BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis Sistem

Analisis sistem merupakan proses menganalisa uraian tentang keterkaitan permasalahan baik berupa dugaan atau fakta yang terjadi terhadap sistem dengan melakukan perbandingan yang relevan terhadap teori-teori yang telah dipahami sehingga menyebabkan terjadinya suatu penelitian, dalam hal ini adalah pada sebuah sistem manajemen basis data relasional secara umum. Kemudian setelah mengemukakan permasalahan maka ditemukan sebab permasalah dalam sistem manajemen basis data relasional tersebut, sehingga memudahkan proses penelitan dengan manganalisis penerapan suatu solusi yakni menerapkan sistem manajemen basis data non-relasional.

3.1.1. Analisis Masalah

Analisis masalah merupakan hal yang pertama kali dilakukan dalam suatu penelitian, menganalisis berarti mengidentifikasikan permasalah yang terjadi pada suatu sistem. Masalah dapat diidentifikasikan berdasarkan penyebab, sehingga akibat dari permasalahan tersebut dapat tangani dengan menerapkan suatu solusi.

Sebuah perangkat lunak berbasis web yang menggunakan fitur manajemen sistem basis data memerlukan pengolahan data yang cepat melalui penggunaan sintak *query*. Selain itu, pemodelan yang diterapkan dalam manajemen sistem basis data harus sedinamis mungkin, hal ini dikarenakan kebutuhan pemeliharaan data dan pengolahan data yang akan dilakukan dikemudian hari, berikut rincian analisis masalah berdasarkan rumusan masalah yang telah dipaparkan sebelumnya pada subbab **1.2** adalah :

1. Pemeliharaan data terkait dalam hal basis data berarti mengubah, menambah, atau menghapus struktur dalam penampungan data. Kegiatan ini dilakukan dikarenakan adanya kebutuhan suatu sistem. Imbas dalam pemeliharan data seperti ini adalah perubahan beruntun terhadap proses *server-side programming* dalam pengolahan data yakni (jika diperlukan) merubah pola pengolahan data yang terkait terhadap perubahan yang

dilakukan terhadap pemeliharaan basis data. Sehingga proses ini akan memakan waktu dalam proses pemeliharaan.

2. Pengolahan data, dalam penelitian ini didapat analisis masalah terhadap pengolahan data secara spesifik yaitu proses *querying* data. Pada kasus yang ditemukan untuk merubah, menambah, menghapus, atau mungkin mendapatkan baris data yang terkait dalam relasional data seorang *programmer server-side* harus memperhatikan *constraints* atau ketergantungan data yang ada didalam relasional tersebut, hal ini mengakibatkan eksekusi *query* berimbas pada waktu pemrosesan. Sebagai contoh dalam pembacaan data transaksi penjualan yang melibatkan banyak *constraints*.

Analisis masalah tersebut secara umum mengakibatkan kebutuhan waktu dalam hal pemrosesan data baik yang dilakukan dalam pemeliharan dan pengolahan data. Dalam sudut pandang keilmuan informatika uraian masalah yang telah dianalisis tersebut memerlukan suatu solusi yakni berupa desain basis data yang dinamis sehingga kebutuhan waktu dalam pemrosesan data baik pemeliharaan dan pengolahan dapat ditekan seminimal mungkin.

3.1.2. Analisis Basis Data yang Sedang Berjalan

Analisis basis data dilakukan untuk mengukur seberapa besar permasalah yang terdapat pada sistem manajemen basis data, sehingga pada akhirnya terkumpul informasi yang akan menuntun proses analisa dalam penelitian.

Subbab ini akan menjelaskan analisis basis data yang terkait dalam penelitan seperti daftar tabel, diagram relasi.

3.1.2.1. Daftar Tabel

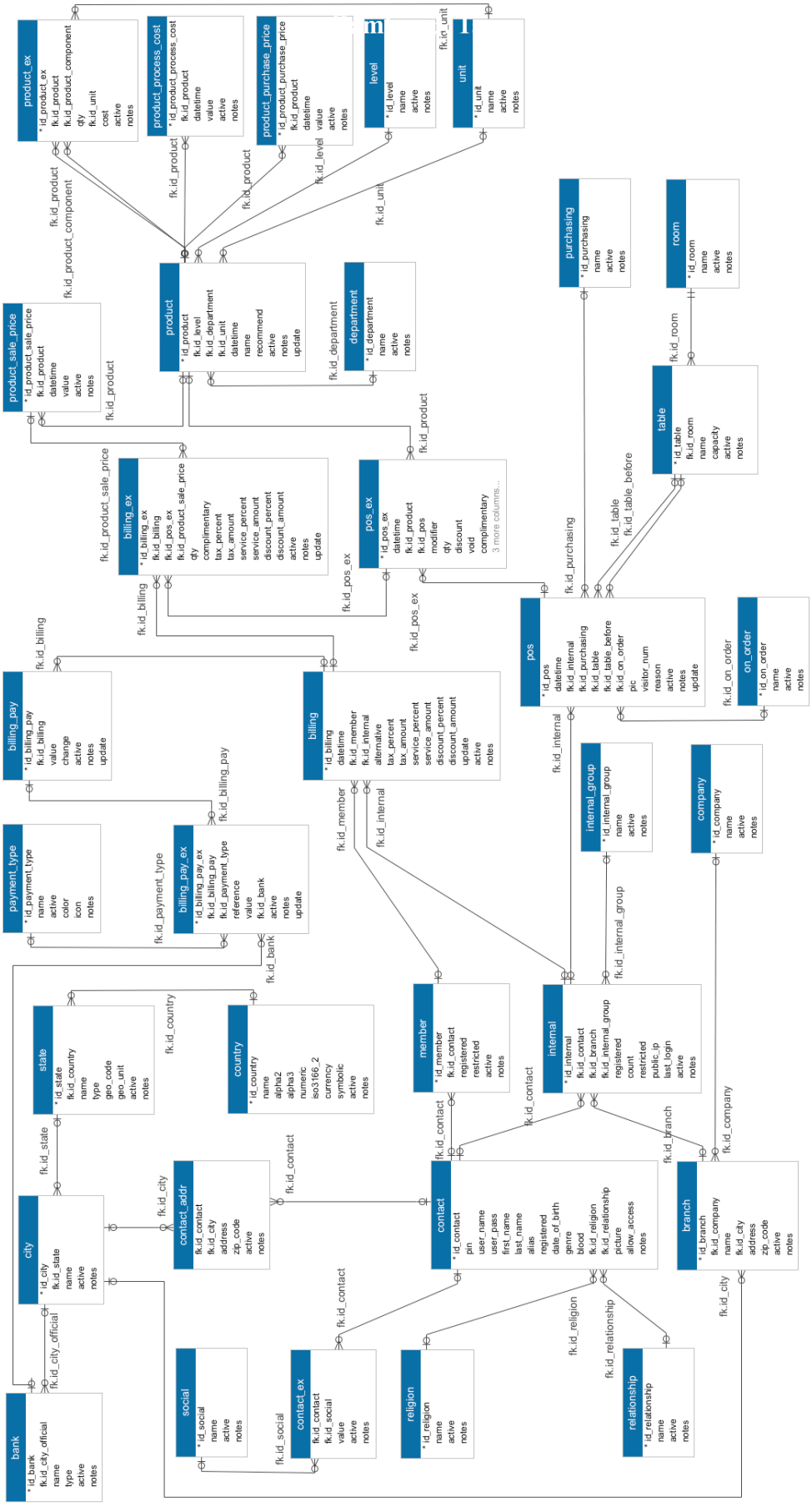
Berdasarkan daftar tabel pada Lampiran A.1 yang didapat dari PT. Xyclus Teknologi terhadap sistem manajemen basis data MySQL untuk aplikasi restoran, yang kemudian dilakukan proses revisi, sehingga dihasilkan rekap tabel yang dibentuk dengan batasan keperluan pengolahan data hanya pada proses transaksi produk atau menu yang dibeli konsumen. Berikut daftar tebel data yang dimaksud disajikan pada Tabel 3. 1.

Tabel 3. 1. Daftar tabel basis data dari aplikasi restoran setelah revisi

Name	***Engine***	***Rows***	***Avg_row_length***
bank	InnoDB	121	135
billing	InnoDB	3550	124
billing_ex	InnoDB	31468	150
billing_pay	InnoDB	3341	117
billing_pay_ex	InnoDB	3586	109
branch	InnoDB	1	16384
city	InnoDB	102	160
company	InnoDB	2	8192
contact	InnoDB	37	442
contact_addr	InnoDB	2	8192
contact_ex	InnoDB	2	8192
country	InnoDB	275	178
department	InnoDB	3	8192
internal	InnoDB	37	442
internal_group	InnoDB	9	1820
level	InnoDB	5	3276
member	InnoDB	0	0
on_order	InnoDB	11	1489
payment_type	InnoDB	7	2340
pos	InnoDB	7837	202
pos_ex	InnoDB	32802	144
product	InnoDB	638	128
product_ex	InnoDB	3010	76
product_process_cost	InnoDB	328	49
product_purchase_price	InnoDB	335	146
product_sale_price	InnoDB	301	54
purchasing	InnoDB	3	5461
relationship	InnoDB	2	8192
religion	InnoDB	3	5461
room	InnoDB	5	3276
social	InnoDB	11	1489
state	InnoDB	33	496
table	InnoDB	43	381
unit	InnoDB	24	682

3.1.2.2. Diagram Relasi

Diagram relasi merupakan hubungan antara tabel yang terdapat pada suatu *RDBMS*, berikut yang digunakan dalam penelitian ini terdapat pada Gambar 3.1.



