PEMANFAATAN *MOBILE DATABASE* UNTUK EFFEKTIFITAS KONEKSI PADA *MOBILE LMS*

Fahri Firdausillah[1], Harisuddin Hakim[2]
[1]*Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka*
E-mail : elfaatta@gmail.com

[2]*Fakultas Ilmu Komputer Universitas Dian Nuswantoro*
E-mail : hanzchu@gmail.com

ABSTRAK

Learning Management System (LMS) banyak dimanfaatkan oleh institusi pendidikan dan juga perusahaan komersial sebagai media pembelajaran yang terintegrasi dengan dokumentasi, administrasi, sekaligus pelaporan. Selain dalam bentuk web application untuk perangkat PC dan laptop, saat ini juga tersedia Mobile-LMS (M-LMS) dalam bentuk web berbasis mobile yang dapat diakses melalui mobile browser dengan menggunakan koneksi internet. Namun demikian saat ini perangkat bergerak seperti Smartphone, PDA, dan ebook reader mempunyai keterbatasan pada kualitas koneksi internet yang tidak stabil dan juga memori yang jauh lebih kecil dari PC. Kondisi ini mengakibatkan pengaksesan M-LMS menjadi terkendala. Paper ini menawarkan sebuah framework untuk M-LMS dengan memanfaatkan mobile database. Pada saat terkoneksi dengan server (online), M-LMS melakukan sinkronisasi antara database server dengan database lokal sehingga memungkinkan penggunaan M-LMS meskipun dalam kondisi offline. Mobile database diimplementasikan dengan menggunakan embedded database yang memerlukan kapasitas relatif kecil untuk mengantisipasi permasalahan keterbatasan media penyimpanan, sehingga alokasi memori lebih efektif. Dari hasil penelitian diketahui, dengan menggunakan framework ini M-LMS dapat digunakan dengan lebih fleksibel, karena pengguna tidak harus selalu online untuk mendapatkan data dari server. Selain itu, karena M-LMS hanya perlu mengakses database lokal, performa aplikasi juga meningkat dan pemanfaatan sumber daya menjadi lebih efektif.

Kata kunci : Mobile Database, M-LMS, Sinkronisasi

1. PENDAHULUAN

Learning Management System (LMS) merupakan sebuah system pembelajaran berbasis *e-Learning*, yang memiliki berbagai macam fasilitas–fasilitas untuk mendukung proses pembelajaran, seperti forum diskusi, enslikopedia mini, kuis, workshop, survey, dan masih banyak lainnya. Tentunya hal ini juga dilengkapi dengan dokumentasi, laporan, dan evaluasi guna mengetahui sejauh mana proses belajar tersebut telah berjalan [2]. Fasilitas yang tersedia pada LMS memungkinkan pengguna untuk saling bertukar ilmu (*knowledge sharing*) yang merupakan aspek penting dalam metode pendidikan modern [3].

Selain itu, salah satu elemen dari pembelajaran di abad 21 adalah proses transfer ilmu dapat dilakukan dimana saja dan kapan saja. Proses belajar yang dulunya hanya dilakukan di dalam kelas, saat ini juga dilakukan di luar kelas, bukan hanya menggunakan *textbook* namun juga mencari pengalaman melalui interaksi dengan lingkungan sekitar. Hal ini sangat mendukung seseorang untuk terus mengembangkan kemampuan berdasarkan potensi yang dimiliki, karena mereka memainkan peran sebagai pihak yang belajar sekaligus menjalankan kewajibannya untuk mengontrol proses belajar mereka [1].

Perangkat bergerak seperti smartphone dan PDA sekarang banyak digunakan bukan hanya untuk alat komunikasi, namun juga berperan sebagai komputer mini. Banyak sekali aplikasi yang bisa dipasang pada perangkat bergerak tersebut untuk mendukung mobilitas dalam belajar (M-Learning). Berdasarkan kemampuan konektivitasnya ada tiga jenis aplikasi yang dapat dipasang pada perangkat bergerak, yaitu aplikasi Online, Offline, dan Hybrid. Aplikasi offline menawarkan akses yang cepat dan lebih handal, karena terinstall di dalam perangkat bergerak itu sendiri dan tidak memerlukan koneksi internet. Namun aplikasi offline ini juga mempunyai kelemahan pada data yang tidak dapat diperbaharui (*up to date*) dengan mudah. Untuk mengupdate data atau aplikasi, pengguna harus meng-*install* versi terbaru secara manual.

