan global terkait penggunaan MongoDB, dapat disimpulkan bahwa proses migrasi dari RDBMS ke MongoDB memerlukan waktu yang cukup lama. Salah satu proses yang memakan waktu cukup banyak adalah transformasi skema yang terdapat pada basis data relasional menjadi model data berorientasi dokumen pada MongoDB.

Penelitian ini membahas tentang pengembangan sistem transformasi skema basis data relasional menjadi model data berorientasi dokumen pada MongoDB. Proses transformasi dilakukan dengan memanfaatkan struktur dan relasi antar tabel yang ada di dalam skema sebagai parameter utama pada algoritma pembentukan model. Dalam proses pembentukan model dokumen perlu dilakukan penyesuaian terhadap spesifikasi dokumen yang ditetapkan oleh MongoDB agar model dokumen yang terbentuk dapat diimplementasikan pada MongoDB.

Model dokumen yang terbentuk dari proses transformasi ini dapat berupa dokumen tunggal, embedded document, referenced document ataupun kombinasi ketiganya. Model dokumen yang terbentuk bergantung dari tipe, aturan, dan nilai kardinalitas relasi antar tabel yang ada di dalam skema basis data relasional.

Kata kunci*— MongoDB, basis data relasional, model dokumen, transformasi.*

Abstract

MongoDB is a database that uses document-oriented data storage models. In fact, to migrate from a relational database to NoSQL databases such as MongoDB is not an easy matter especially if the data are extremely complex. Based on the documentation that has been done by several global companies related to the use of MongoDB, it can be concluded that the process of migration from RDBMS to MongoDB require quite a long time. One process that takes quite a lot is transformation of relational database schema into a document-oriented data model on MongoDB.

This research discusses the development transformation system of relational database schema to the document oriented data model in MongoDB. The process of transformation is done by utilizing the structure and relationships between tables in the scheme as the main parameters of the modeling algorithm. In the process of the modeling documents, it necessary to adjustments the specifications of MongoDB document that formed document model can be implemented in MongoDB.

Document models are formed from transformation process can be a single document, embedded document, referenced document or combination of these. Document models are formed depending on the type, rules, and the value of the relationships cardinality between tables in the relational database schema.

Keywords*— MongoDB, relational database, document model, transformation.*

Received May 19th,2015; Revised January 1st, 2016; Accepted January 20th, 2016

1. PENDAHULUAN

Salah satu basis data NoSQL yang populer saat ini adalah MongoDB. MongoDB merupakan basis data yang menggunakan model penyimpanan data berorientasi dokumen. Pada kenyataanya, untuk beralih dari basis data relasional ke suatu basis data NoSQL seperti MongoDB bukanlah perkara yang mudah terlebih lagi jika data yang dimiliki sangat kompleks. Berdasarkan dokumentasi yang telah dilakukan oleh beberapa perusahaan global terkait penggunaan MongoDB, dapat disimpulkan bahwa proses migrasi dari RDBMS ke MongoDB memerlukan waktu yang cukup lama. Proses yang memakan waktu cukup banyak adalah analisis kebutuhan dan transformasi skema yang terdapat pada basis data relasional menjadi model data berorientasi dokumen pada MongoDB [1]. Dari dokumentasi yang dilakukan oleh Apollo Group tentang proses migrasi dari penggunaan RDBMS Oracle ke MongoDB, tim proyek menghabiskan 25% waktu yang ada untuk melakukan pemodelan data [2]. Proses pemodelan skema ini merupakan inti dari proses migrasi yang dilakukan. Para *developer* atau *database engineer* memodelkan skema pada basis data relasional ke dalam model dokumen yang diinginkan pada MongoDB sesuai dengan kebutuhan sistem yang ada secara manual. Semakin kompleks skema yang ada pada basis data relasional, maka dibutuhkan waktu yang semakin lama dan usaha yang semakin besar untuk menyelesaikan proses migrasi basis data.

