IMPLEMENTASI OBJECT RELATIONAL MAPPING PADA PETA DIGITAL SURABAYA BERBASIS

Arga Krismiyanto
Mahasiswa Jurusan Teknik Informatika, Fakultas Teknologi Informasi,
Institut Teknologi Adhi Tama Surabaya, Jl. Arief Rachman Hakim 100 Surabaya – 60117
Email : argacrish@gmail.com

Abstrak

Paradigma *Object-Oriented Programming* sangat banyak dipakai di kalangan pengembang perangkat lunak, di samping itu banyak perangkat lunak yang tidak bisa lepas dari penggunaan *Database Relational*. Keduanya sangat penting kegunaannya dalam pengembangan perangkat lunak. Akan tetapi, muncul ketidakcocokan paradigma antara keduanya yang sering disebut *Object-Relational Mismatch*, ketidakcocokan tersebut meliputi aspek *Granularity, Subtypes, Identity, Association,* dan *Navigasi Data*. Untuk mengatasi hal itu, dibutuhkan suatu metode pemetaan objek-objek yang mempresentasikan tabel-tabel entitas *database*, metode pemetaan tersebut adalah *Object Relational Mapping* (ORM). Pada aplikasi peta yang dibangun, metode pemetaan ini mempunyai peran untuk menangani pengolahan data informasi peta yang sangat banyak dan komplek. Di samping itu, peran metode pemetaan ini sangat penting terutama dalam hal kecepatan memproses data (*running time*). Hal ini dikarenakan metode pemetaan ini menyediakan model akses (*query*) yang tidak melibatkan secara langsung kepada *database*, melainkan pengaksesan terhadap objek-objek yang telah dipetakan. Model akses (*query*) yang disediakan oleh *Object Relational Mapping* (ORM) mempunyai perbandingan *running time* sebesar 1:1.51 lebih cepat, bila dibandingkan dengan sistem yang tidak menggunakan metode pemetaan ini, dengan batas pengujian jumlah *record* tabel sebanyak 50.000. Hal ini sangat menguntungkan untuk sebuah aplikasi *mobile.*

Kata kunci : *Object-Oriented Programming(OOP), Object Relational Mapping (ORM), Object-Relational Mismatch.*

Abstrac

Object-Oriented Programming Paradigm is very widely used among developers of software, in addition, a lot of software can not be separated from the use of Relational Databases. Both are very important usefulness in software development. However, emerging paradigm mismatch between them is often called Object-Relational Mismatch. It includes aspects of granularity, subtypes, Identity, Association, and Navigation Data. To overcome this, we need a method of mapping objects which presents tables database entity and it is called Object Relational Mapping (ORM). In the application building map, mapping method has a role to handle the data processing of map which is very much and complex. In addition, the role of mapping method is very important, especially in terms of data processing speed (running time). This is because the mapping method provides access model (query) that do not involve directly to the database but accessing to the objects that have been mapped. Access model (query) is provided by the Object Relational Mapping (ORM) has running time ratio of 1: 1:51 faster, if it is compared with systems that do not use this mapping method with the number of test limit records table as much as 50,000, It will have very beneficial for a mobile application.

PENDAHULUAN

Perkembangan paradigma pemrograman berorientasi objek saat ini sangat pesat, banyak perangkat lunak saat ini yang dikembangkan dengan paradigma berorientasi objek. Namun, muncul permasalahan ketidaksesuaian antara basisdata relasional dengan pengembangan aplikasi yang menggunakan konsep berorientasi objek yang sering disebut *Object-Relational Mismatch*, ketidaksesuaian tersebut meliputi aspek *Granularity, Subtypes, Identity, Association,* dan *Navigasi Data*.

Oleh karena ketidaksesuaian tersebut, maka diimplementasikan konsep *Object Relational Mapping* (ORM). Setiap objek yang akan memetakan menjadi tabel-tabel pada basis data relasional dibungkus oleh suatu interface dengan menerapkan konsep *design pattern*. Hal tersebut bertujuan untuk memudahkan integrasi dengan lapisan aplikasi (*controller*) mengakses data tersebut.

