APLIKASI *MOBILE* SQL UNTUK ADMINISTRASI BASISDATA POSTGRESQL MEMANFAATKAN JAVA SERVLET

Erick Costanio dan Hapnes Toba
Fakultas Teknologi Informasi, Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri No. 65 Bandung 40164
Email: costanio@gmail.com, hapnestoba@yahoo.com

ABSTRAK: *Mobile* SQL adalah alat administrasi basisdata PostgreSQL berbasis *mobile* menggunakan koneksi HTTP. Teknologi *mobile* yang bersifat fleksibel membuat administrator dapat segera melakukan perawatan basisdata jika diperlukan tanpa harus berada di lokasi server. Pengembangan aplikasi menggunakan teknologi Java ME dan Servlet, serta dapat dijalankan menggunakan telepon seluler sebagai klien dengan spesifikasi MIDP 2.0, CLDC 1.1 dan GPRS yang telah diaktifkan, sedangkan untuk komputer *server* diperlukan *web server* dan DBMS. Fitur utama pada aplikasi ini antara lain: menjalankan perintah SQL, membuat tabel dan pengguna, melakukan *backup* dan *recovery* basisdata, menulis dan menjalankan bahasa prosedural SQL - PL/pgSQL, serta mengirim *log* melalui *email*. Aplikasi Mobile SQL dapat digunakan untuk melakukan administrasi basisdata menggunakan telepon seluler.

Kata kunci: Mobile SQL, Java ME, DBMS, administrasi basisdata

***ABSTRACT**: Since a database administrator is not always be in a fix place, we need something that make it possible for the administrator to manage the database anywhere. A Mobile SQL technology is flexible enough for an administrator to do this maintenance job immediately, such as for query operation, and database recovery. Our goal is to develop such application using Java ME and Servlet technology, thus making it runs on every mobile device that has MIDP 2.0, CLDC 1.1, and GPRS enabled device. While on a server computer, it needs web server and a DBMS. For this purpose, an object-oriented DBMS Postgre has been choosen. The main features of this application are: executing SQL queries; creating tables and users; making database backup and recovery; creating and executing SQL procedural language – PL/pgSQL; and sending a log through email. This application will also notify another adminstrator through email for any changes that has been made in the database*

Keywords: *Mobile SQL, Java ME, Postgre, database administration, servlet*

PENDAHULUAN

Pada era komputerisasi sekarang, hampir semua kegiatan dikerjakan menggunakan komputer sehingga penyimpanan data juga dilakukan secara komputerisasi. Oleh karena itu, diperlukan suatu sistem bernama *Database Management System* (DBMS) yang menangani data secara komputerisasi. DBMS merupakan kumpulan *libraries*, aplikasi dan utilitas yang membuat pengembang aplikasi tidak perlu mengkhawatirkan detail penyimpanan dan pengaturan data.

DBMS yang berisi banyak data memerlukan administrator basisdata untuk mengatur dan merawat kumpulan data tersebut. Namun sering kali seorang administrator basisdata kewalahan merawat basisdata karena faktor tempat dan waktu, contohnya: administrator basisdata menerima panggilan mendadak di malam hari, hanya untuk meng-*update* beberapa *record* dalam basisdata. Contoh lain, administrator basisdata sedang sibuk sehingga tidak dapat mem-*backup* data, sedangkan hari ini adalah waktu terakhir untuk *backup*.

Dari contoh permasalahan di atas, penulis mengusulkan suatu aplikasi pada telepon seluler (*mobile application*) untuk dapat membantu administrator basisdata menjalankan fungsinya. Dengan aplikasi semacam ini, administrator basisdata tidak harus selalu datang ke *server* untuk melakukan *query*, dan hal ini tentu akan menjaga berlangsungnya proses bisnis [2]. Administrator basisdata cukup membuka aplikasi pada telepon seluler dan mengirim *query* ke *web server*, kemudian *Web server* akan mengakses basisdata dan memberikan *service* untuk *request* yang diinginkan itu.

ADMINISTRASI BASISDATA

Administrasi basisdata merupakan kumpulan tugas-tugas yang dilakukan seorang administrator basisdata untuk merawat data, menjaga *performance server* basisdata, dan mengamankan basisdata [4].

Salah satu dari beberapa alasan utama menggunakan basisdata adalah untuk memiliki kontrol diantara data dan program yang mengakses data tersebut. Orang yang bertindak sebagai kontrol diantara data dan program dinamakan administrator basisdata. Fungsi utama seorang administrator basisdata antara lain adalah sebagai berikut :

- Administrator basisdata membuat bagan basisdata dengan menjalankan statement DDL.
- Membuat struktur penyimpanan dan mendefinisikan metode akses.
- Melakukan modifikasi bagan basisdata.
- Memberikan hak akses suatu data kepada user.
- Perawatan rutin seperti: melakukan *backup* basisdata, memastikan cukupnya tempat penyimpanan data, mengawasi tugas-tugas yang berjalan pada basisdata dan memastikan tidak terjadi penurunan *performance*.