Saat ini beberapa *tool* ETL (*Extract Transform Load*) seperti Informatica, Pentahu, dan Talend telah memiliki fitur untuk melakukan migrasi dari basis data relasional ke MongoDB, akan tetapi fitur-fitur tersebut hanya mengakomodasi proses perpindahan data. Data yang ada pada basis data relasional akan diekstraksi sedemikian rupa sesuai dengan format yang ada agar dapat diubah menjadi bentuk dokumen pada MongoDB. Walaupun telah melalui tahap ekstraksi agar data siap dipindahkan, namun kekurangan yang ada pada fitur-fitur ini adalah model dokumen yang ada pada MongoDB harus dibuat terlebih dahulu oleh para *developer* atau *database engineer* secara manual sebelum data siap dipindahkan. Berdasarkan permasalahan ini, maka pembuatan sistem transformasi skema basis data relasional menjadi model data berorientasi dokumen pada MongoDB diharapkan dapat mengakomodasi proses migrasi basis data yang dilakukan. Sistem ini diharapkan pula mampu memberikan solusi untuk pembentukan model dokumen pada MongoDB sesuai dengan skema basis data relasional yang digunakan. Tujuan lain dikembangkannya sistem ini adalah meningkatkan efisiensi sumber daya dan waktu yang dibutuhkan pada proses migrasi basis data sehingga tidak menganggu proses bisnis yang ada pada suatu institusi atau perusahaan.

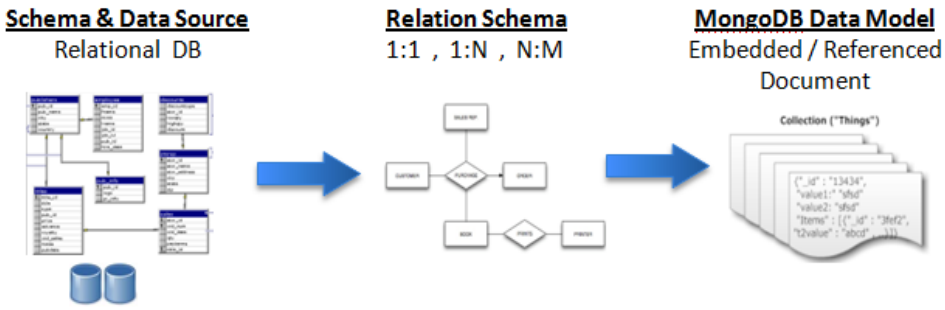
Sebelumnya telah dilakukan pula beberapa penelitian yang membahas tentang transformasi basis data relasional [3,4] dan model data pada MongoDB [5,6,7,8]. Berbeda dengan penelitian sebelumnya, penelitian ini akan membahas tentang pengembangan sistem yang dapat melakukan transformasi skema basis data relasional menjadi model dokumen MongoDB. Sistem transformasi yang dibangun harus menjamin setiap data yang terdapat pada basis data relasional dapat direpresentasikan secara utuh dan konsisten pada model dokumen MongoDB yang dihasilkan. Penelitian yang dilakukan berfokus pada hasil transformasi skema basis data, sehingga untuk masalah optimasi waktu proses transformasi dan sumber daya yang digunakan tidak diperhitungkan. *Relational Database Management System* (RDBMS) yang digunakan pada penelitian ini sebagai implementasi basis data relasional adalah MySQL.

2. METODE PENELITIAN

Pengembangan sistem transformasi dimulai dari tahap perancangan yang diikuti dengan pengimplementasian sistem dalam bentuk aplikasi berbasis web dan diakhiri oleh tahapan pengujian sistem. Perancangan terhadap sistem yang dibangun meliputi arsitektur sistem, rancangan antarmuka, serta proses transformasi skema basis data relasional menjadi model dokumen pada MongoDB yang di dalamnya mencakup pembentukan algoritma transformasi dan penanganan spesifikasi dokumen.

2.1 Proses Transformasi

Sistem yang dibangun secara garis besar akan melakukan beberapa tahapan proses. Proses awal yang dilakukan adalah membaca suatu skema basis data relasional yang dimasukan oleh pengguna. Skema yang dimaksud pada penelitian ini adalah struktur dan relasi antar tabel yang ada di dalam basis data. Skema yang dimasukan tadi akan dianalisis oleh sistem untuk mendapatkan parameter-parameter yang diperlukan pada proses transformasi. Selanjutnya pada proses transformasi, algoritma pembentukan model akan menggunakan parameter tadi untuk mendapatkan model dokumen yang nantinya dapat diimplementasikan pada MongoDB. Secara sederhana, alur proses transformasi ditunjukan oleh Gambar 1.



Gambar 1 Alur proses transformasi

Untuk melakukan proses transformasi dari suatu skema basis data yang ada pada MySQL, sistem yang dibangun akan memanfaatkan informasi yang terdapat pada "*information_schema*" untuk menganalisis *metadata* yang terkait dengan struktur tabel, relasi, dan kardinalitas data. *Information_schema* merupakan basis data bawaan DBMS yang digunakan untuk menyimpan *metadata* dari seluruh skema basis data yang didefinisikan pada MySQL. *Metadata* pada *information_schema* inilah yang nantinya akan menjadi parameter utama dalam pengimplementasian algoritma transformasi.