Implementasi *Object Relational Mapping* (ORM) ini dibagi menjadi beberapa tahapan, yaitu pemetaan *Data Transfer Object* dengan menghilangkan paradigma ketidaksesuaian antara basis data relasional dengan pemrograman berorientasi objek, membuat fungsi-fungsi akses data dengan mengimplementasikan *Data Access Object* (DAO) *pattern*, menyederhanakan fungsi akses data dengan memanfaatken konsep *design pattern*, membuat skema basis data, dan menyatukan lapisan persistensi dan lapisan aplikasi.

Pada aplikasi peta digital *offline*, implementasi *Object Relational Mapping* (ORM) sangat dibutuhkan untuk pemetaan objek-objek relasional dari informasi peta yang diperoleh, disamping itu sebagai model transaksi yang menyediakan layanan pengolahan data peta.

Berdasarkan latar belakang diatas dapat dirumuskan permasalahan yaitu :

1 Bagaimana mengimplementasikan metode *Object Relational Mapping* untuk memetakan informasi Peta Openstreetmap ke dalam *Object Relational*?
2 Bagaimana mengimplementasikan teknologi *Object Relational Mapping* sebagai penyedia layanan pengolah informasi pada Aplikasi Peta Digital *Offline* Android?

Dengan melihat permasalahan diatas, terdapat beberapa batasan masalah, yaitu sebagai berikut :

1 Menggunakan metode *Object Relational Mapping*.
2 Sumber data yaitu data xml Openstreetmap dan inputan dari user.
3 Menggunakan *framework* GreenDao.
4 Aplikasi Peta Digital bersifat *offline* dan berjalan di *platform* Android.
5 *Database* yang digunakan adalah SQLite *database*.
6 Minimal target android versi 2.30 (*Gingerbread*).

TINJAUAN PUSTAKA

Object-Relational Mismatch adalah ketidaksesuaian antara basis data yang menggunakan konsep relasional dengan

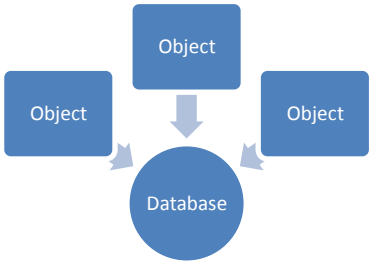
pengembangan aplikasi yang menggunakan konsep berorientasi objek. (Bauer, C. & Raja, G. 2007) memperkenalkan daftar objek / masalah ketidaksesuaian relasional:

1. *Granularity*
   Pemecahan *entity* pada atribut-atribut yang lebih kompleks. Misalnya *Class Person* mempunyai atribut *address* dengan tipe data *Address*. Sedangkan pada basis data relasional tidak memungkinkan pada table *Person* ada kolom *address* dengan tipe data *address*, yang mungkin dilakukan adalah memecah *address* menjadi *street_address, city_addres.*
2. *Subtypes*
   Pembeda antara *superclass* dan *subclass*. Pada pemrograman berbasis objek dikenal istilah *inheritance* (pewarisan) dari *class parent* kepada *class child*. Sedangkan pada basis data relasional tidak dikenal proses pewarisan antar tabel.
3. *Identity*
   Terdapat perbedaan fungsi antara lambang sama dengan (=) dan *method equals()* pada objek tetapi merujuk nilai yang sama pada *primary key* basis data relasional.
4. *Association*
   Hubungan dua entitas pada objek dikenal dengan *references* sedangkan pada relasional dikenal dengan *foreign key*. Asosiasi antar *entity* terdiri dari: *one-to-one*, *one-to-many*, dan *many-to-many*.
5. *Data Navigation*
   Proses pencarian pada basis data menggunakan *join table* sedangkan pada objek memanggil suatu *method getter*. Navigasi dibagi dua macam berdasarkan arahnya, antara lain: *directional* dan *bidirectional*. (Bauer dan King 2007).

Dengan adanya ketidaksesuaian tersebut, maka muncul metode pemetaan *Object Relational Mapping*, yaitu pemetaan dari model relasional ke model objek, atau sebaliknya, dimana tabel *database* dipetakan ke dalam kelas-kelas, atau sebaliknya kelas-kelas dipetakan ke dalam bentuk *database* relasional. (Ambler, Scott W. 2007).

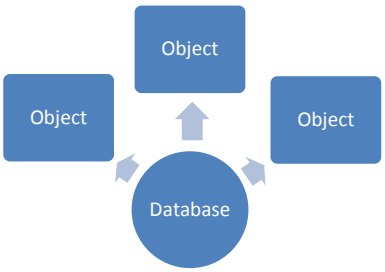
Dalam *Object Relational Mapping* (ORM) terdapat dua jenis pemetaan, yaitu *forward mapping* dan *reverse mapping*.