TEKNOLOGI JAVA ME

Teknologi Java ME sangat tergantung pada perangkat yang digunakan, bisa dari kemampuan perangkat atau dari dukungan perangkat terhadap Java ME [1, 3]. Java ME merupakan bagian dari Java SE tapi tidak semua *library* java SE dapat digunakan pada java SE dan Java ME memiliki beberapa *library* khusus yang tidak dimiliki java SE.

Konfigurasi dan *profile* yang digunakan pada aplikasi adalah konfigurasi CLDC 1.1 (*Connected Limited Device Configuration*) dan *profile* MIDP 2.0 (*Mobile Information Device Profile*). CLDC merupakan spesifikasi dasar dari Java ME yang berupa *kilobyte virtual machine*, *library* dan API (*Aplication Programming Interface*). MIDP adalah spesifikasi untuk sebuah profil Java ME. MIDP membahas sesuatu yang lebih spesifik untuk sebuah perangkat misalnya kemampuan layar perangkat, memori, kemampuan multimedia, inputan perangkat. Untuk menulis MIDP diperlukan sebuah aplikasi yang dinamakan MIDlet. Sebagai bagian dari kelas *java.micoedition.midlet*. MIDlet hanya dapat dikembangkan untuk perangkat MIDP dan hanya dapat digunakan pada MIDP dan CLDC APIs.

MIDlet membuat, mengalokasikan dan menghapus beberapa tipe berbeda dari sumber berdasarkan waktu hidup, termasuk obyek, data dan *threads*. MIDlet membuat obyek yang dibutuhkan ketika dieksekusi, menghapus obyek yang sudah tidak dibutuhkan atau ketika MIDlet diberhentikan sementara atau dihancurkan. MIDlet membaca data dari tempat penyimpanan ketika *startup* dan menyimpan data ketika dimatikan. MIDlet menciptakan *threads* untuk melakukan tugas secara bersamaan dan menghentikan *thread* ketika waktunya tepat.

HTTP *SERVLET*

Hyper Text Transfer Protocol (HTTP) adalah *protocol* komunikasi yang diperlukan setiap perangkat MIDP untuk melakukan koneksi dan meminta informasi. HTTP membuat *streaming* yang membawa pesan antara HTTP *Client* dan *web server* dimana *web server* akan mengembalikan hasil *request* berupa dokumen dan *responds code* kepada client. Hal ini bekerja secara baik pada lingkungan *desktop* tapi tidak demikian pada perangkat MIDP. Tidak semua perangkat MIDP memiliki koneksi langsung ke Internet dan biasanya tidak mendukung *socket*. Perangkat MIDP juga tidak memiliki *User Interface* yang mendukung tampilan HTML.

Request dan *responses* memiliki dua bagian yaitu *headers* dan *content*. URL yang diketikkan pada *web browser* akan membuat sebuah HTTP *request* dan mengirimnya ke *server*. *Server* mencari file yang diminta dan mengirimkannya melalui HTTP *response*. Header dari *response* mendeskripsikan tentang beberapa hal seperti tipe *web server*, tipe file *response*, panjang dari file *response*, dan informasi lainnya. Isi dari *response* adalah data itu sendiri.

Servlet merupakan salah satu teknologi yang dapat mengolah respon yang bersifat dinamis melalui HTTP. *Servlet* dapat menangani beberapa protokol seperti FTP, SMTP, HTTP. Namun dalam praktiknya hanya server HTTP yang banyak dilayani, khususnya dengan *class* Http*Servlet*.