2.1.1 Algoritma pembentukan model dokumen

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

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

Aturan relasi A : pembentukan model untuk tabel yang tidak berelasi dengan tabel lain

Jika suatu tabel tidak berelasi dengan tabel lainnya di dalam skema basis data, maka semua *tuple* pada tabel tersebut akan menjadi dokumen tunggal pada MongoDB dan tergabung dalam *collection* yang sama.

Aturan relasi B : pembentukan model untuk tabel yang hanya berelasi dengan sebuah tabel lain

Jika suatu tabel (misalnya tabel A) hanya berelasi dengan satu buah tabel lain (misalnya tabel B), maka terdapat beberapa aturan yang akan diberlakukan dalam proses pembentukan model dokumen:

1. Jika data pada tabel B mengacu ke data yang ada pada tabel A dengan nilai kardinalitas relasi 1:1 dan relasi antar kedua tabel tersebut bertipe *full reference*, maka model dokumen yang akan terbentuk adalah *embedded document* dimana *tuple* pada tabel A akan menjadi bagian dari *tuple* yang ada pada tabel B.
2. Jika data pada tabel B mengacu ke data yang ada pada tabel A dengan nilai kardinalitas relasi N:1 dan relasi antar kedua tabel tersebut bertipe *full reference*, maka model dokumen yang akan terbentuk adalah *embedded document* dimana *tuple* pada tabel A akan menjadi bagian dari *tuple* yang ada pada tabel B. Selain itu pula, semua *tuple* pada tabel A juga akan menjadi dokumen tunggal pada MongoDB.
3. Jika data pada tabel A mengacu ke data yang ada pada tabel B dan relasi antar kedua tabel tersebut bertipe *full reference*, maka model dokumen yang akan terbentuk adalah *embedded document* dimana *tuple* pada tabel A akan menjadi bagian atau sub-data dari *tuple* yang ada pada tabel B.
4. Jika relasi antar tabel A dan tabel B bertipe *partial reference*, maka model dokumen yang akan terbentuk adalah *embedded document* dimana *tuple* pada tabel A akan menjadi bagian dari *tuple* yang ada pada tabel B. Selain itu pula, semua *tuple* pada tabel A juga akan menjadi dokumen tunggal pada MongoDB.

Aturan relasi C : pembentukan model untuk tabel yang berelasi dengan beberapa tabel lain

Jika suatu tabel (misalnya tabel A) memiliki relasi dengan beberapa tabel lain (misalnya tabel X), maka untuk setiap tabel (tabel $X_{[i]}$) yang berelasi dengan tabel A akan dikenakan aturan sebagai berikut:

1. Jika tabel $X_{[i]}$ hanya berelasi dengan tabel A, maka model dokumen yang akan terbentuk mengikuti aturan pembentukan model pada anturan B.
2. Jika tabel $X_{[i]}$ tidak hanya berelasi dengan tabel A melainkan juga dengan tabel lainnya dan data pada tabel A mengacu pada data di tabel tersebut, maka model dokumen yang akan terbentuk adalah *referenced document* dimana setiap *tuple* pada tabel A akan mengacu data-data yang ada pada tabel $X_{[i]}$. Selain itu, semua *tuple* pada tabel $X_{[i]}$ juga akan menjadi dokumen tunggal pada MongoDB.

2.1.2 Penanganan spesifikasi dokumen MongoDB

Selain relasi antar tabel yang ada pada skema basis data relasional, faktor lain yang berbengaruh dalam proses pembentukan model dokumen pada MongoDB adalah spesifikasi dari dokumen itu sendiri. Terdapat beberapa spesifikasi dasar yang terdapat pada dokumen MongoDB sehingga diperlukan beberapa penyesuaian dalam proses pembentukan model dokumen yang dibuat. Spesifikasi yang dimaksud yaitu limitasi ukuran dokumen, *indentifier* untuk sebuah dokumen, data yang bersifat inkremental, dan translasi tipe data.