1. *Forward Mapping* akan mengambil objek yang sudah dibuat dalam *class* dan kemudian membuat skema *database* dari objek tersebut. Berikut adalah gambaran dari pemetaan jenis *Forward Mapping* :



Gambar 1. Pemetaan *forward mapping*

2. Sedangkan *Reverse Mapping* akan mengambil *database* yang sudah ada dan akan menciptakan objek-objek dari tabel-tabel yang sudah ada di dalam *database*. Berikut gambaran jenis pemetaan *Reverse Mapping:*



Gambar 2. Pemetaan *reverse mapping*

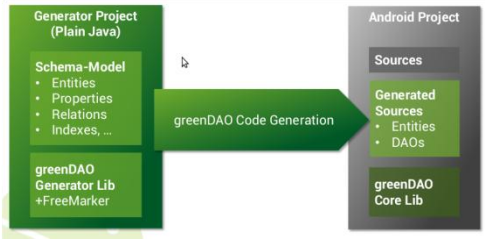
GreenDAO adalah sebuah proyek open source untuk membantu pengembang Android bekerja dengan data stored di SQLite. SQLite adalah salah satu *database* relasional yang mengagumkan dilingkungan Android. Namun, untuk mengembangkan itu memerlukan banyak hal yang harus dilakukan dengan kata lain dibutuhkan *tools* tambahan untuk mendukung kinerjanya. Menggunakan perintah *Structured Query Language* (SQL) dan parsing hasil query adalah pekerjaan cukup membosankan. GreenDAO akan sangat membantu dan menggantikan pekerjaan tersebut yaitu *mapping object* untuk tabel *database*. Dengan cara ini memungkinkan untuk dapat menyimpan, memperbarui, menghapus, dan permintaan untuk objek Java menggunakan API berorientasi objek sederhana. Tujuan desain utama GreenDAO ini

1. Kinerja maksimum
2. API mudah digunakan
3. Sangat dioptimalkan untuk Android
4. Pemakaian memori minimal
5. Ukuran *library* kecil

Berikut gambaran umum pembuatan entitas dan domain project dari GreenDAO :

Gambar 3. Fungsionalitas GreenDAO

GreenDAO juga menawarkan fitur *auto-generate* objek, yang di-*generate* secara otomatis sesuai dengan model skema entitas yang dibentuk. Berikut gambaran umum fitur auto-generate objek :



Gambar 4. Fitur auto-generate objek GrennDAO

Berikut adalah kelas-kelas inti dari *interface* penting hasil dari generator GreenDAO :

1. *DaoMaster*

   Fungsi utamanya adalah memegang objek database (SQLiteDatabase) dan mengelola kelas DAO (bukan objek) untuk skema tertentu. Kelas ini memiliki metode *static* untuk membuat atau menghapus tabel.

   Terdapat kelas helper yang merupakan implementasi dari SQLiteOpenHelper yang membuat skema dalam *database* SQLite .
2. *DaoSession*

   Mengelola semua objek *Data Access Object* (DAO) yang tersedia untuk skema tertentu, untuk memperolehnya menggunakan salah satu metode *getter*. DaoSession juga menyediakan beberapa *generic persistence methods* seperti *insert, load, update, refresh*

dan *delete* untuk entitas. Objek *DaoSession* juga melacak lingkup identitas.

3. *DAOs*
   Data Access *Object* (DAOs) yang bersifat *persist* dan digunakan untuk *query* entitas. Untuk setiap entitas yang dihasilkan GreenDAO mempunyai DAO. yang memiliki metode *persist* lebih banyak dari DaoSession, misalnya : *Count*, *LoadAll*, dan *insertInTx*.
4. *Entity*
   Entity adalah *object Persistable*. Objek-objek ini yang mewakili baris *database* menggunakan properti Java standar (seperti POJO atau JavaBean).