PERANCANGAN APLIKASI

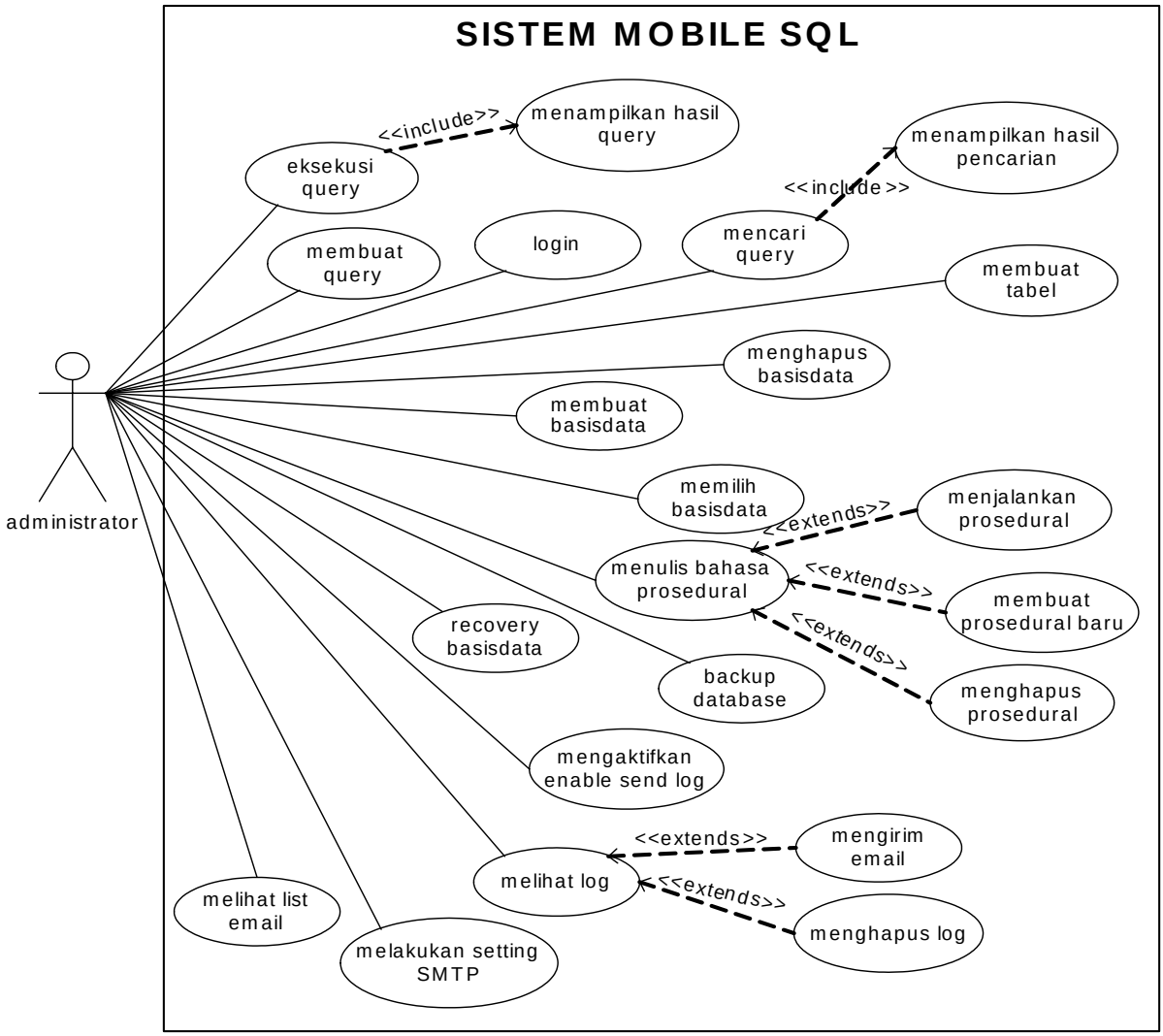
Fitur Aplikasi

Mobile SQL dibuat untuk memungkinkan dilakukannya administrasi basisdata melalui telepon seluler, menghemat waktu administrator basisdata dalam melakukan perawatan, untuk dapat mengetahui perubahan basisdata yang terjadi melalui email, dan dapat menulis bahasa prosedural (*Procedural Language*). Fitur-fitur yang terdapat pada Mobile SQL yaitu:

1. *Login*
   Fitur *login* diperlukan untuk otorisasi pengguna. Fitur *login* mencocokkan informasi pengguna yang ada pada *server* basisdata. Apabila informasi cocok maka pengguna akan diperbolehkan masuk aplikasi. Apabila gagal/tidak cocok maka pengguna tidak diperbolehkan masuk aplikasi.
2. *Query Tool*
   Fitur ini memungkinkan administrator membuat, menyimpan dan menjalankan *query*. Terdapat tiga tipe SQL *Command* yang dapat dijalankan

administrator [4]. Pertama adalah *Data Manipulation Language* (DML). Untuk "*Select*" *Statement*, pengembalian nilai akan ditampilkan pada tabel yang dibuat menggunakan Custom *Item Java ME*. Kedua adalah *Data Definition Language* (DDL). Pada DDL administrator dapat membuat tabel, melakukan perubahan dan menghapus struktur data dari tabel. Ketiga adalah *Data Control Language* (DCL). Pada DCL administrator dapat memberikan, membatasi atau menghapus hak akses *user* terhadap basisdata. Hak Akses yang diberikan dapat berupa *Read Only*, *Update*, *Delete*, *Add*, dan untuk manipulasi struktur basisdata seperti atribut, tabel, indeks.