Limitasi ukuran dokumen

Ukuran data maksimal untuk sebuah dokumen pada MongoDB adalah 16 MB (*Megabyte*). Dengan adanya limitasi ini, maka tidak dimungkinakan untuk menyimpan data yang berukuran lebih dari 16 MB di dalam sebuah dokumen pada MongoDB. Limitasi ukuran data ini mengacu pada spesifikasi yang telah ditetapkan oleh MongoDB dengan mempertimbangkan aspek penggunaan memori dan efisiensi *bandwidth* dalam proses transmisi data [9]. Untuk mengakomodasi limitasi ini, MongoDB telah menyediakan mekanisme untuk penyimpanan data berukuran besar yang disebut dengan GridFS. GridFS merupakan mekanisme penyimpanan data dengan cara memecah data yang besar menjadi beberapa data berukuran kecil

(≤ 16 MB) yang disebut dengan *chunk*. Dengan menggunakan mekanisme ini maka limitasi ukuran dokumen yang ada dapat dipenuhi.

MySQL sendiri memiliki beberapa tipe data yang digunakan untuk mengakomodasi penyimpanan data berukuran besar seperti tipe data *text* dan *blob* yang dapat menangani penyimpanan data kurang lebih hingga 4 GB (*Gigabyte*). Melihat dari spesifikasi tipe data berjenis *text* dan *blob*, maka mekanisme GridFS di MongoDB akan diterapkan pada model dokumen yang terbentuk dari tabel-tabel yang menggunakan *field* dengan tipe data *mediumtext*, *longtext*, *mediumblob*, dan *longblob*.

Identifier dokumen

Setiap dokumen pada MongoDB harus memiliki *identifier* yang digunakan sebagai pembeda antara satu dokumen dengan dokumen yang lainnya. Pada tahap pembentukan model dokumen, *primary key* pada sebuah tabel akan ditransformasikan menjadi *identifier* (*key* "_id") pada model dokumen yang terbentuk. Ini merupakan kasus ideal dan hanya berlaku untuk tabel-tabel yang memiliki sebuah *primary key*. Permasalahan mulai muncul ketika terdapat suatu tabel di dalam skema basis data yang tidak memiliki *primary key* atau memiliki lebih dari satu *primary key*. Pada kasus ini, maka model dokumen yang dibentuk tidak dapat mengikuti struktur *primary key* yang didefinisikan pada tabel tersebut karena dokumen pada MongoDB juga memiliki karakteristik tersendiri. Karakteristik ini mengharuskan sebuah dokumen hanya boleh memiliki sebuah *identifier* (*key* "_id"). Untuk menangani kasus ini, maka pada model dokumen yang dibentuk akan didefinisikan sebuah *identifier* baru dengan memanfaatkan fungsi ObjectId() untuk mendapatkan nilai yang unik pada *key* "_id". Dengan mekanisme ini, maka tidak akan ada dokumen yang memiliki beberapa *identifier* dan memastikan setiap dokumen hanya memiliki sebuah *identifier*.

Data yang bersifat inkremental

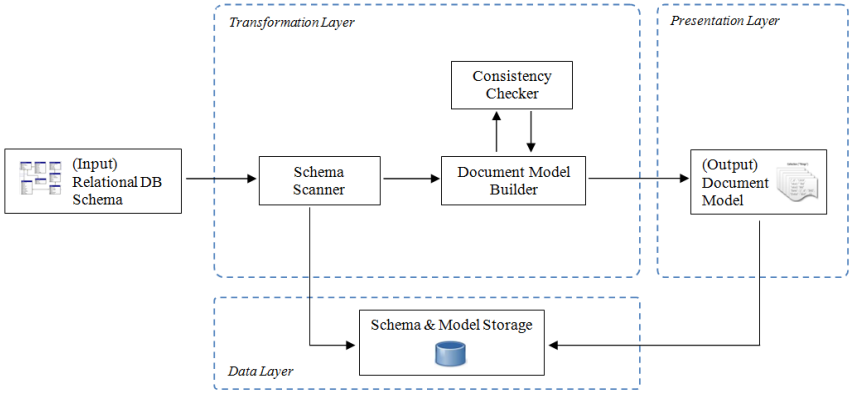
MongoDB tidak memiliki fitur untuk menangani data-data yang bersifat inkremental. Oleh karena itu pendefinisian tentang data yang bersifat inkremental harus dilakukan secara manual. Pendefinisian ini dilakukan dengan membangun *collection* tersendiri yang berisikan model dokumen yang dapat mengatur data-data bersifat inkremental.

Translasi tipe data

MySQL memiliki beragam tipe data yang digunakan dalam mekanisme pendefinisian data. Secara unum, tipe data yang ada pada MySQL dapat dikelompokan menjadi tipe data *numeric*, *string*, serta *date and time*. Untuk menjaga konsistensi data dari model yang terbentuk, tipe data yang didefinisikan pada MySQL harus disesuaikan dengan tipe data yang ada pada MongoDB.