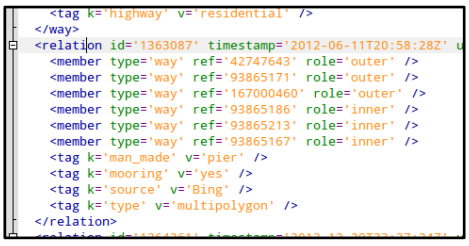
Openstreetmap adalah proyek yang bertujuan untuk membuat dan menyediakan data geografi seperti peta jalan secara gratis. Openstreetmap merepresentasikan informasi geografi tersebut dalam format *file* xml dengan empat element yang terdiri atas *Node*, *Relation*, *Way* dan *Tag*. Penjelasan dari masing-masing elemen:

1. *Node*
   Elemen ini mendefinisikan persimpangan atau lokasi. *Node* memiliki atribut utama id, lon (*longitude*), dan lat (*latitude*).
2. *Way*
   Elemen ini mendefinisikan jalan. *Way* hanya memiliki satu atribut utama yaitu id. *Way* terdiri atas satu atau lebih segment.
3. *Relation*
   Elemen ini mendefinisikan relasi antara *Node* dan *Way*.
4. *Tag*
   Elemen ini dapat mendefinisikan nama atau kategori dari lokasi. Elemen ini juga dapat mendefinisikan nama jalan, kategori jalan, dan arah dari jalan tersebut. Elemen ini dapat menjadi anak dari elemen *Node* , segment, dan *Way*.

METODE PENELITIAN

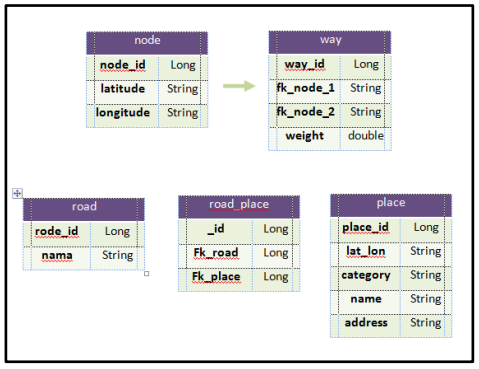
Metode *Object Relational Mapping* ini menggunakan dua pendekatan pemodelan, pendekatan pemetaan *table-per-concrete-class* dan pendekatan model pemetaan *forward mapping*. Pemetaan *table-per-concrete-class* yaitu pemetaan objek diperoleh dari entitas konkrit yang diperoleh,dalam artian setiap entity dipetakan menjadi satu objek, sedangkan pemodelan *forward mapping* adalah pembuatan database dari objek yang sudah dibuat, dengan kata lain membentuk terlebih dahulu objek-objek yang dibutuhkan.

Langkah pertama yaitu mengidentifikasi konkrit kelas dari sumber data Openstreetmap, pada tahap ini juga ditentukan objek-objek yang akan dibentuk dari hasil identifikasi konkrit kelas yang mungkin bisa dibentuk dan dijadikan satu objek. Berikut hasil identifikasi konkrit kelas :



Gambar 5. Isi dari file openstreetmap (*xml*)

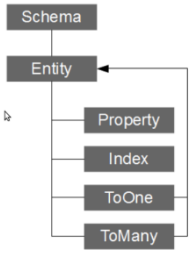
Dari data openstreetmap diatas, dapat kita peroleh hasil identifikasi, berikut konkrit kelas yang terbentuk.



Gambar 6. Hasil identifikasi konkrit kelas

Dari tabel tersebut, pada entitas way, penentuan atribut "*fk_node_1*" dan "*fk_node_1*" ditentukan berdasarkan teori *directed graph*, yaitu satu *way* mempunyai beberapa *node* didalamnya, oleh karena di dalam *way* ada *node* lebih dari satu maka kardinalitas dari *way* ke *node* adalah *many to one*, dengan *attribute* dari entitas *way* "*fk_node_1*" dan "*fk_node_2*" adalah *foreign key* dari entitas *node*.

Langkah kedua yaitu memetakan objek dari identifikasi entitas. GreenDAO menyediakan *generate* skema *database* dari objek yang dibuat, Berikut struktur skema objek dari GreenDAO :