3. *Query Manager*
   Fitur ini dilengkapi dengan GUI untuk memberi kemudahan bagi administrator dalam mendefinisikan, memanipulasi, dan mendapatkan data. *Query Manager* memiliki beberapa fungsi seperti membuat, modifikasi atau menghapus tabel/*login role*/*group role*. Melihat deskripsi tabel. Membuat *foreign key* dan *indexes* pada tabel. Merubah struktur tabel pada basisdata.
4. *Procedural Editor*
   Fitur ini membantu administrator dalam menulis bahasa prosedural yang dipakai oleh sebuah DBMS sehingga administrator dapat melihat dan menggunakan kode fungsi seperti berikut [4]:
   - *Operator* seperti [], -, ^, AND, OR
   - *Built in Function* seperti SysDate, SUM, AVG, TO_DAT
   - *Key word* seperti IF , THEN, CASE, LOOP, FOR
   - *Exception Level* seperti DEBUG, LOG, INFO
   - *Trigger Procedure Variables* seperti NEW, OLD, TG_NAME, TG_WHEN



Gambar 1. ***Use Case Diagram***

Fungsi lain adalah pemberian *line number* dan *highlight* pada kode fungsi diatas. Seorang administrator dapat menjalankan prosedural, membuat *file* baru, membuka *file*, dan menyimpan *file*.

5. *Backup* dan *Recovery*
   Fitur ini digunakan untuk melakukan *backup* dan *recovery* basisdata.
6. *Preference*
   Pada *Preference* ini, *user* dapat mengaktifkan *Enable send log*. Fungsi *enable send log* adalah: mengirim informasi *log* melalui *email* kepada administrator (lain) secara langsung apabila ada pengguna yang melakukan administrasi basisdata, membuat daftar *email* administrator lain, mengirim *log* melalui *email* kepada administrator lain, membersihkan *log* yang ada pada memori telepon seluler, dan melakukan *setting* SMTP (*Simple Mail Transfer Protocol*).

Desain Aplikasi

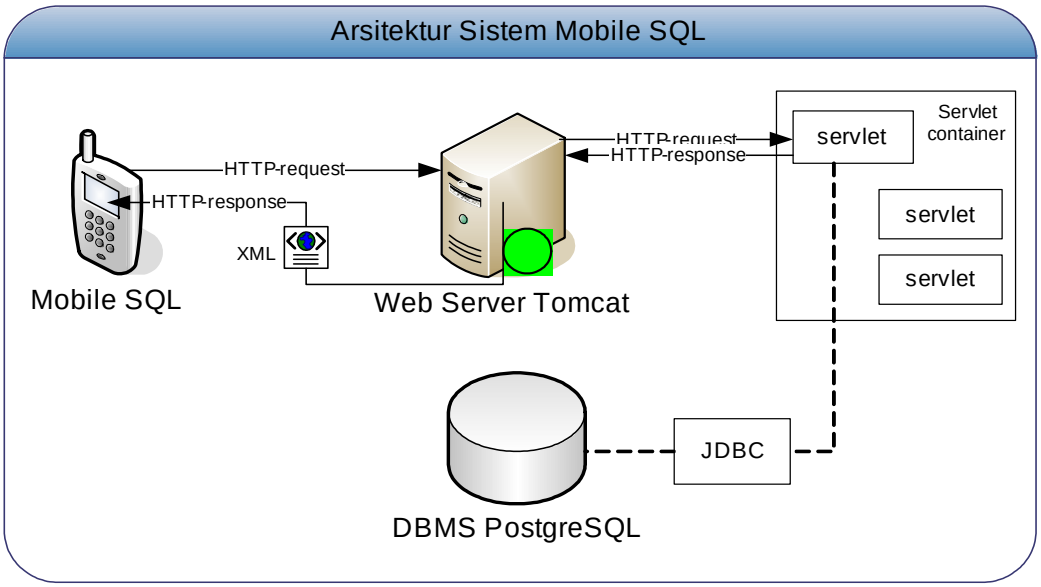
Gambar 1 menunjukkan *use case diagram* yang menggambarkan fitur-fitur diatas. *Use Case Mobile SQL* terdiri dari satu aktor yaitu administrator basisdata. Seorang administator basisdata dapat melakukan beberapa fungsi utama seperti menjalankan *query*, membuat basisdata, membuat *table*, membuat *user*, menulis bahasa prosedural, melakukan *backup* dan *recovery* basisdata. Untuk melihat hasil *query*, administrator harus menjalankan *query* terlebih dahulu. Untuk memilih, membuat, atau menghapus basisdata, administrator harus *login* terlebih dahulu. Status email tergantung pada email yang dikirimkan. "Sukses" untuk email yang berhasil di kirim. "Gagal" untuk email yang tidak berhasil dikirim.