Gambar 3. 1. Diagram relasi basis data aplikasi restoran ruang lingkup penjualan

3.1.3. Analisis penggunaan *query* pembacaan

Analisis penggunaan *query* pembacaan data diperlukan guna proses penerapan yang akan dilakukan, penggunaan *query* pembacaan data mencerminkan kebutuhan data yang berimbas pada bentuk pemodelan pada NoSQL MongoDB. Penjelasan ini telah dipaparkan sebelumnya pada subbab **2.4.5.2**, adapun rutinitas yang dimaksud dapat dilihat pada Tabel 3. 2.

Tabel 3. 2. Daftar *query* pembacaan yang digunakan

No	Konteks *query*	Tabel yang digunakan	Baris data	Rata-rata waktu (*ms*)
1	Harga jual produk	*product*, dan *product_sale_price*	300	13,5
2	Harga beli produk	*product*, dan *product_purchase_price*	337	19,4
3	Pemesanan (*master*)	*pos, contact, internal, internal_group, branch, purchasing, table, room,* dan *on_order*	8133	320,2
4	Pemesanan (*detail*)	*pos_ex, product*	32424	453,9
5	Rekap pendapatan (*billing*)	*billing, billing_pay, billing_pay_ex, payment_type,* dan *bank*	6	17.1

Berikut rincian *query* yang terdapat pada Tabel 3. 2, adalah sbb :

1. Harga jual produk

Query ini bertujuan untuk mengambil data harga jual terakhir pada produk, tabel yang diperlukan dalam *query* ini adalah *product*, *product_sale_price*.

```
# tabel yang digunakan : product, product_sale_price

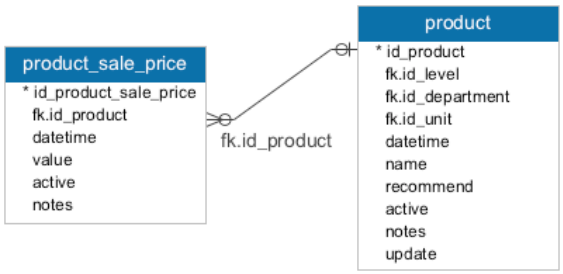
SELECT a.id_product, a.name, c.value
FROM product AS a
JOIN (
    SELECT * FROM (
    SELECT * FROM product_sale_price
    ORDER BY `datetime` DESC
  ) AS x GROUP BY `fk.id_product`
) AS c ON a.`id_product` = c.`fk.id_product`;
```

Untuk informasi yang didapat dalam penggunaan *query* ini disajikan pada Tabel 3. 3.

Tabel 3. 3. Baris data pembacaan *query* harga jual produk

id_product	*name*	*value*
9dac7a25	Ikan Malas Stim Kecap Asin 900 Gr	432000
9dc11a3d	Gado Gado Nusantara	22000
9e2874a1	Three-kind Baby Mushroom	38000
..	..	..
9edeb375	Nasi Hainan Ayam Panggang Asin	35000

Sedangkan diagram relasi untuk tabel yang digunakan dalam query tersebut dapat dilihat pada Gambar 3. 2.



Gambar 3. 2. Diagram relasi dalam penggunaan *query* pengambilan data harga jual produk

Berikut pengumpulan data yang dilakukan terhadap aktifitas data pada tabel-tabel tersebut yang dirangkum sebagai rutinitas pada suatu periode dapat dilihat pada Tabel 3. 4. Keterangan lebih lanjut terhadap periode rutinitas ini data dapat dilihat pada Lampiran C. 5, dan Lampiran C. 4.

Tabel 3. 4. Rutinitas tabel yang digunakan pada *query* pengambilan data harga jual produk.

No	**Nama Tabel**	*Create*	*Read*	*Update*	*Delete*
1	*product*	Jarang	~	Ada	Tidak Ada
2	*product_sale_price*	Jarang	Sering	Ada	Tidak Ada

Berdasarkan informasi yang didapat pada Tabel 3. 4, dan hasil *query* pada Tabel 3. 3, berikut analisis basis data yang didapat :

a. Data yang dihasilkan terpusat pada tabel *product*, hal ini mengakibatkan *product* adalah sebuah *collection*.

b. Dalam hal ini berdasarkan pembahasan sebelumnya pada subbab **2.4.5.2** dan **2.4.5.3**, data pada tabel *product* dan

product_sale_price yang termasuk dalam *query* ini memiliki intensitas pembacaan yang lebih tinggi dibandingkan pemrosesan lainnya sehingga hal ini akan mengalami *embedding* data *product_sale_price* pada data induk (*product*).

2. Harga beli produk

Query ini bertujuan untuk mengambil informasi data harga beli terakhir pada produk, tabel yang diperlukan dalam *query* ini adalah *product*, *product_purchase_price.*

```
# tabel yang digunakan : product, product_purchase_price

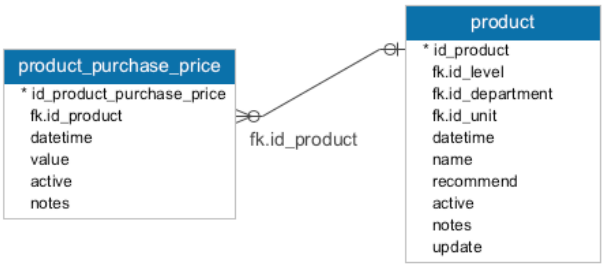
SELECT a.id_product, a.name, c.value
FROM product AS a
JOIN (
    SELECT * FROM (
    SELECT * FROM product_purchase_price
    ORDER BY `datetime` DESC
  ) AS x GROUP BY `fk.id_product`
) AS c ON a.`id_product` = c.`fk.id_product`;
```

Untuk informasi yang didapat dalam penggunaan *query* ini disajikan pada Tabel 3. 5.

Tabel 3. 5. Baris data pembacaan *query* harga beli produk

id_product	*name*	*value*
01e502db	Lada Putih Halus	80000
01ff0a49	Sunquick	48000
…	…	…
04d35373	Udang Kupas 40-50	105000

Sedangkan diagram relasi untuk tabel yang digunakan dalam query tersebut dapat dilihat pada Gambar 3. 3.



Gambar 3. 3. Diagram relasi *query* pengambilan data harga beli produk

Berikut pengumpulan data yang dilakukan terhadap aktifitas data pada tabel-tabel tersebut yang dirangkum sebagai rutinitas pada suatu periode

dapat dilihat pada Tabel 3. 6. Keterangan lebih lanjut terhadap periode rutinitas ini data dapat dilihat pada Lampiran C. 5, dan Lampiran C. 4.

Tabel 3. 6. Rutinitas tabel-tabel yang digunakan pada *query* pengambilan data harga beli produk

No	Nama Tabel	*Create*	*Read*	*Update*	*Delete*
1	*product*	Jarang	~	Ada	Tidak Ada
2	*product_product_price*	Jarang	Sering	Ada	Tidak Ada

Berdasarkan informasi yang didapat pada Tabel 3. 6, dan hasil *query* pada Tabel 3. 5 berikut analisis basis data yang didapat :

a. Data yang dihasilkan terpusat pada tabel *product*, hal ini mengakibatkan *product* adalah sebuah *collection*.
b. Dalam hal ini berdasarkan pembahasan sebelumnya pada subbab **2.4.5.2** dan **2.4.5.3**, data tabel *product* dan *product_sale_price* yang termasuk dalam *query* ini memiliki intensitas pembacaan yang lebih tinggi dibandingkan pemrosesan lainnya sehingga hal ini akan mengalami *embedding* data *product_sale_price* pada data induk (*product*)