Selain itu aplikasi offline juga memiliki kelemahan lain yaitu tidak adanya interaksi antara guru dan murid yang merupakan element vital dalam sebuah pembelajaran [3].

Di lain pihak, aplikasi online menawarkan pembaharauan data yang lebih mudah. Hal ini karena aplikasi tersebut terhubung langsung dengan server yang memungkinkan untuk di perbaharui tiap waktu. Tetapi pemrosesan data yang lama dan tidak stabilnya koneksi menyebabkan hal tesebut menjadi sebuah kendala tersendiri dalam pengembangan aplikasi berbasis online. Sedangkan aplikasi hybrid menggabungkan keduanya, terpasang dalam perangkat bergerak dan mampu mengakses data yang ada pada server.

Pada makalah ini penulis dikembangkan sebuah *Mobile LMS* (MLMS) *Framework* untuk mengoptimalkan proses transfer data, dan memodelkan sebuah system yang dapat diakses secara optimal baik secara offline ataupun online. Framework yang dibangung memungkinkan MLMS dapat diakses secara *offline* dan *online* dengan memanfaatkan *mobile database*. Didalam framework tersebut, mobile database ditanam pada perangkat bergerak di sisi pengguna untuk meyimpan data tertentu dari server. Database yang ditanam tersebut mengandung kerangka kecil dari system tersebut, sehingga tidak memerlukan banyak ruang dalam media penyimpanan perangkat bergerak. Setelah semua data yang dibutuhkan tersimpan pada database lokal, MLMS tidak perlu terhubung dengan server untuk mendapatkan data dari server. Cukup dengan mengakses database lokal, aplikasi tersebut bisa dijalankan layaknya dalam kondisi online. Dalam periode tertentu, pengguna juga dapat melakukan sinkronisasi database lokal dengan database *server*. Selain untuk memperbaharui data yang ada pada database lokal, sinkronisasi juga berfungsi untuk mentransfer perubahan data yang dilakukan pada saat MLMS bekerja dalam keadaan *offline*, ke dalam database utama yang ada pada sisi *server*.

Susunan paper ini pada bab selanjutnya adalah sebagai berikut (2) Latar Belakang teknologi dan framework MLMS yang ada saat ini (3) Arsitektur Sistem menjelaskan landasan kerja sistem dari sisi server dan klien (perangkat bergerak) (4) Algorithma sinkronisasi yang digunakan akan dijelaskan pada bagian ini (5) Pembahasan tentang alur kerja sistem secara keseluruhan (6) dan yang terakhir adalah kesimpulan dan rencana penelitian selanjutnya yang terkait dengan paper ini.

2. LATAR BELAKANG

Ada tiga jenis aplikasi *mobile* yang bisa diterapkan pada pengembangan Mobile LMS yaitu *mobile web*, *platform independent mobile application*, dan *native mobile application*. *Mobile web application* mempunyai kelebihan mudah diakses karena tidak memerlukan proses instalasi, hanya cukup menggunakan mobile browser yang sudah tersedia pada kebanyakan mobile device [4]. Contoh *platform independent* adalah aplikasi yang dibuat dengan J2ME [5] atau Flash Lite [6], aplikasi jenis ini dapat diinstal di mobile device manapun asal memiliki *runtime*-nya. Aplikasi jenis ini biasanya memiliki keterbatasan dalam pemanfaatan mobile resource. Sedangkan Native mobile application adalah aplikasi yang dibangun khusus untuk suatu platform tertentu. Perangkat lunak jenis ini berjalan relatif lebih cepat dibandingkan dua jenis aplikasi yang lain, dan juga dapat sepenuhnya memanfaatkan berbagai sumberdaya dan fitur yang tersedia pada perangkat bergerak.

Saat ini ada banyak Learning Management System (LMS) yang menyediakan versi mobile, diantaranya MoMo (Mobile Moodel), MLE (*Mobile Learning Engine*), dan DRONA (hanya untuk Blackberry). Versi mobile yang tersedia kebanyakan dalam bentuk mobile web application yang hanya dapat diakses jika tersedia koneksi internet. Paper ini mengenalkan framework MLMS berbasis applikasi lokal baik berupa *platform independent application* maupun *native application* yang mampu penyimpanan database LMS pada perangkat bergerak secara lokal.