Arsitektur sistem dapat digambarkan seperti diagram pada Gambar 2.
Proses eksekusi *query* adalah sebagai berikut dalam setiap *request* [5]:

- Telepon seluler terinstall Mobile SQL mengirimkan data berupa informasi *user* dan *query* melalui HTTP *request*.
- *Web server* menerima dan melanjutkan data kepada *servlet*.
- *Servlet* akan menerima data dari HTTP *request* sesuai dengan parameter masing-masing.
- *Servlet* mengakses basisdata PostgreSQL melalui JDBC.
- HTTP *response* dibuat dalam format XML.
- Mobile SQL melakukan *parsing* XML dan menampilkan data pada layar telepon seluler.

Salah satu fitur utama dalam Mobile SQL adalah eksekusi *query* pada fitur *Query Tool*. *Query* yang akan dieksekusi dapat dipilih dari *query list* atau langsung eksekusi *query* tanpa menyimpan *query* terlebih dahulu. Eksekusi langsung dapat dilakukan dengan dua cara, pemodelan aktivitas dapat dilihat pada Gambar 3. Pendekatan pertama, pengguna menulis *query* lalu menjalankan *query*. Cara kedua adalah menulis *query*, menyimpan *query*, kemudian menjalankan *query*. Bila berhasil maka aplikasi akan ditampilkan hasil *query*, bila gagal maka aplikasi akan menampilkan *alert*.

Query yang disimpan akan masuk ke dalam memori telepon seluler. *Query* yang yang menampilkan informasi seperti *SELECT* akan ditampilkan pada tampilan telepon seluler. Untuk *Query* yang tidak menampilkan informasi akan ditampilkan status eksekusi berupa sukses atau gagal. Apabila gagal administrator akan diberitahu *error* apa yang terjadi.



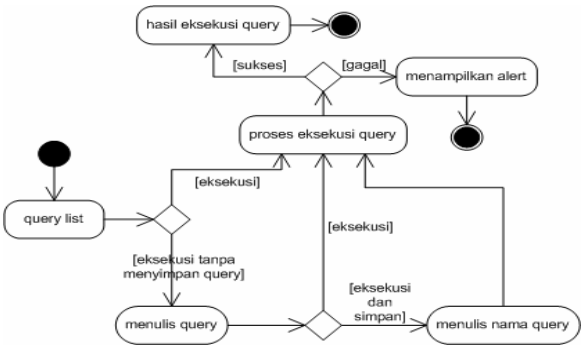
Gambar 2. Arsitektur Sistem

HASIL IMPLEMENTASI

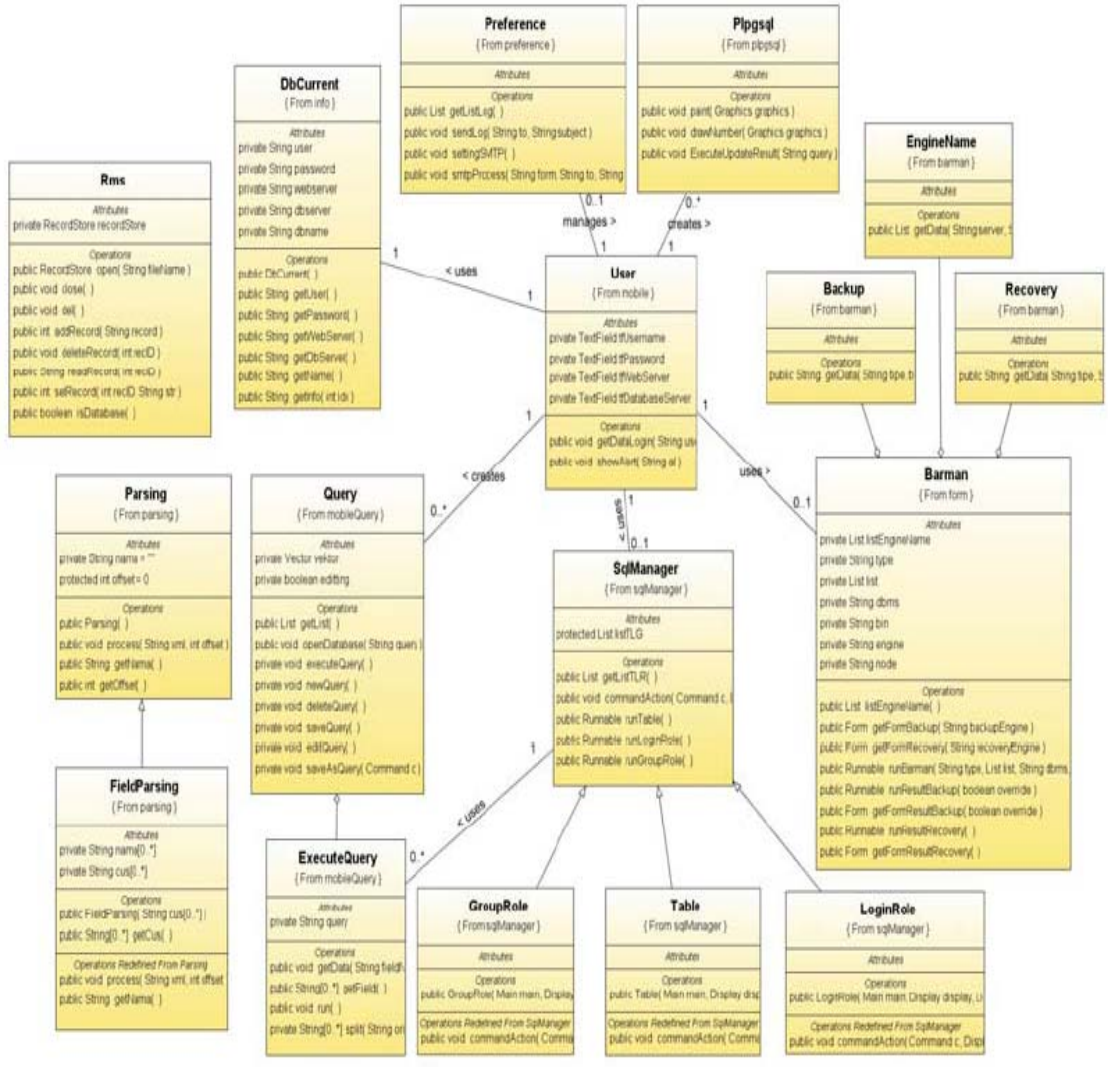
Pengembangan aplikasi telah berhasil diujicoba dengan menggunakan Sistem Operasi Microsoft Windows XP, Java Development Kit 1.6, *Emulator* Wireless Toolkit 2.5.2, *Web Server* Apache Tomcat 6, dan DBMS PostgreSQL 8.2.