3. Pemesanan produk (*master*)

Query ini bertujuan untuk mengambil informasi data pemesanan keseluruhan, tabel yang diperlukan dalam *query* ini adalah *pos, contact, internal, internal_group, branch, purchasing, table, room,* dan *on_order*.

```
# tabel yang digunakan :
# pos, internal, contact, internal_group, branch,
# purchasing, table, room, onorder

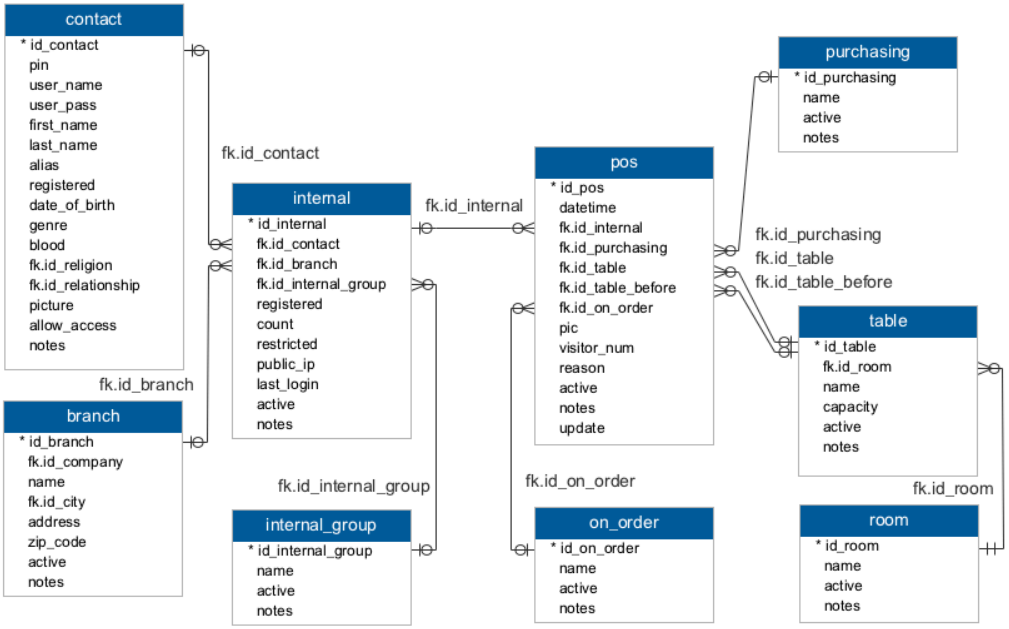
SELECT
  a.`datetime`, a.`pic`,
  a.`fk.id_internal`,
  b.`fk.id_contact`,
  ba.`user_name`, ba.`first_name`, ba.`last_name`,
  b.`fk.id_internal_group`,
  bb.`name`,
  b.`fk.id_branch`,
  bc.`name`,
  a.`fk.id_purchasing`,
  c.`name`,
  a.`fk.id_table`,
  a.`fk.id_table_before`,
  d.`fk.id_room`, d.`name`,
  da.`name`,
  e.`fk.id_room`, e.`name`,
  ea.`name`,
```

```
  a.`fk.id_on_order`,
  f.`name`
FROM pos AS a
LEFT JOIN `internal` AS b ON a.`fk.id_internal` = b.`id_internal`
LEFT JOIN `contact` AS ba ON b.`fk.id_contact` = ba.`id_contact`
LEFT JOIN `internal_group` AS bb
  ON b.`fk.id_internal_group` = bb.`id_internal_group`
LEFT JOIN `branch` AS bc ON b.`fk.id_branch` = bc.`id_branch`
LEFT JOIN `purchasing` AS c ON a.`fk.id_purchasing` = c.`id_purchasing`
LEFT JOIN `table` AS d ON a.`fk.id_table` = d.`id_table`
LEFT JOIN `room` AS da ON d.`fk.id_room` = da.`id_room`
LEFT JOIN `table` AS e ON a.`fk.id_table_before` = e.`id_table`
LEFT JOIN `room` AS ea ON e.`fk.id_room` = ea.`id_room`
LEFT JOIN `on_order` AS f ON a.`fk.id_on_order` = f.`id_on_order`;
```

Untuk informasi yang didapat dalam penggunaan query ini disajikan pada Tabel 3. 40, sedangkan diagram relasi untuk tabel yang digunakan dalam *query* tersebut dapat dilihat pada Gambar 3. 4.

Tabel 3. 7. Baris data pembacaan query master pemesanan produk

datetime	*fk.id_internal*	*user_name*	*first_name*	*last_name*	*group*	*fk.id_branch*	*…*	*type*	*fk.id_table*	*fk.id_room*	*tab_name*
2014-01-18	64315f8a	ila	Ila	Ila	Floor	a799a65a	…	Delivery	0989274e	63269ca6	Meja 001
2014-01-17	2fe5c559	roni	Roni	Sumara	Floor	a799a65a	…	Dine In	f8c07322	11d1cc4c	Meja 019
2014-01-17	06808b6d	oki	Oki	Agus Lukito	Captain	a799a65a	…	Dine In	60ba5046	63269ca6	Meja 015
2014-01-17	549f194a	asep	Asep	Sulaeman	Captain	a799a65a	…	Dine In	1ea3d6f3	63269ca6	Meja 014
…	…	…	…	…	…	…	…	…	…	…	…
2014-01-17	549f194a	asep	Asep	Sulaeman	Captain	a799a65a	…	Dine In	60ba5046	63269ca6	Meja 015



Gambar 3. 4. Diagram relasi *query* pengambilan data master pemesanan

Berikut pengumpulan data yang dilakukan terhadap aktifitas data pada tabel-tabel tersebut yang dirangkum sebagai rutinitas pada suatu periode (bulan) dapat dilihat pada Tabel 3. 7. Keterangan lebih lanjut terhadap periode rutinitas ini data dapat dilihat pada Lampiran C. 7, dan Lampiran C. 6.

Tabel 3. 7. Rutinitas tabel-tabel yang digunakan pada *query* pengambilan data master pemesanan.

No	Nama Tabel	*Create*	*Read*	*Update*	*Delete*
1	*pos*	Sering	~	Jarang	Tidak Ada
2	*contact*	Jarang	~	Ada	Tidak Ada
3	*internal*	Jarang	~	Ada	Tidak Ada
4	*internal_group*	Tidak Ada	~	Tidak Ada	Tidak Ada
5	*branch*	Tidak Ada	Sering	Tidak Ada	Tidak Ada
6	*purchasing*	Tidak Ada	Sering	Tidak Ada	Tidak Ada
7	*table*	Tidak Ada	Sering	Tidak Ada	Tidak Ada
8	*room*	Tidak Ada	Sering	Tidak Ada	Tidak Ada
9	*on_order*	Tidak Ada	Sering	Tidak Ada	Tidak Ada

Berdasarkan informasi yang didapat pada Tabel 3. 7, dan hasil *query* pada Tabel 3.40, berikut analisis basis data yang didapat :

a. Data yang dihasilkan terpusat pada tabel *pos*, hal ini mengakibatkan *pos* adalah sebuah *collection*.

b. Dalam hal ini berdasarkan pembahasan sebelumnya pada subbab **2.4.5.2** dan **2.4.5.3**, data *pos, contact, internal, internal_group, branch, purchasing, table, room,* dan *on_order* yang termasuk dalam *query* ini memiliki intensitas pembacaan yang lebih tinggi dibandingkan pemrosesan lainnya sehingga hal ini akan mengalami *embedding* data *contact, internal, internal_group, branch, purchasing, table, room,* dan *on_order* pada data induk (*pos*)

4. Pemesanan produk (*detail)*

Query ini bertujuan untuk mengambil informasi data pemesanan setiap produk, tabel yang diperlukan dalam *query* ini adalah *pos_ex, product.*

```
# tabel yang digunakan : pos_ex, product

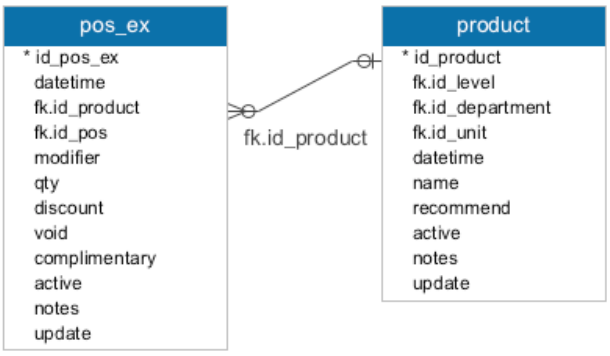
SELECT
  a.`fk.id_product`, b.name, a.modifier, a.qty,
  a.discount, a.void, a.complimentary
FROM
  pos_ex AS a
LEFT JOIN product AS b ON a.`fk.id_product` = b.`id_product`;
```

Untuk informasi yang didapat dalam penggunaan query ini disajikan pada Tabel 3. 8.

Tabel 3. 8. Baris data pembacaan *query* detil pemesanan

fk.id_product	*name*	*modifier*	*qty*	*discount*	*void*	*complimentary*
6fe4a40c	Iga Bakar Madu	PEDES	1	0	0	0
10034ea9	Kangkung Balacan	TIDAK PEDES	1	0	0	0
2bdde303	Tom Yum Kwetiew		1	0	0	0
798a6adc	Ayam Pete Kecap	PETE DI PISAH	1	0	0	0
…	…	…	…	…	…	…
ece9f82d	Pisang Goreng Coklat		1	0	0	0

Sedangkan diagram relasi untuk tabel yang digunakan dalam query tersebut dapat dilihat pada Gambar 3. 5.



Gambar 3. 5. Diagram relasi *query* pengambilan data detil pemesanan

Berikut pengumpulan data yang dilakukan terhadap aktifitas data pada tabel-tabel tersebut yang dirangkum sebagai rutinitas pada suatu periode dapat dilihat pada Tabel 3. 9. Keterangan lebih lanjut terhadap periode rutinitas ini data dapat dilihat pada Lampiran C. 7, dan Lampiran C. 6.

Tabel 3. 9. Rutinitas tabel-tabel yang digunakan pada *query* pengambilan data detil pemesanan

No	**Nama Tabel**	***Create***	***Read***	***Update***	***Delete***
1	*pos_ex*	Sering	~	Ada	Tidak Ada
2	*product*	Jarang	~	Ada	Tidak Ada

Berdasarkan informasi yang didapat pada Tabel 3. 9, dan hasil *query* pada Tabel 3. 8 berikut analisis basis data yang didapat :

a. Data yang dihasilkan terpusat pada tabel *pos*, hal ini mengakibatkan *pos* adalah sebuah *collection*.

b. Dalam hal ini berdasarkan pembahasan sebelumnya pada subbab **2.4.5.2** dan **2.4.5.3**, data *pos, pos_ex,* dan *product* yang termasuk dalam *query* ini memiliki intensitas pembacaan yang lebih tinggi dibandingkan pemrosesan lainnya sehingga hal ini akan mengalami *embedding* data *pos_ex,* dan *product* pada data induk (*pos*).