2.2 Arsitektur Sistem

Terdapat tiga komponen utama pada mekanisme transformasi yang dilakukan pada sistem ini. Ketiga komponen tersebut adalah *schema scanner*, *document model builder*, dan *consistency checker*. *Schema scanner* berfungsi untuk membaca skema basis data relasional yang dimasukan oleh pengguna. Dari proses pembacaan ini, maka dapat diketahui data, tabel, dan relasi yang ada di dalam skema. *Document model builder* bertugas untuk membangun model dokumen dari skema yang dimasukan dengan memanfaatkan algoritma pembentukan model dokumen dengan menggunakan data, tabel, dan relasi yang ada sebagai parameternya. Dalam pembentukan dokumen, diperlukan pula modul *consistency checker* yang berfungsi untuk memastikan konsistensi data pada model dokumen yang dibentuk. Integrasi antar ketiga komponen ini dapat dilihat dari arsitektur sistem yang digambarkan pada Gambar 2.



Gambar 2 Arsitektur sistem transformasi

Sistem transformasi yang dibangun terdiri dari tiga bagian utama yaitu *transformation layer*, *data layer*, dan *presentation layer*. Pada bagian *transformation layer* terdapat komponen-komponen pembentuk mekanisme transformasi skema basis data relasional. Inilah bagian inti dari sistem yang dibangun. *Data layer* berisikan media penyimpanan dari data-data yang ada pada skema masukan pengguna. Selain data-data dari skema yang dimasukan, media penyimpanan ini digunakan pula untuk menyimpan model dokumen hasil dari keluaran sistem. Model dokumen yang terbentuk dari hasil transformasi tidak hanya disimpan pada media penyimpanan namun ditampilkan pula kepada pengguna seperti yang terlihat pada *presentation layer*.

2.3 Pengujian Sistem

Sistem transformasi yang dibangun harus menjamin setiap data yang terdapat pada basis data relasional dapat direpresentasikan secara utuh dan konsisten pada model dokumen MongoDB yang dihasilkan. Model dokumen yang terbentuk akan diuji integritasnya sesuai dengan skema yang dimodelkan. Pengujian ini akan dilakukan terhadap struktur dan model data pada setiap tipe model dokumen yang dihasilkan. Melalui pengujian ini akan diketahui kemampuan sistem untuk menangani isu integritas data pada model dokumen hasil proses transformasi.

3. HASIL DAN PEMBAHASAN

3.1 Model Dokumen

Model dokumen yang terbentuk dari proses transformasi ini dapat berupa dokumen tunggal, embedded document, referenced document ataupun kombinasi ketiganya. Model dokumen yang terbentuk bergantung dari tipe, aturan, dan nilai kardinalitas relasi antar tabel yang ada di dalam skema basis data relasional.

3.1.1 Dokumen tunggal

Model dokumen yang terbentuk dari *tuple* data pada tabel yang tidak berelasi dengan tabel lain adalah dokumen tunggal. Setiap *field* pada tabel asal akan menjadi *key* pada model dokumen yang terbentuk. Relasi tabel dan model dokumen yang terbentuk dari kasus ini dijabarkan sebagai berikut :

1. Ilustrasi relasi tabel : 

2. Model dokumen : dokumen tunggal
3. Struktur model dokumen :

```
{
        "Field_A1" : <value>,
        "Field_A2" : <value>,
```

```
        "Field_A3" : <value>,
         …
        "Field_An" : <value>
}
```

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

```
SELECT [Field_A1 … Field_An] FROM tabel_A
```

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

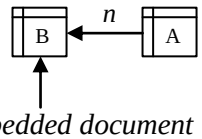
```
db.[tabel_A].find()
```

Dari pengujian yang dilakukan, kedua *query* ini menampilkan data yang sama. Hal ini membuktikan bahwa model dokumen yang dihasilkan untuk tabel yang tidak berelasi dapat menjaga konsistensi dan keutuhan data.