Hasil Implementasi berupa aplikasi MIDlet berekstensi *Java Archive* (.jar) yang dapat dijalankan pada telepon seluler berbasis *java*.

Rancangan *Use Case* pada Gambar 1, diimplementasikan sesuai dengan diagram kelas pada Gambar 4.



Gambar 3. Activity Diagram pada Eksekusi *Query*



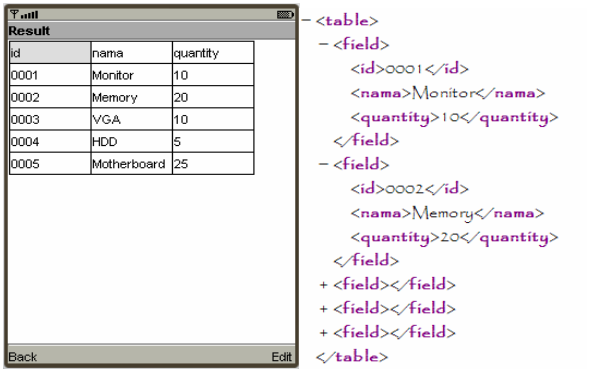
Gambar 4. Kelas Diagram Implementasi

Penjelasan singkat untuk setiap kelas adalah sebagai berikut:

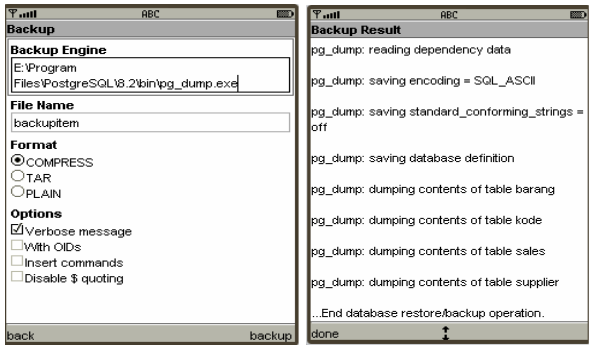
Nama Kelas	Deskripsi Singkat
RMS	Kelas untuk mengelola *record* dalam sebuah tabel pada basisdata.
DbCurrent	Menyediakan informasi mengenai basisdata yang sedang digunakan/ aktif.
User	Kelas untuk mengelola dan validasi pengguna aplikasi Mobile SQL dan database.
Query	Kelas untuk mengelola *query* yang tersimpan dalam sistem Mobile SQL.
Execute*Query*	Kelas untuk memberikan hasil eksekusi *query*.
SQLManager	Kelas untuk menampilkan daftar tabel, role dan login role.
Table	Kelas untuk menambahkan tabel pada server basisdata.
LoginRole	Kelas untuk menambahkan login role pada server basisdata.
GroupRole	Kelas untuk menambahkan group role pada server basisdata.
Barman	Kelas untuk eksekusi backup dan recovery.
Recovery	Kelas untuk mengirim parameter recovery ke dalam web server.
Backup	Kelas untuk mengirim parameter backup ke dalam web server.
Engine	Kelas untuk mengirim parameter *engine backup* dan *recovery Postgre SQL* ke dalam web server.
Plpgsql	Kelas editor dan eksekusi bahasa prosedural PL/pqSQL.
Preference	Kelas yang bertanggung jawab untuk koneksi ke dalam SMTP *server*, daftar email dan *log*.
Parsing	Kelas untuk melakukan parsing XML dari informasi sebagai hasil eksekusi *query*.