5. Rekap pendapatan (*billing*)

Query ini bertujuan untuk mengambil informasi data pemesanan setiap produk, tabel yang diperlukan dalam *query* ini adalah *billing, billing_pay, billing_pay_ex, payment_type,* dan *bank.*

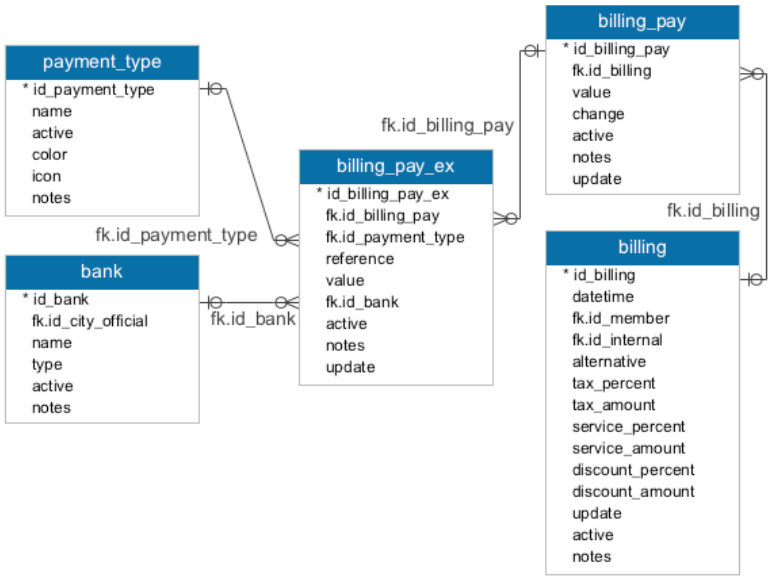
```
SELECT
  concat(d.`name`, ' (', count(d.`name`), ')') AS jenis_bayar,
  sum(c.value) - sum(b.change) AS pendapatan
FROM
  `billing` AS a
LEFT JOIN `billing_pay` AS b ON b.`fk.id_billing` = a.id_billing
LEFT JOIN `billing_pay_ex` AS c
  ON b.`id_billing_pay` = c.`fk.id_billing_pay`
LEFT JOIN `bank` AS ba
  ON ba.`id_bank` = c.`fk.id_bank`
LEFT JOIN payment_type AS d
  ON c.`fk.id_payment_type` = d.`id_payment_type`
GROUP BY d.`name`
ORDER BY b.`fk.id_billing`;
```

Untuk informasi yang didapat dalam penggunaan query ini disajikan pada Tabel 3. 10.

Tabel 3. 10. Baris data pembacaan *query* data master *billing*

jenis_bayar	pendapatan
Debit Card (311)	144802503
Down Payment (54)	14744150
Credit Card (3)	10159650
Cash (1540)	429492793
Complimentary (1614)	879809520

Sedangkan diagram relasi untuk tabel yang digunakan dalam query tersebut dapat dilihat pada Gambar 3. 6.



Gambar 3. 6. Diagram relasi *query* pengambilan data master billing

Berikut pengumpulan data yang dilakukan terhadap aktifitas data pada tabel-tabel tersebut yang dirangkum sebagai rutinitas pada suatu periode (bulan) dapat dilihat pada Tabel 3. 11. Keterangan lebih lanjut terhadap periode rutinitas ini data dapat dilihat pada Lampiran C. 9, dan Lampiran C. 8.

Tabel 3. 11. Rutinitas tabel-tabel yang digunakan pada *query* pengambilan data master rekap billing

No	**Nama Tabel**	***Create***	***Read***	***Update***	***Delete***
1	*billing*	Sering	~	Ada	Tidak Ada
2	*billing_pay*	Sering	~	Ada	Tidak Ada
3	*billing_pay_ex*	Sering	~	Ada	Tidak Ada
4	*payment_type*	Tidak Ada	~	Ada	Tidak Ada
5	*bank*	Tidak Ada	~	Tidak Ada	Tidak Ada

Berdasarkan informasi yang didapat dari Tabel 3. 11, dan hasil *query* pada Tabel 3. 10, berikut analisis basis data yang didapat :

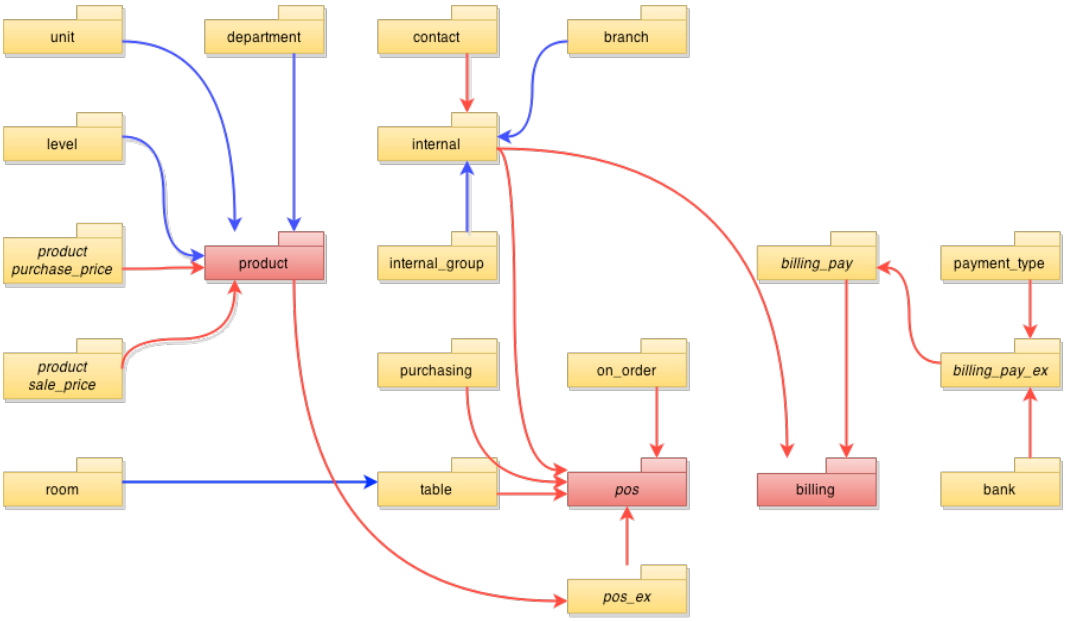
a. Hasil *query* yang dilakukan hanya merupakan proses *summary* data *billing*, sehingga data yang dihasilkan pada tabel *billing*, mengakibatkan *billing* adalah sebuah *collection*.
b. Dalam hal ini berdasarkan pembahasan sebelumnya pada subbab **2.4.5.2** dan **2.4.5.3**, data *billing, billing_pay, billing_pay_ex, payment_type,* dan *bank* yang termasuk dalam *query* ini memiliki intensitas pembacaan yang lebih tinggi dibandingkan pemrosesan lainnya sehingga hal ini akan mengalami *embedding* data *billing_pay, billing_pay_ex, payment_type, dan bank* pada data induk (*billing*)

3.1.4. Analisis Pemodelan Data NoSQL MongoDB

Berdasarkan proses analisis sebelumnya pada subbab **3.1.3**, pertimbangan terhadap analisis penggunaan *query* pembacaan menghasilkan rangkuman *collection* yang terbentuk, beberapa *collection* yang dimaksud diantaranya :

a. *Collection product* yang didapat berdasarkan analisis terhadap penggunaan *query* pembacaan *harga jual produk* dan *harga beli produk.*
b. *Collection pos* yang didapat berdasarkan analisis terhadap penggunaan *query* pembacaan *pemesanan* (*master*) dan *pemesanan* (*detail*).
c. *Collection billing* yang didapat berdasarkan analisis terhadap penggunaan *query* pembacaan transaksi *rekap pendapatan* (*billing*)

Analisis ini menghasilkan tiga *collection* dengan model *embedded*, berikut gambaran *collection* yang dimaksud terdapat pada Gambar 3. 7. Adapun keterangan terhadap komponen yang terkait dapat dilihat pada daftar simbol.



Gambar 3. 7. Pemodelan data NoSQL MongoDB untuk aplikasi restoran untuk proses konversi

Sedangkan keterangan terperinci terhadap Gambar 3. 7. terdapat pada Tabel 3. 12.

Tabel 3. 12. Daftar *collection* NoSQL MongoDB untuk aplikasi restoran

Collection	***References***	***Embedded***
product	*level, unit,* dan *department*	*product*, *product_purchase_price*, dan *product_sale_price*
pos	*Internal_group,* dan *branch*	*pos*, *contact*, *internal*, *purchasing*, *table*, *room*, *on_order*, *pos_ex*, dan *product*
billing	-	*billing*, *billing_pay*, *billing_pay_ex*, *payment_type*, dan *bank*

Berdasarkan korespondensi antara Gambar 3. 7 dan Tabel 3. 12 rincian model yang akan dilakukan konversi secara programatik pada subbab **3.1.4.1**. adalah sebagai berikut :

1. Collection product, bentuk pemodelan yang dihasilkan adalah :

```
{
  table: 'product',
  joins: [
    { table: 'unit', type : 'references' },
    { table: 'department', type : 'references' },
    { table: 'level', type : 'references' },
    { table: 'product_sale_price', type : 'embedded' },
    { table: 'product_purchase_price', type : 'embedded' }
  ]
};
```

2. *Collection pos*, bentuk pemodelan yang dihasilkan adalah :

```
{
  table: 'pos',
  joins: [
    { table: 'purchasing', type : 'embedded' },
    { table: 'on_order', type : 'embedded' },
    { table: 'table', type : 'embedded' },
    {
      table: 'internal',
      type : 'embedded',
      joins: [
        { table: 'internal_group', type : 'references' },
        { table: 'branch', type : 'references' },
        { table: 'contact', type : 'embedded' }
      ]
    },
    {
      table: 'pos_ex',
      type : 'embedded',
      joins: [
        { table: 'product', type : 'embedded' }
      ]
    }
  ]
};
```