3.1.2 Embedded document

Model dokumen yang terbentuk dari *tuple* data pada tabel yang berelasi dengan tabel lain adalah dapat berupa dokumen tunggal, *embedded document*, dan *referenced document*. Terdapat dua macam karakteristik dari embedded document. Ada *embedded document* yang memiliki sub-dokumen dan ada pula yang tidak. Pada *embedded document* yang memiliki sub-dokumen, *tuple* data yang menjadi bagian pada dokumen induk akan memiliki *parent key*. Sedangan pada *embedded document* yang tidak memiliki sub-dokumen, *field* dari *tuple* data yang menjadi bagian pada dokumen induk tidak akan memiliki *parent key*. Relasi tabel dan model dokumen yang terbentuk untuk *embedded document* yang memilki sub-dokumen dijabarkan sebagai berikut :

1. Ilustrasi relasi tabel :



2. Model dokumen : *embedded document*
3. Kardinalitas : *many*
4. Tipe relasi : *full reference*
5. *Foreign key* : Field_A2
6. *Referenced field* : Field_B1
7. Struktur model dokumen :

```
{
        "Field_B1" : <value>,
        "tabel_A"  : [
                        {
                            "Field_A1" : <value>,
                            "Field_A3" : <value>,
                            …
                            "Field_An" : <value>
                        }
                        …
                     ],
        "Field_B3" : <value>,
         …
        "Field_Bn" : <value>
}
```

Pada *embedded document* yang terbentuk, "Field_A2" tidak menjadi *key* pada model dokumen karena sebenarnya data-data pada "Field_A2" sama dengan data pada "Field_B1".

Penghilangan *key* ini ditujukan untuk menghidari redundansi data yang terjadi pada model dokumen. Untuk mencocokan data yang ada di tabel asal dengan data yang ada pada dokumen hasil implementasi model dilakukan dengan membandingkan hasil pengaksesan data. Pengaksesan data yang ada di tabel asal dapat dilakukan dengan menjalankan *query* pengaksesan data dalam format sebagai berikut:

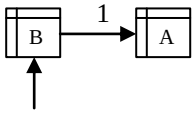
```
SELECT [Field_B1 … Field_Bn, Field_A1 … Field_An]
FROM tabel_B
JOIN tabel_A ON tabel_B.Field_B1 = tabel_A.field_A2
```

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

```
db.[tabel_B].find()
```

Relasi tabel dan model dokumen yang terbentuk untuk *embedded document* yang tidak memilki sub-dokumen dijabarkan sebagai berikut:

1. Ilustrasi relasi tabel :



2. Model dokumen : *embedded document*
3. Kardinalitas : *one*
4. Tipe relasi : *full reference*
5. *Foreign key* : Field_B2
6. *Referenced field* : Field_A1
7. Struktur model dokumen :

```
{
        "Field_B1" : <value>,
        "Field_B2" : <value>,
        "Field_A2" : <value>,
        "Field_A3" : <value>,
         …
        "Field_An" : <value>,
        "Field_B3" : <value>,
         …
        "Field_Bn" : <value>
}
```

Pada *embedded document* yang terbentuk, "Field_A1" tidak menjadi *key* pada model dokumen karena sebenarnya data-data pada "Field_A1" sama dengan data pada "Field_B2". Penghilangan *key* ini ditujukan untuk menghidari redundansi data yang terjadi pada model dokumen. Untuk mencocokan data yang ada di tabel asal dengan data yang ada pada dokumen hasil implementasi model dilakukan dengan membandingkan hasil pengaksesan data. Pengaksesan data yang ada di tabel asal dapat dilakukan dengan menjalankan *query* pengaksesan data dalam format sebagai berikut:

```
SELECT [Field_B1 … Field_Bn, Field_A1 … Field_An]
FROM tabel_B
JOIN tabel_A ON tabel_B.Field_B2 = tabel_A.field_A1
```

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

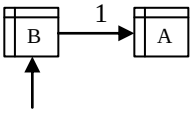
```
db.[tabel_B].find()
```

Dari pengujian yang dilakukan, *query* pengaksesan data untuk model *embedded document* ini menampilkan data yang sama. Hal ini membuktikan bahwa model *embedded document* yang dihasilkan dapat menjaga konsistensi dan keutuhan data.

3.1.3 Referenced document

Relasi tabel dan model dokumen yang terbentuk untuk *embedded document* yang tidak memilki sub-dokumen dijabarkan sebagai berikut:

1. Ilustrasi relasi tabel :



2. Model dokumen : *referenced document*
3. Kardinalitas : *one*
4. Tipe relasi : *partial reference*
5. *Foreign key* : Field_B2
6. *Referenced field* : Field_A1
7. Struktur model dokumen :

```
{
        "Field_B1" : <value>,
        "Field_B2" : { $ref : "dokumen_collection_A" },
         …
        "Field_Bn" : <value>
}
```

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

```
SELECT [Field_B1 … Field_Bn, Field_A1 … Field_An]
FROM tabel_B
JOIN tabel_A ON tabel_B.Field_B2 = tabel_A.field_A1
WHERE tabel_B.Field_B1 = '<value>'
LIMIT 1
```