Gambar 5 (kiri) adalah tampilan eksekusi *query* pada fitur *Query Tool*. Hasil *query* dapat berupa tampilan tabel atau hanya hasil berupa sukses atau gagal. Hasil *query* ditampilkan dalam bentuk tabel dan dapat diatur sehingga memudahkan administrator untuk melihat melalui scrolling, pergerakan kiri, kanan, atas dan bawah.

Tampilan eksekusi *query* sebenarnya adalah hasil parsing perubahan record yang terjadi dan dijadikan sebagai file xml oleh *servlet*. File xml dapat dilihat pada Gambar 5 (kanan).

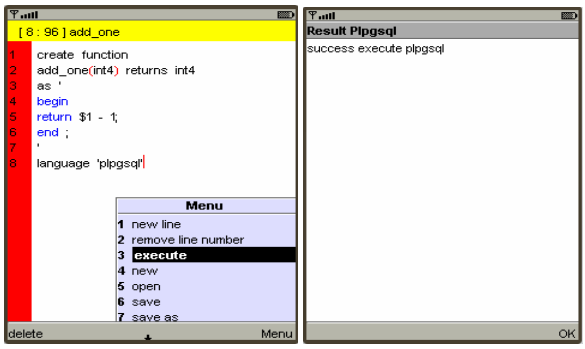


Gambar 5. Hasil Eksekusi *Query*



Gambar 6. Melakukan Backup

Dalam melakukan *backup* ada beberapa hal yang harus di perhatikan diantaranya adalah *BackupEngine* dan *FileName*. Walaupun *BackupEngine* telah ditentukan sebelumnya oleh *user* namun *user* tetap dapat menggantinya. *FileName* tidak perlu diberikan *extensions* karena aplikasi akan memberikan *extensions* sesuai dengan *format backup*.



Gambar 7. Menulis Bahasa Prosedural

Gambar 7 menunjukkan tampilan *text editor* ketika menulis bahasa prosedural. Bila berhasil dijalankan akan menampilkan pesan sukses.

PENGUJIAN APLIKASI

Pengujian aplikasi dilakukan dengan dua cara, yaitu untuk uji fungsionalitas terhadap fitur dan pengujian unit untuk alur program.

Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan untuk setiap fitur yang terdapat dalam Mobile SQL dengan menggunakan emulator dan perangkat telepon selular sehingga penguji tidak perlu mengetahui struktur program. Sebuah fitur program dinyatakan lulus uji apabila data keluaran yang dihasilkan dari skenario pengujian sesuai harapan tanpa melihat bagaimana proses untuk mendapatkan keluaran tersebut.

Tabel 1 adalah contoh hasil pengujian untuk fitur Login.

Tabel 1. Hasil pengujian fitur login

Fitur	Masukan				Keluaran
	User-name	pass-word	web server	database server	
Daftar *user*	root	root	localhost: 8084	localhost	Sukses, *user* berhasil mendaftar
	root	root	332312	localhost	Gagal, *malformed* URL
	root	root	localhost: 8084	abcdefg	Gagal, kesalahan koneksi
			Password		
Login *user*			root		Sukses, *user* berhasil *login*
			abcdefg		Gagal, *password* salah
			User		
Hapus *user*			root		Sukses hapus *user*
			budi		Gagal, *user* tidak ada

Tabel 2 menjelaskan perbandingan uji fungsionalitas menggunakan telepon seluler dan *emulator*.

Tabel 2. Perbandingan uji coba fungsionalitas

	Telepon Seluler	*Emulator*
Perangkat Pengujian	Motorola L-6	*Emulator Wireless Toolkit* 2.5.2
Memori	Relative lebih kecil, untuk pengujian digunakan memori 5 MB.	Sesuai dengan memori komputer
Maksimal *record*	Menangani maksimal 160 *record*	Menangani maksimal 1000 *record*
Pengiriman dan penerimaan data	Lebih lambat	Sangat cepat
Koneksi HTTP	Koneksi yang tidak ditutup menyebabkan *error*	Koneksi yang tidak ditutup tidak menyebabkan *error*