3. *Collection billing*, bentuk pemodelan yang dihasilkan adalah :

```
{
  table: 'billing',
  joins: [
    {
      table : 'billing_pay',
      joins : [
```

```
        {
          table: 'billing_pay_ex',
          type : 'embedded',
          joins: [
            { table: 'bank', type : 'embedded' },
            { table: 'payment_type', type : 'embedded'}
          ]
        }
      ]
    }
  ]
};
```

3.1.4.1. Analisis Konversi Pemodelan Data NoSQL MongoDB

Analisis pemodelan data yang telah didapatkan pada subbab **3.1.4** merupakan kumpulan *collection* yang didapatkan dengan cara programatik. Dalan penelitian ini konversi data diperlukan guna mencapai titik poin eksistensi data yang telah ada dan berjalan sebelumnya, berikut cara atau metode yang akan digunakan dalam proses konversi adalah :

1. Membuat sebuah kelas khusus yang berjalan pada *server-side* dengan dua parameter inputan yaitu : suatu variabel model *collection* dan nama *collection*. Hal ini bertujuan untuk menghasilkan pembentukan *collection* terhadap model yang bersangkutan, dalam proses ini digunakan kelas dengan nama *Export,* dapat dilihat pada Lampiran B. 4.
2. Membuat lojik pemodelan data terhadap masing-masing *collection* berdasarkan analisis pemodelan data yang telah dijelaskan pada subbab **3.1.4**, kemudian instansiasi objek baru terhadap kelas *Export* dengan parameter logik pemodelan yang telah dibuat dan nama *collection*. Logik pemodelan lebih terperinci dapat dilihat pada bagian Lampiran B. 1.

3.2. Analisis Kebutuhan Non-funsional

Analisis kebutuhan non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan. Pada

analisis kebutuhan sistem non-fungsional ini dijelaskan analisis mengenai perangkat keras (hardware), perangkat lunak, dan pengguna (*user*) sebagai bahan analisis kekurangan dan kebutuhan yang harus dipenuhi dalam perancangan.

3.2.1. Analisis Kebutuhan Perangkat Keras

Perangkat keras yang digunakan dalam menerapkan non-relasional basis data NoSQL MongoDB dapat dilihat pada Tabel 3. 13.

Tabel 3. 13. Spesifikasi kebutuhan perangkat keras

Nama Perangkat	Spesifikasi
Processor	Dual Core 1.0 GHz
RAM	512 GB
Hard Drive	5 GB

3.2.2. Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan dalam menerapkan non-relasional basis data NoSQL MongoDB dapat dilihat pada Tabel 3. 14.

Tabel 3. 14. Spesifikasi kebutuhan perangkat lunak

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Windows, Mac OSX, Linux
Web Server	NodeJS v0.10.31
Database Server	MySQL v5.5.34
	MongoDB v2.6.4
Web Browser	Chrome, Firefox

3.2.3. Analisis Pengguna

Perangkat lunak yang dibangun dalam penelitian ini memiliki target pengguna, yakni dalam hal ini telah ditentukan minimal seorang database administrator yang memiliki hak akses penuh terhadap database server, dengan asumsi seorang pengguna mengerti konsep relasional basis data pada umumnya sehingga dapat berinteraksi dengan perintah query yang diberikan terhadap sistem.

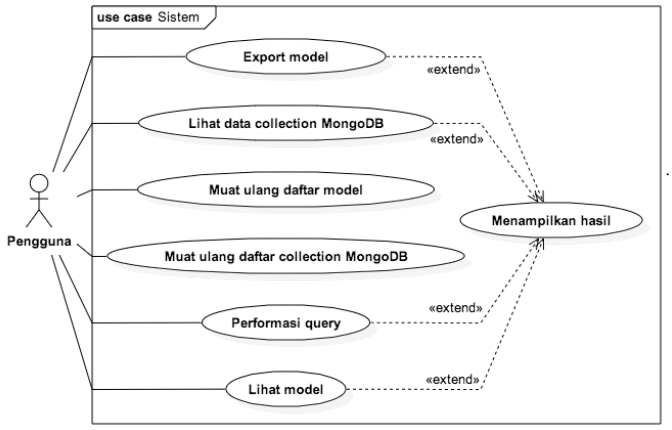
3.3. Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional membahas perancangan sistem yang dibangun dalam bentuk analisis diagram-diagram UML (*Unified Modeling*

Language), UML digunakan sebagai alat untuk menganalisis kebutuhan fungsional terhadap sistem yang dibangun.

3.3.1. *Use Case Diagram*

Use case diagram digunakan untuk memodelkan proses interaksi atau merepresentasikan interaksi antara aktor dengan sistem. Aktor disini adalah pengguna, yang dapat memulai penerapan non-relasional basis data NoSQL MongoDB, berikut disajikan pada Gambar 3. 8.



Gambar 3. 8. *Use case diagram*

3.3.1.2. *Use Case Scenario*

Tiap bagian *Use case* memiliki skenario yang menunjukkan proses apa saja yang terjadi, pengguna memberikan perintah dalam bentuk interaksi yang telah disajikan dalan tiap *Use case*, dan respon apa yang deiberikan sistem kepada pengguna terhadap interaksi tersebut, berikut disajikan pada Tabel 3. 15, Tabel 3. 16, Tabel 3. 17, Tabel 3. 18, Tabel 3. 19, dan Tabel 3. 20.

Tabel 3. 15. *Use case scenario* : Export query MySQL

Identifikasi	
Nama *Use case*	Export model
Tujuan	Menginstruksi sistem untuk membentuk pemodelan data *collection* pada suatu *database* MongoDB
Aktor	Pengguna
Deskripsi	
Skenario utama	
Kondisi awal	Pengguna berada pada halaman konversi
Aksi aktor	Reaksi sistem
1. Memilih salah satu model	3. Mengeksekusi perintah *export*

dari daftar model 2. Menekan tombol "Export"	4. Memberi *output* berupa respon data
Kondisi akhir	Pengguna malihat *output* yang ditampilkan sistem pada *container pesan*

Tabel 3. 16. *Use case scenario* : Lihat model

Identifikasi	
Nama *Use case*	Lihat model
Tujuan	Menginstruksi sistem menampilkan pemodelan terhadap suatu *collection*
Aktor	Pengguna
Deskripsi	
Skenario utama	
Kondisi awal	Pengguna berada pada halaman konversi
Aksi aktor	Reaksi sistem
1. Memilih salah satu model dari daftar model 2. Menekan tombol "Show model"	3. Mengeksekusi perintah *show model* 4. Memberi *output* berupa respon data
Kondisi akhir	Pengguna malihat *output* yang ditampilkan sistem pada *container pesan*

Tabel 3. 17. *Use case scenario* : Lihat data collection MongoDB

Identifikasi	
Nama *Use case*	Lihat data collection MongoDB
Tujuan	Menginstruksi sistem untuk menampilkan data collection
Aktor	Pengguna
Deskripsi	
Skenario utama	
Kondisi awal	Pengguna berada pada halaman konversi
Aksi aktor	Reaksi sistem
1. Memilih salah satu collection dari daftar collection 2. Menekan tombol "Show data"	3. Mengeksekusi perintah untuk menampilkan data collection 4. Memberi *output* berupa respon data
Kondisi akhir	Pengguna malihat *output* yang ditampilkan sistem pada *container pesan*

Tabel 3. 18. *Use case scenario* : Muat ulang daftar model

Identifikasi	
Nama *Use case*	Muat ulang daftar model
Tujuan	Memuat ulang daftar model
Aktor	Pengguna
Deskripsi	

Skenario utama	
Kondisi awal	Pengguna berada pada halaman konversi
Aksi aktor	Reaksi sistem
1. Menekan tombol "Logo refresh"	2. Mengeksekusi perintah muat ulang daftar model 3. Memberi *output* berupa data yang telah dimuat ulang
Kondisi akhir	Daftar query yang telah dimuat ulang

Tabel 3. 19. *Use case scenario* : Muat ulang daftar collection MongoDB

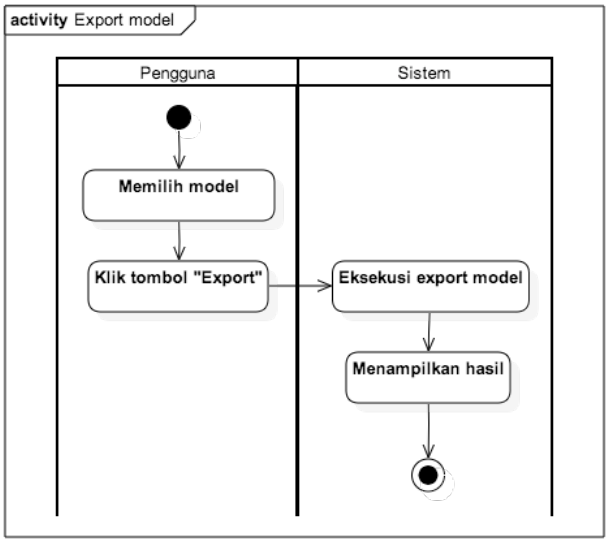
Identifikasi	
Nama *Use case*	Muat ulang daftar collection MongoDB
Tujuan	Memuat ulang daftar collection MongoDB
Aktor	Pengguna
Deskripsi	
Skenario utama	
Kondisi awal	Pengguna berada pada halaman konversi
Aksi aktor	Reaksi sistem
1. Menekan tombol "Logo refresh"	2. Mengeksekusi perintah muat ulang daftar collection 3. Memberi *output* berupa data yang telah dimuat ulang
Kondisi akhir	Daftar collection yang telah dimuat ulang