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

```
var tb = db.tabel_B.findOne({"Field_B1":"<value>"})
var dbRef = anggota.Field_B2
db[dbRef.$ref].findOne({"_id":(dbRef.$id)})
```

Dari pengujian yang dilakukan, *query* pengaksesan data untuk model *referenced document* ini menampilkan data yang sama. Berbeda dengan model *embedded document*, model *referenced document* tidak dapat dapat menangani aturan relasi *restrict* dan *cascade* pada proses *update* dan *delete* data. Pada model *embedded document*, *tuple* data dari setiap tabel yang berelasi digabungkan sedemikian rupa menjadi sebuah dokumen agar proses *update* dan *delete* terhadap suatu data otomatis menjadi kesatuan yang utuh. Ini menyebabkan integritas data yang ada di dalam basis data dapat terjaga. Pada model *referenced document* dimana data yang saling berkaitan masih terpisah di dokumen berbeda, hal ini menyebabkan proses *update* dan *delete* data tidak menjadi kesatuan yang utuh sehingga integritas data tidak dapat dipertahankan. Namun untuk proses pembacaan atau pengaksesan data, model *referenced document* masih dapat menjaga konsistensinya.

3.2 Model Data

Setiap tipe data pada MySQL harus dimodelkan sesuai dengan tipe data yang ada pada MongoDB. Pemodelan ini dilakukan dengan mentranslasikan setiap tipe data MySQL menjadi tipe data pada MongoDB. Mekanisme translasi ini harus memperhatikan spesifikasi masing-masing tipe data agar pemodelan yang dilakukan dapat mempertahankan konsistensi data. Daftar translasi tipe data pada sistem transformasi yang dibangun ditunjukan pada Tabel 1.

Tabel 1 Daftar translasi tipe data

Kategori	Tipe Data MySQL	Tipe Data MongoDB
Numeric	bigint	NumberLong()
	bit	string
	decimal	string
	double	numeric
	float	numeric
	int	NumberLong()
	mediumint	NumberInt()
	smallint	NumberInt()
	tinyint	NumberInt()
String	binary	BinData()
	blob	BinData()
	char	string
	enum	string
	longblob	BinData()
	longtext	string
	mediumblob	BinData()
	mediumtext	string
	set	string
	text	string
	tinyblob	BinData()
	tinytext	string
	varbinary	BinData()
	varchar	string
Date and Time	date	string
	datetime	string
	time	string
	timestamp	ISODate()
	year	string

Pada mekanisme translasi ini terdapat beberapa keterbatasan pada tipe data *numeric* yang dimiliki oleh MongoDB. Keterbatasan ini terkait dengan besaran nilai yang dapat ditangani dari tipe data *numeric* yang dimiliki MongoDB. Salah satu tipe data yang tidak dapat ditangani secara maksimal pada MongoDB adalah tipe data "bigint". Pada MySQL, tipe data "bigint" digunakan untuk mengakomodasi nilai bilangan bulat yang besar. Tipe data "bigint" mampu menangani nilai dari -9223372036854775808 sampai 9223372036854775807 untuk bilangan bulat bertanda (*signed integer*) dan 0 sampai 18446744073709551615 untuk bilangan bulat tidak bertanda (*unsigned integer*). Pada kasus *signed integer*, tipe data "NumberLong" yang merupakan tipe data *numeric* terbesar pada MongoDB mampu menyamai rentang nilai yang dimiliki oleh tipe data "bigint". Namun pada kasus *unsigned integer*, tipe data

"NumberLong" tidak dapat menangani nilai hingga 18446744073709551615. Hal ini terjadi pula untuk tipe data "decimal" pada MySQL yang memiliki rentang nilai -99999999999999999999999999999999.99999999999999999999999999999999 hingga 99999999999999999999999999999999.99999999999999999999999999999999. Karena tipe data "decimal" memiliki rentang nilai yang sangat besar, maka tipe data ini ditranslasikan menjadi tipe data "string" pada MongoDB sehingga perlu dilakukan penyesuaian untuk melakukan operasi perhitungan.

4. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, maka dapat diperoleh beberapa kesimpulan, yaitu:

1. Transformasi skema basis data relasional menjadi model data berorientasi dokumen pada MonggoDB dilakukan dengan memanfaatkan struktur dan relasi antar tabel yang ada di dalam skema sebagai parameter utama pada algoritma pembentukan model. Dalam proses pembentukan model dokumen perlu dilakukan penyesuaian terhadap spesifikasi dokumen yang ditetapkan oleh MongoDB seperti pembentukan *identifier* (_id), limitasi ukuran dokumen, penanganan data yang bersifat inkremental, serta translasi tipe data agar model dokumen yang terbentuk dapat diimplementasikan pada MongoDB.
2. Model dokumen yang terbentuk dari proses transformasi ini dapat berupa dokumen tunggal, *embedded document*, *referenced document*, atau kombinasi ketiganya bergantung dari tipe, *rule*, dan nilai kardinalitas relasi antar tabel yang dimodelkan.
3. Model *referenced document* tidak dapat menjaga *referential integrity* pada proses *update* dan *delete* data. Berbeda dengan model *embedded document*, model *referenced document* tidak dapat dapat menangani aturan relasi *restrict* dan *cascade* pada proses *update* dan *delete* data. Pada model *embedded document*, *tuple* data dari setiap tabel yang berelasi digabungkan sedemikian rupa menjadi sebuah dokumen agar proses *update* dan *delete* terhadap suatu data otomatis menjadi kesatuan yang utuh. Ini menyebabkan integritas data yang ada di dalam basis data dapat terjaga. Sedangkan pada model *referenced document* dimana data yang saling berkaitan masih terpisah di dokumen berbeda, hal ini menyebabkan proses *update* dan *delete* data tidak menjadi kesatuan yang utuh sehingga integritas data tidak dapat dipertahankan.
4. Pada pemodelan data, terdapat beberapa keterbatasan pada tipe data *numeric* yang dimiliki oleh MongoDB. Tipe data "NumberLong" yang merupakan tipe data *numeric* terbesar pada MongoDB tidak mampu menyamai rentang nilai yang dimiliki oleh tipe data "bigint" dan "decimal".

5. SARAN

Dari penelitian yang telah dilakukan, adapun saran yang dapat diberikan untuk pengembangan penelitian ini antara lain:

1. Kedepannya dapat dilakukan penelitian untuk transformasi skema basis data relasional menjadi model dokumen pada MongoDB pada RDBMS selain MySQL. Setiap RDBMS memiliki karakteristik tersendiri sehingga membutuhkan penyesuaian yang berbeda-beda dalam proses pembentukan model dokumen.
2. Untuk mendapatkan model dokumen yang lebih sesuai dengan kebutuhan aplikasi dapat dilakukan penelitian tentang pembentukan model dokumen yang tidak hanya didasarkan pada skema basis data, tetapi harus disesuaikan pula dengan *query* pengaksesan data yang dilakukan selama berjalannya aplikasi. Dengan demikian, model dokumen yang dibentuk tidak hanya digunakan sebagai model penyimpanan

data saja, namun dapat pula meningkatkan performa pengaksesan data yang dilakukan aplikasi ke dalam basis data.

DAFTAR PUSTAKA

[1] Anonim, 2013, *RDBMS to MongoDB Migration Guide: A MongoDB White Paper*, MongoDB Inc., New York.
[2] Lamoreaux, B., 2012, *Migrating from Oracle – How Apollo Group Evaluated MongoDB: A MongoDB White Paper*, MongoDB Inc., New York.
[3] Fong, J., Pang, F. dan Bloor, C., 2001, *Converting Relational Database into XML Document*, Strategic Research Grant 7001078, City University of Hong Kong, Hong Kong.
[4] Riskadewi dan Karya, G., 2004, Representasi dan Sinkronisasi Basis Data Relasional dengan Dokumen XML, *J. Integral*, 9, 1, 27-35.
[5] Bonnet, L., Laurent, A., Laurent, B., Sala, M. dan Sicard, N., 2011, REDUCE, YOU SAY: What NoSQL can do for Data Aggregation and BI in Large Repositories, *Proceedings of the 22nd International Workshop on Database and Expert Systems Applications*, IEEE Computer Society, hal. 483-488.
[6] Zhao, G., Weichai, H., Liang, S. dan Tang, Y., 2013, Modeling MongoDB with Relational Model, *Proceedings of the 4th International Conference on Emerging Intelligent Data and Web Technologies*, IEEE Computer Society, hal. 115-121.
[7] Kaur, K. dan Rani, R., 2013, *Modeling and Querying Data in NoSQL Database,*. IEEE International Conference on Big Data 2013.
[8] Kanade, A., Kanade, S. dan Gopal, A., 2014, A Study of Normalization and Embedding in MongoDB, *Proceedings of the IEEE International Advance Computing Conference (IACC 2014)*, IEEE Computer Society, hal. 416-421.
[9] Anonim, 2014, *MongoDB Documentation Release 2.6.0*, MongoDB Inc., New York.