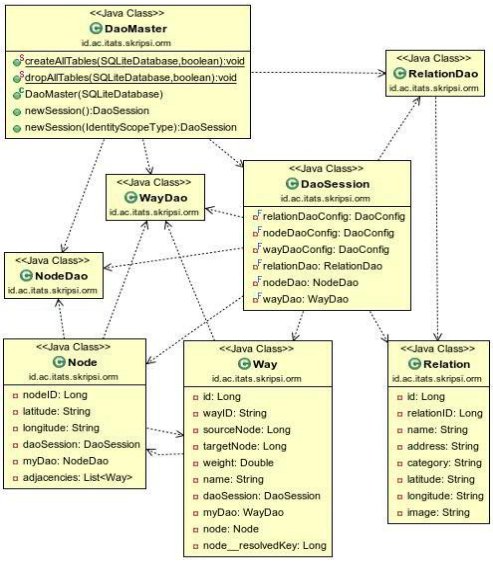


Gambar 7. Struktur Skema Greendao

Berikut potongan *source code* untuk pemetaan objek dengan :

```
. . .
Entity node = schema.addEntity("Node");
node.setTableName("NODE");
node.addLongProperty("nodeID").index().
primaryKey();
node.addStringProperty("latitude");
node.addStringProperty("longitude");

Entity way = schema.addEntity("Way");
way.setTableName("WAY");
way.addIdProperty().autoincrement();
way.addStringProperty("wayID").index();
Property sourceNode =
way.addLongProperty("fk_sourceNode").g
etProperty();
way.addToOne(node, sourceNode,
"sourceNode");
Property targetNode =
way.addLongProperty("fk_targetNode").ge
tProperty();
way.addToOne(node, targetNode,
"targetNode");

way.addDoubleProperty("weight");
node.addToMany(way, sourceNode,
"sourceAdjacencies");
node.addToMany(way, targetNode,
"targetAdjacencies");
. . .
```

Dari pendefinisian skema objek diatas, diperoleh beberapa kelas sesuai dengan skema yang telah didefinisikan. Berikut kelas diagram yang menunjukkan hasil dari pendefinisian skema diatas :

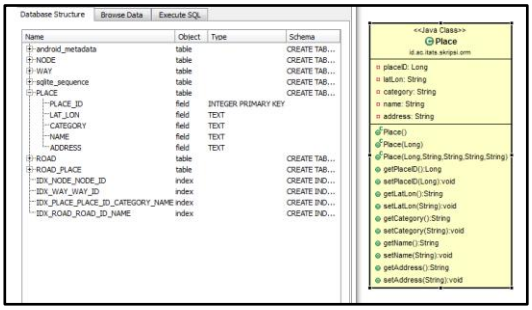


Gambar 8. Kelas diagram hasil pendefinisian skema

Langkah berikutnya adalah pembuatan *database* dari objek-objek yang telah dibuat. Berikut potongan *source code* pembuatan *database*-nya:

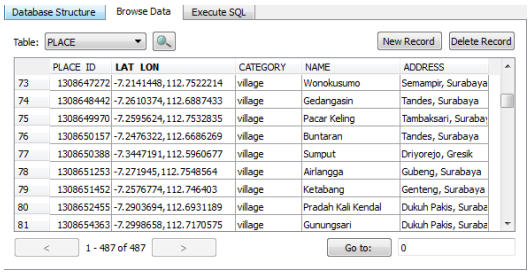
```
. . .
private static String DB_NAME =
"ormDB";
private static String DB_PATH = "";
private static SQLiteDatabase dataBase;
private Context context =
RoutingEngine.getAppContext();
private static DevOpenHelper openHelper;
private DaoMaster daoMaster;

DB_PATH =
context.getApplicationInfo().dataDir +
"/databases/";

openHelper = new
DevOpenHelper(context, DB_NAME,
null);

dataBase =
openHelper.getReadableDatabase();
daoMaster = new DaoMaster(dataBase);
. . .
```

Objek-objek yang telah dibuat adalah merupakan representasi dari tabel-tabel yang ada di *database*. Berikut gambaran dan selesai :



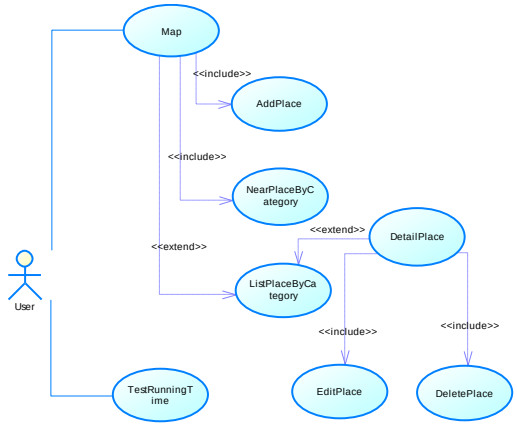
Gambar 9. Gambaran objek dan database

Setelah perancangan objek dan *database*, langkah berikutnya adalah *parsing* data *xml* dari Openstreetmap sebagai master data awal dari aplikasi ini, dan juga proses input data ke *database* yang telah dibuat. Berikut hasil parsing dan input data ke dalam *database* :