Tabel 3. 20. *Use case scenario* : Performansi query

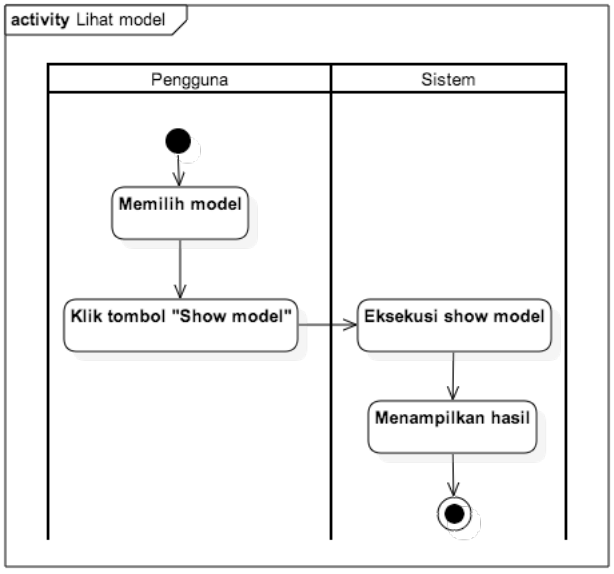
Identifikasi	
Nama *Use case*	Performansi query
Tujuan	Memberi perbandingan performa dari MySQL dan MongoDB (*query*)
Aktor	Pengguna
Deskripsi	
Skenario utama	
Kondisi awal	Pengguna berada pada halaman query
Aksi aktor	Reaksi sistem (*client-side*)
1. Meng-input query pada teks editor MongoDB dan MySQl 2. Menekan tombol "Submit"	3. Mengeksekusi query MongoDB dan MySQL 4. Memberi *output* berupa respon data
Kondisi akhir	Pengguna malihat *output* yang ditampilkan sistem pada *container pesan*

3.3.2. *Activity Diagram*

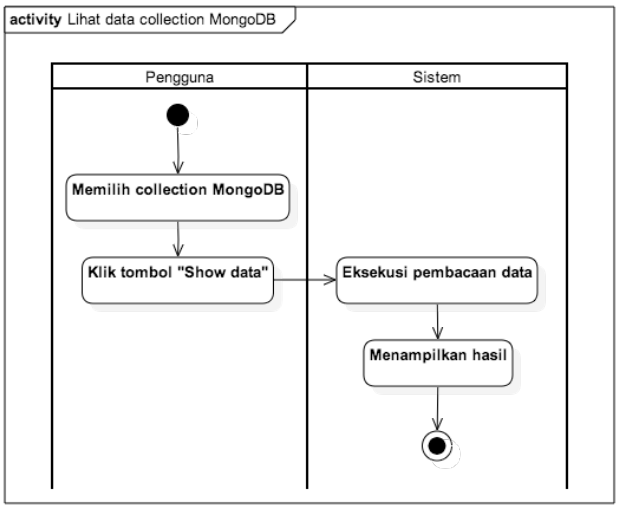
Activity diagram mendeskripsikan alur aktivitas dari setiap *Use case*. *Activity diagram* pada perangkat lunak ini ditunjukkan pada Gambar 3. 9, Gambar 3. 10, Gambar 3. 11, Gambar 3. 12, Gambar 3. 13, dan Gambar 3. 14.



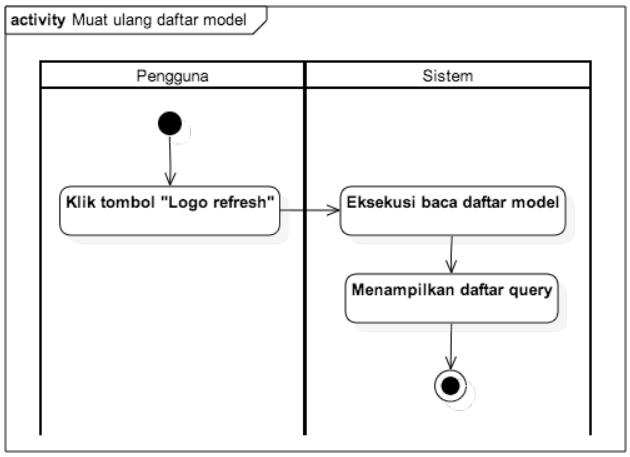
Gambar 3. 9. *Activity diagram* : Export model



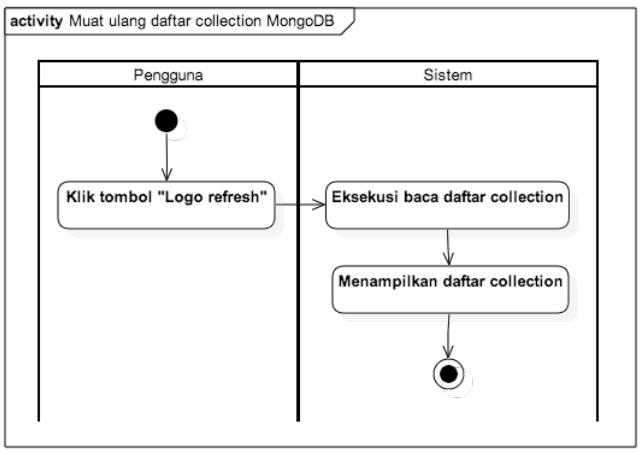
Gambar 3. 10. *Activity diagram :* Lihat model



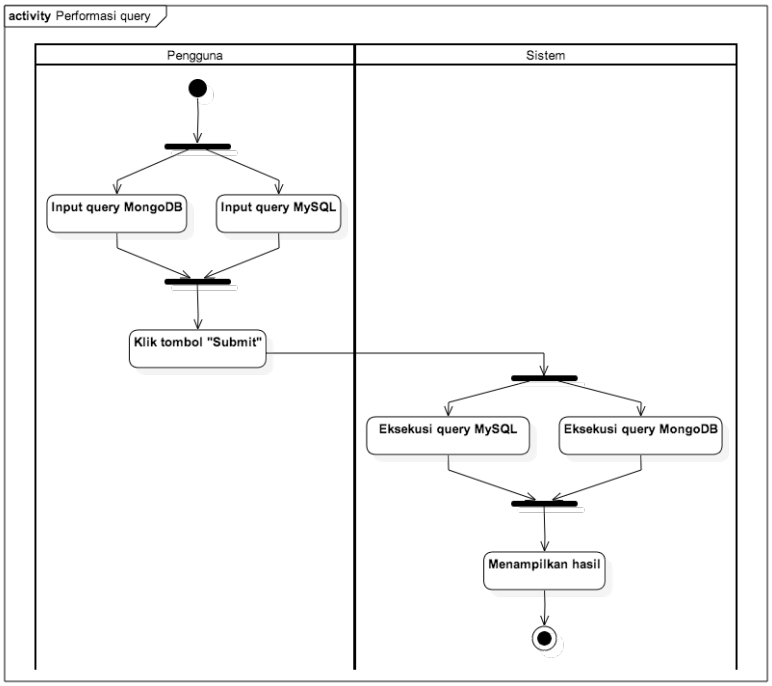
Gambar 3. 11. *Activity diagram* : Lihat data collection MongoDB



Gambar 3. 12. *Activity diagram* : Muat ulang daftar model



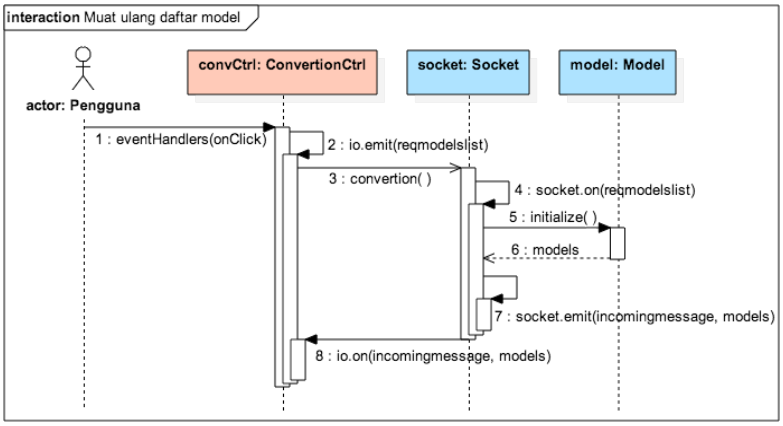
Gambar 3. 13. *Activity diagram* : Muat ulang daftar collection MongoDB



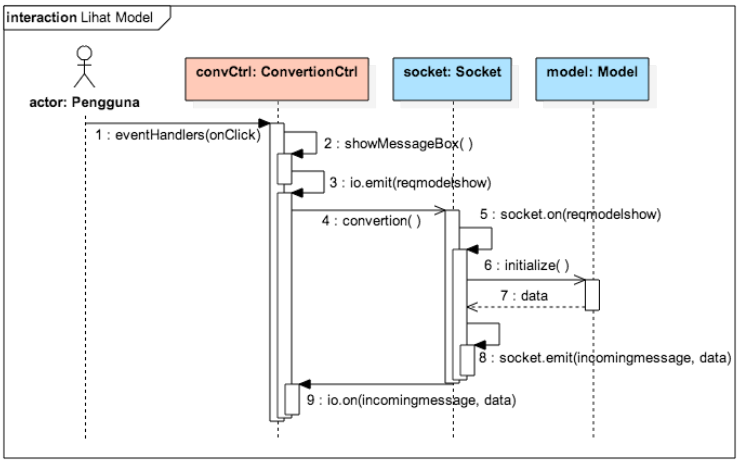
Gambar 3. 14. *Activity diagram* : Performansi *query*