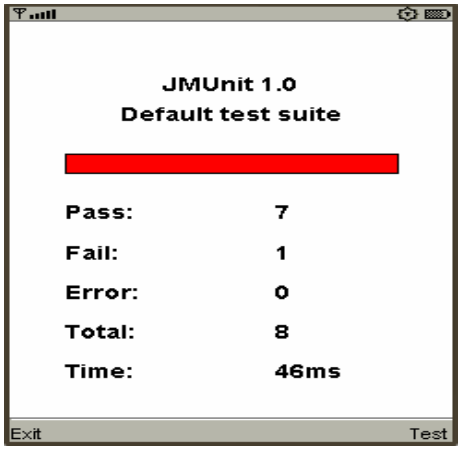
Pengujian Unit

Uji unit dilakukan dengan membuat skenario pengujian dengan perangkat JMUnit (*Java Mobile Unit*). Untuk pengujian unit, penguji harus mengetahui struktur program karena menggunakan *method* yang merupakan bagian terkecil dari program berorientasi obyek. Pada Gambar 8, ditampilkan hasil pengujian dengan JMUnit untuk kelas RMS yang di dalamnya memiliki 8 method, dengan hasil 7 sesuai harapan, dan satu tidak.

Tabel 3 adalah contoh hasil pengujian terhadap kelas RMS, yang dipakai untuk mengelola *record* dalam memori perangkat telepon selular.

Tabel 3. Hasil pengujian kelas RMS

Testing Method	Parameter	Expected Result	Result
addRecord(String record):int	"root½root¼ localhost:8084 »localhost"	1	1
setRecord(int recordId, String str):boolean	1 , "root½root¼ localhost:8084 »localhost"	True	True
isDatabase():boolean	-	True	True
readRecord(int recordId):String	1	erick½erick¼ localhost:8084 »localhost	erick½erick¼ localhost:8084 »localhost
deleteRecord(int recordId):void	1	"deleted"	"deleted"
del():void	-	"delete record"	"delete record"
open(String fileName):RecordStore	"dbuser"	javax.microedition.rms. RecordStore@f828ed68	javax.microeditio n.rms. RecordStore@f828ed68
close():void	-	"close record store"	"ERROR: RecordStore not open"



Gambar 8. Pengujian JMUnit Terhadap Kelas RMS

KESIMPULAN DAN SARAN

Setelah melakukan pengujian terhadap metode *white box* (dengan JMUnit) dan *black box*, penulis menarik beberapa kesimpulan antara lain:

1. Administrator dapat melakukan *maintenance* basisdata melalui aplikasi bergerak dalam bentuk telepon genggam.
2. Administrator dapat mengetahui perubahan yang terjadi di server basisdata melalui email.
3. Aplikasi dapat digunakan untuk menulis bahasa prosedural.
4. Aplikasi berukuran 125 KB, berjalan dengan baik pada ponsel *java-enable* dengan spesifikasi MIDP 2.0 dan CLDC 1.1.
5. Aplikasi menangani *record* dengan maksimal 160 *record*. Keterbatasan ini dipengaruhi oleh memori telepon seluler yang hanya 5 MB.
6. Fitur *backup* dan *recovery* memiliki kekurangan yaitu lokasi *file backup* tidak dapat di-*explore* dan harus ditentukan oleh pengguna. Hal ini disebabkan terutama karena keterbatasan *bandwidth* pada saat akses ke dalam *server*.

Beberapa pengembangan aplikasi yang perlu dilakukan di masa yang akan datang antara lain:

1. Aplikasi mendukung beberapa DBMS sehingga jangkauan penguna menjadi lebih luas.
2. Pengembangan fitur *searching* yang lebih kompleks sehingga hasil pencarian lebih akurat mengingat jumlah data yang dapat meningkat terus menerus.
3. Meningkatkan keamanan proses pengiriman dan penerimaan data antara telepon seluler dan *web server*. Hal ini diperlukan mengingat data-data penting basisdata.

DAFTAR PUSTAKA

1. Sallahudin, M. dan Rosa, A.S. 2006. *Pemrograman J2ME: Belajar Cepat Pemrograman Perangkat Telekomunikasi Mobile*. Bandung: Informatika.
2. Mardiono, T. 2006. *Membangun Solusi Mobile Business dengan Java*. Jakarta: PT Elex Media Komputindo.
3. Knudsen, J. 2003. *Wireless Java Developing with J2ME, Second Edition*. Akses terakhir 13 Maret 2007, dari http://knowfree.net/.
4. Matthew, N. dan Stone, R. 2005. *Beginning Databases with PostgreSQL From Novice to Professional*, Second Edition. Apress. Akses terakhir 24 Juni 2007, dari http://www.pdfchm.com/.
5. Jendrock, E., Ball, J., Carson, D., Evans, I., Fordin, S., dan Haase, K. 2006. *The JavaTM EE 5 Tutorial*, Third Edition: For Sun Java System Application Server Platform Edition 9. Addison Wesley Professional. Akses terakhir 25 Oktober 2007, dari http://www.itstudy8.org/.