Gambar 10. Isi *database* hasil dari inputan awal

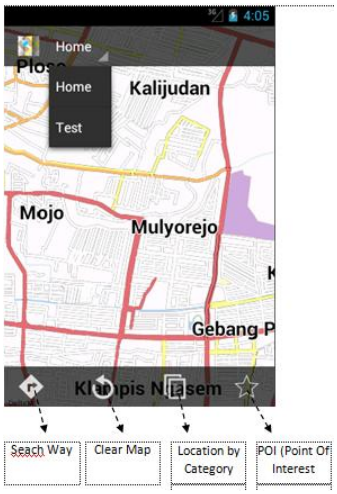
Perancangan *user interface* dilakukan untuk memudahkan user berinteraksi dengan sistem. Berikut diagram *use case* :



Gambar 11. *Use Case* Diagram Sistem

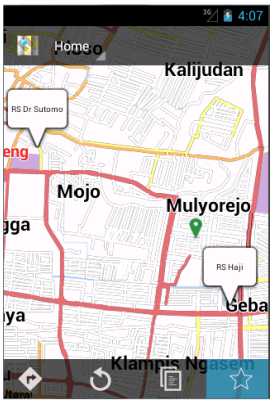
HASIL

Aplikasi yang dibagun ini digunakan untuk memudahkan user memanipulasi data khususnya untuk layanan CRUD (Create, Read, Update, dan Delete) data Peta Digital Surabaya. Berikut adalah tampilah awal dari aplikasi ini.



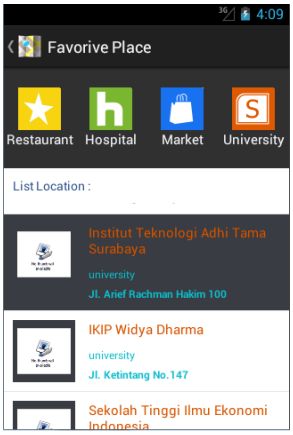
Gambar 12. Tampilan utama aplikasi

Tampilan utama aplikasi mempunyai fitur utama yaitu menu pencarian POI (*Point Of Interest)* atau lokasi menarik di surabaya sesuai kategori yang dipilih, disamping itu user bisa menambah lokasi yang diinginkan dan menyimpannya. Berikut adalah tampilan *Point Of Interest*.



Gambar 13. Tampilan *Point Of Interest*

Menu *Location By Category* digunakan untuk mencari nama lokasi sesuai dengan kategori yang dipilih, pada menu ini user disediakan *interface* untuk men-*edit* atau menghapus data peta. Berikut adalah tampilan halaman *Location by Category*.



Gambar 14. Tampilan *Location by Category*

Pengujian digunakan untuk membandingkan hasil kinerja sistem dengan menggunakan metode *Object Relational Mapping* (ORM) dan hasil kinerja sistem tidak menggunakan metode ini. Pengujian aplikasi dilakukan di dua media, yaitu pengujian pada *Emulator* dan pada *Handphone.*

1. Spesifikasi *Emulator*

Android versi 4.2.2 – API Level 17, RAM 512 Mb, VM Heap 128, SD Card 200 Mb, CPU ARM armeabi-v7a.

2. Spesifikasi *Handphone*
   Android versi 4.1.1 – API Level 16, RAM up to 512Mb, CPU snapdragon 1 Ghz, SD card.

Parameter pengujian sistem yang dipakai adalah tabel Road_Place yang mempunyai 99.666 record dan terdiri dari 3 kolom yaitu id (*long integer*), fk_road (*long integer*), fk_place (*long integer*). Model *query* pengujian yang dipakai menggunakan *query object* yang dimiliki oleh *framework Object Relational Mapping* (ORM) yaitu GreenDAO dan model query menggunakan sintak *Structured Query Language* (SQL) Cursor. Dalam pengujian ini keduanya akan dibandingkan hasil running time dalam satuan second. Berikut hasil pengujian di *Emulator*.