3.3.3. Sequence Diagram

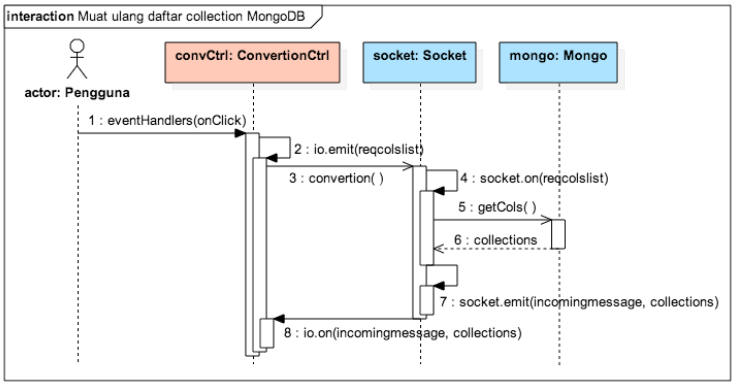
Sequence diagram menggambarkan interaksi antar masing-masing objek pada setiap *use case* dalam runtunan waktu. Interaksi ini berupa pengiriman serangkaian data antar objek-objek yang saling berinteraksi, berikut disajikan pada Gambar 3. 15, Gambar 3. 16, Gambar 3. 17, Gambar 3. 18, Gambar 3. 19, dan Gambar 3. 19.



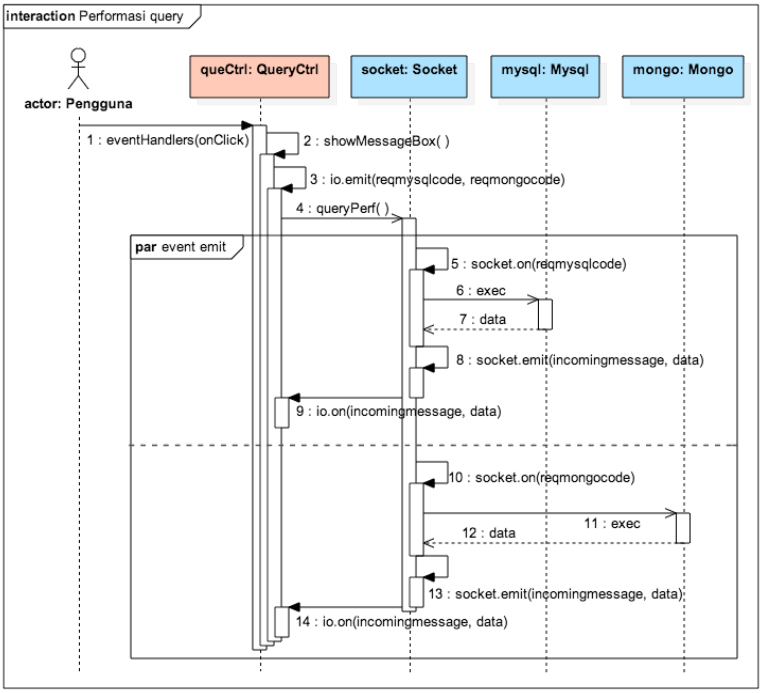
Gambar 3. 15. *Sequence diagram* : Muat ulang daftar model



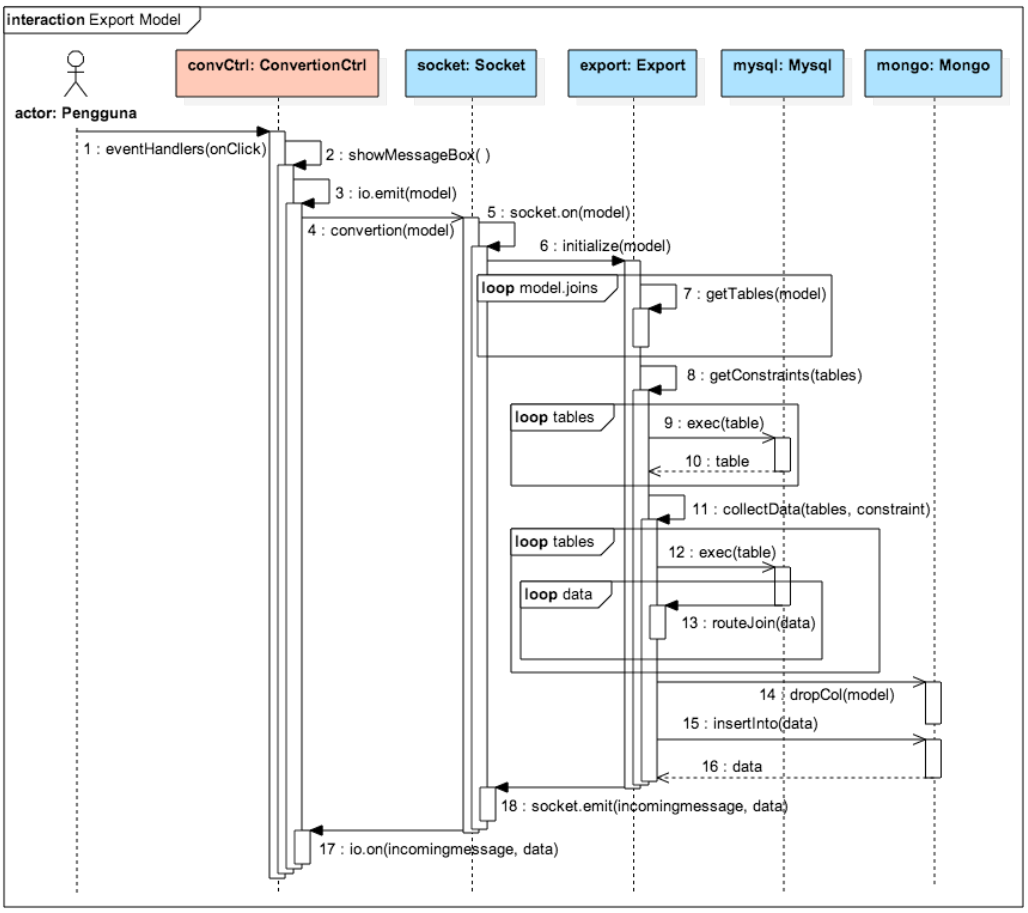
Gambar 3. 16. *Sequence diagram* : Lihat model



Gambar 3. 17. *Sequence diagram* : Muat ulang daftar collection MongoDB



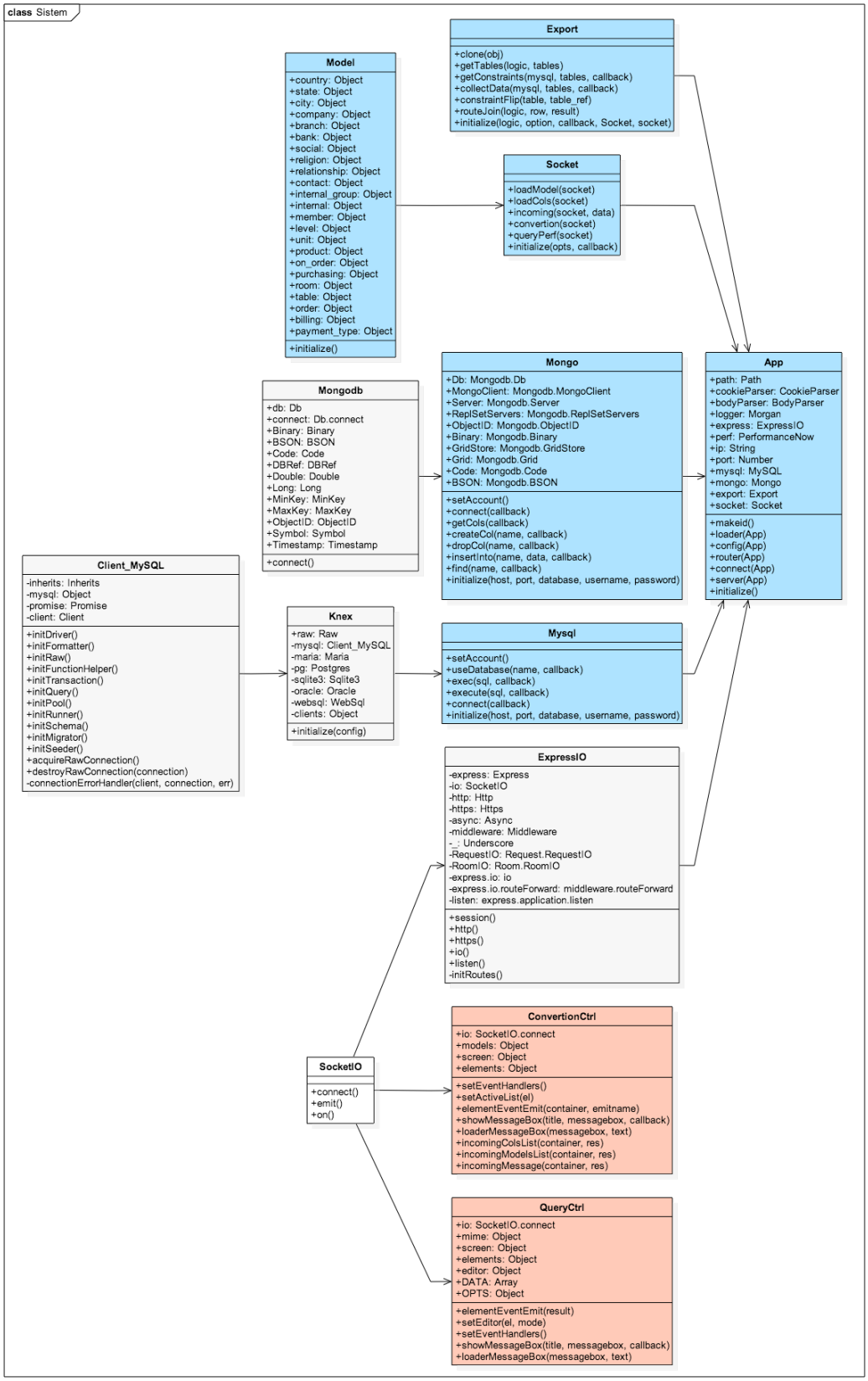
Gambar 3. 18. *Sequence diagram* : Performansi *query*



Gambar 3. 19. *Sequence diagram* : Export model

3.3.4. *Class Diagram*

Class diagram menggambarkan struktur dan hubungan antar objek-objek yang ada pada sistem. Struktur tersebut meliputi atribut dan method yang ada pada masing-masing class seperti yang disajikan pada Gambar 3. 20.