Tabel 1. Hasil pengujian di *Emulator*

Limit	Running Time Read Table (second)	
Record	ORM (GreenDao)	SQL Cursor
1000	0.17575671	0.12699564
5000	0.655504	0.91795516
10000	1.4218229	1.772055
15000	1.9257964	2.772172
20000	2.9430258	4.3937783
25000	4.2848606	6.1339526
30000	6.6333237	8.869842
35000	7.9629455	12.203024
40000	10.338376	15.490026
45000	11.8720665	19.030912
50000	15.182456	22.292103
Average	5.763266737	8.545710518
Perbandingan	1	1.48

Tabel 2. Hasil Pengujian di *Handphone*

Limit	Running Time Read Table (second)	
Record	ORM (GreenDao)	SQL Cursor
1000	0.063543335	0.05637333
5000	0.1397767	0.17734
10000	0.24597499	0.33923334
15000	0.350615	0.50091004
20000	0.63613	0.718085
25000	0.6579084	0.9721383
30000	0.93967336	1.4790934
35000	1.066066	1.7843167
40000	1.521885	2.16342
45000	1.6020634	2.7548666
50000	2.0514967	3.0172317
Average	0.843193899	1.269364401
Perbandingan	1	1.51

Pada hasil pengujian sistem dapat diperoleh rata-rata dan perbandingan *running time* yang menunjukkan bahwa sistem yang menggunakan metode *Object Relational Mapping* (ORM) lebih cepat dengan perbandingan 1 : 1.48 pada *Emulator* dan 1 : 1.51 pada Handphone.

Proses *query* yang dilakukan keduanya sama-sama mempunyai hasil (*return*) berupa *List object*, dan batas running time yang dihitung belum sampai hasil (*return*) ditampilkan, bisa dikatakan proses yang dihitung hanya sampai membentuk *List object* yang berisi hasil *query*.

Model query yang dimiliki *Object Relational Mapping* (GreenDAO) lebih cepat dikarenakan pada saat *object-object*

ini di instansi, secara otomatis nilai (*values*) object tersebut sama dengan tabel yang direpresentasikan, sehingga pada saat query dilakukan, nilai pada object itulah yang dikembalikan (*return*). Berbeda dengan model *query* dengan menggunakan SQL Cursor yang secara langsung me-*request* data tabel yang ada di database.

KESIMPULAN

Pada hasil pengujian yang dilakukan, kesimpulan utama implementasi metode *Object Relational Mapping* pada aplikasi ini adalah running time proses lebih cepat dibandingkan tidak menggunakan metode ini. Kesimpulan secara keseluruhan adalah sebagai berikut :

1. Proses running time pada aplikasi lebih cepat, dari hasil pengujian pada emulator android, perbandingan antara sistem yang menggunakan Object Relational Mapping dengan sistem yang tidak memakai metode ini adalah 1:1.48 dalam satuan second.
2. Pengujian yang kedua yaitu pada handphone, proses running time aplikasi juga lebih cepat dengan perbandingan 1:1.51 dalam satuan second.
3. Kelemahan menggunakan aplikasi ini adalah pada saat awal aplikasi dijalankan, running time sedikit lambat dikarenakan sistem melakukan instansi semua object yang dibutuhkan oleh sistem.

DAFTAR PUSTAKA

Scott W. Ambler. 2000. Mapping Objects to Relational Databases: O/R mapping in detail.

Chris J Harrison, Omar M Sallabi, Stephen E Eldridge. 2002. A persistent programming environment for teaching object-oriented concepts, Computer Software and Applications Conference.

Bauer, C. & King, G. 2007. Java Persistence with Hibernate. Manning Publication Co.

Sanjaya, David Adi. 2008. Implementasi Model View Controller dan Object Relational Mapping pada Content Management System Sistem Informasi Keuangan. AITI Jurnal Teknologi Informasi, 6(1):30-44

.

Barnes, Jeffrey M. 2007. Object-Relational Mapping as a Persistence Mechanism for Object-Oriented Applications. Minnesota : the Department of Mathematics and Computer Science at Macalester College in Saint Paul.

Simitci, Aysun. 2008. Object-Relational Mapping in Database Design. Bothell : The Master of Science in Computing & Software Systems University of Washington.

PRABHU, P. 2013. MAPPING OBJECTS TO RELATIONAL DATABASE FOR OBJECT PERSISTENCE. Karaikudi : Assistant Professor in Information Technology Directorate of Distance Education Alagappa University.