Gambar 3. 20. *Class Diagram*

3.4. Perancangan Sistem

Perancangan sistem merupakan kegiatan untuk merancang aplikasi yang akan di bangun, tahapannya dimulai dari perancangan model basis data non-relasional NoSQL MongoDB, perancangan antarmuka, dan jaringan semantik.

3.4.1. Perancangan Model Basis Data Non-relasional NoSQL MongoDB

Perancangan model basis data non-relasional NoSQL MongoDB terdiri dari *URL routing*, data awal RDBMS MySQL aplikasi restoran (yang telah direvisi), konversi pemodelan RDBMS ke non-relasional NoSQL MongoDB, dan penambahan data terhadap skema non-relasional NoSQL MongoDB, adalah sebagai berikut rincian perancangannya :

1. *Socket events*

Socket events digunakan untuk menjalankan segala proses yang akan dilakukan terhadap perancangan model basis data dalam menerapkan model non-relasional NoSQL MongoDB, setiap perintah yang terdapat pada *events* akan menghasilkan suatu proses yang guna operasonal data ataupun interaksi.

2. Data awal RDBMS MySQL aplikasi restoran

Data awal RDBMS MySQL aplikasi restoran merupakan basis data dalam bentuk relasional yang dimiliki PT. Xyclus Cipta telah mengalami perbaikan model untuk keperluan penelitian.

3. Konversi pemodelan RDBMS ke non-relasional NoSQL MongoDB

Konversi pemodelan RDBMS ke non-relasional NoSQL MongoDB merupakan pengolahan skema atau pemodelan untuk merubah model tabel RDBMS MySQL beserta *constraint* terhadap tabel lain ke dalam bentuk *collection* yang merupakan basis data dari non-relasional NoSQL MongoDB.

4. Penambahan data terhadap skema non-relasional NoSQL MongoDB

Penambahan data terhadap skema non-relasional NoSQL MongoDB merupakan penambahan data yang diambil dari pembacaan data (*select*

query) suatu tabel RDBMS MySQL beserta *constraint* terhadap tabel lain ke dalam bentuk *collection* yang merupakan basis data dari non-relasional NoSQL MongoDB.

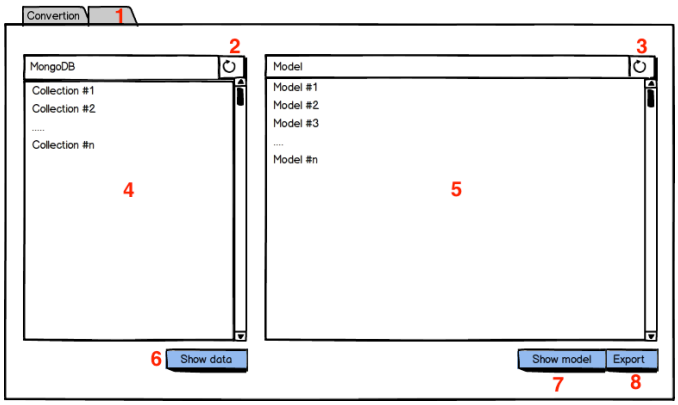
3.4.2. Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk memberikan gambaran visual tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam proses perancangan pemodelan basis data non-relasional basis data NoSQL MongoDB.

3.4.2.1. Perancangan Tampilan Konversi

Perancangan tampilan konversi bertujuan untuk mengambil pemodelan tabel RDBMS MySQL berserta *constraint* terhadap tabel lain, kemudian sistem akan melakukan pembacaan *query* yang sering digunakan pada studi kasus penelitian, selanjutnya untuk dijadikan data masukan terhadap suatu *collection* yang merupakan salah satu basis data dari non-relasional NoSQL MongoDB yang telah ditentukan, berikut pada Gambar 3. 21.

F01



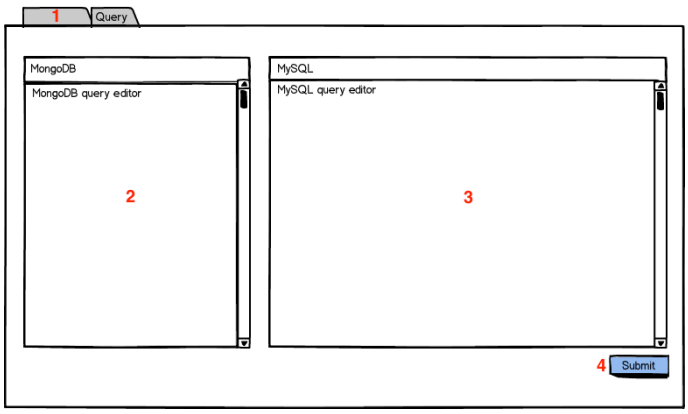
Keterangan
Nama form : F01 / Tampilan konversi
Ukuran layar : *Web browser window default size*

1. Menuju ke **F02.**
2. Tombol *reload,* ketika diklik akan memuat ulang daftar *collection* (4)
3. Tombol *reload,* ketika diklik akan memuat ulang daftar model (5)
4. Daftar *collection* suatu basis data MongoDB
5. Daftar *model.*
6. Tombol *show data,* ketika diklik akan menampilkan *result* pada *Container* pesan (**M01**), yang merupakan hasil eksekusi.
7. Tombol *show model*, ketika diklik akan menampilkan *result* pada *Container* pesan (**M01**), yang merupakan hasil eksekusi
8. Tombol *export*, ketika diklik akan menampilkan *result* pada *Container* pesan (**M01**), yang merupakan hasil eksekusi.

Gambar 3. 21. Perancangan tampilan konversi

3.4.2.2. Perancangan Tampilan Performansi *Query*

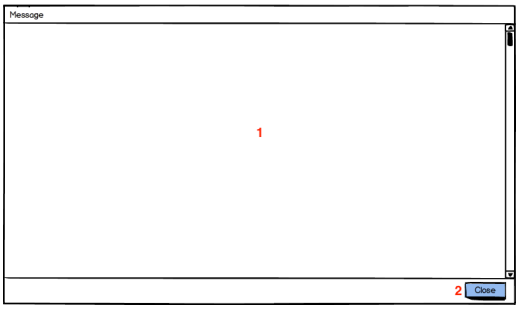
Perancangan tampilan perintah *query* merupakan tampilan pengujian perintah *query* yang dapat dilakukan pengguna terhadap basis data yang digunakan didalamnya, baik SQL MySQL ataupun NoSQL MongoDB, berikut disajikan pada Gambar 3. 22.

F02	
	1. Menuju ke **F01** 2. *TextEditor* MongoDB query, tempat pengguna mengetikkan query 3. *TextEditor* MongoDB query, tempat pengguna mengetikkan query 4. Tombol submit, ketika diklik akan menampilkan *result* pada *Container* pesan (**M01**), yang merupakan hasil eksekusi
Keterangan Nama form : F02 / Tampilan performansi query Ukuran layar : *Web browser window default size*	

Gambar 3. 22. Perancangan tampilan performansi *query*

3.4.2.3. Perancangan Tampilan Pesan

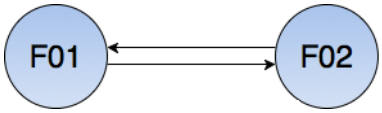
Perancangan tampilan hasil pengujian merupakan tampilan hasil dari setiap perintah yang dilakukan terhadap sistem, tampilan ini menghasilkan tampilan dalam bentuk *parsing text* berformat *JSON*, berikut terdapat pada Gambar 3. 23.

M01	
	1. *Container* pesan, tampilan hasil yang merupakan tampilan respon yang dipicu ketika pengguna berinteraksi dengan pengekseskusian sistem pada **F01** dan **F02**. 2. Tombol *Close,* ketika diklik akan menutup *Container* pesan
Keterangan Nama form : F03 / Tampilan hasil pengujian Ukuran layar : Web browser window default size	

Gambar 3. 23. Perancangan tampilan pesan

3.4.3. Jaringan Semantik

Jaringan semantik merupakan gambaran hubungan tentang navigasi halaman-halaman perancangan terhadap halaman perancangan lainnya. Berikut jaringan sematik terhadap perancangan-perancangan tampilan dalam penelitian ini dapat dilihat pada Gambar 3. 24.



Gambar 3. 24. Jaringan semantik terhadap perancangan tampilan