2.1 Penelitian Terkait

Sultana dan Ishrat mengembangkan E-School yang merupakan e-Learning berbasis web service. E-School mampu mentransformasikan *web application* yang hanya dapat diakses oleh *browser* ke dalam bentuk yang lebih fleksibel dan dapat diakses juga oleh perangkat lunak yang lain [7]. Selain itu Forment dan Guererro juga mengembangkan pengimplementasian dari Moodle dengan nama MoodlBile yang memungkinkan pengaksesan Moodle LMS secara offline dengan mengembangkan aplikasi berbasis J2ME dan J2MEMicroDB untuk penyimpanan [8]. Perbedaan MoodlBile dengan framework ini adalah, secara khusus MoodlBile hanya diimplementasikan dengan menggunakan J2ME karena mobile database yang digunakan hanya dapat diakses menggunakan aplikasi J2ME. Sedangkan framework pada paper ini dapat digunakan

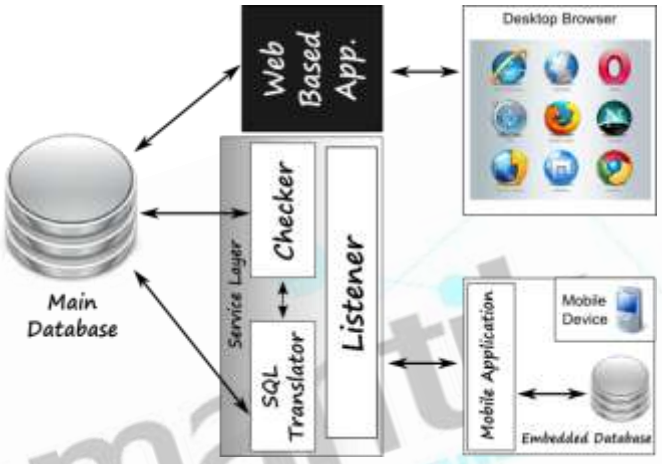
secara lebih umum sehingga lebih fleksibel untuk diimplementasikan di berbagai macam local mobile application termasuk Android, BlackBery, iPhone, dan juga *Java Enabled mobile device* secara umum.

3. ARSITEKTUR SISTEM

Arsitektur pada sistem MLMS ini terbagi dalam dua bagian penting, yaitu sisi server dan sisi klien yang secara umum dapat dilihat pada Gambar 1. Sisi server bertanggung jawab untuk menerima request dari klien, mengecek kemungkinan perubahan pada database server, melakukan sinkronisasi dan mengirimkan hasilnya pada klien. Sedangkan aplikasi sisi klien berperan sebagai antarmuka yang berinteraksi langsung dengan pengguna.

Meskipun konsentrasi makalah ini adalah pada mobile client, namun seperti yang terlihat pada arsitektur Gambar 1, juga terdapat aplikasi berbasis web desktop yang disediakan. Ini ditujukan untuk memfasilitasi pengguna yang ingin mengakses dengan komputer desktop, yang mempunyai tampilan layar yang lebih besar dan papan kunci yang lebih nyaman digunakan.

Gambar 1. Arsitektur yang Ditawarkan untuk MLMS



3.1 Service Layer

Service layer terletak pada server dan bertugas untuk menerjemahkan data yang ditransfer oleh client, kemudian melakukan pengecekan pada database apakah ada perubahan atau penambahan data terbaru, dan juga bertanggung jawab mentransferkan hasilnya pada client dengan bentuk yang dapat dipahami oleh klien. Service layer dapat dibangun menggunakan bahasa pemrograman scripting seperti PHP, JSP, ASP, Ruby, dan lain sebagainya. Bahasa pemrograman tersebut dapat memanipulasi inputan dari client dan menerjemahkan ke dalam bahasa query (SQL) yang dapat langsung dieksekusi oleh database.

Service layer terdiri dari 3 macam kompenen sebagai berikut:

- **Listener** berfungsi untuk menerima input data dari klien dalam bentuk JSON, data yang diterima dari klien adalah data counter, table, dan code_part yang digunakan untuk sinkronisasi serta update data yang dilakukan pada sisi klien selama menggunakan MLMS pada mode offline. Penjelasan lebih rinci tentang proses sinkronisasi pada sub bab
- **Checker** bertugas untuk melakukan pengecekan pada database server apakah ada perubahan data terbaru pada database utama, berdasarkan kebutuhan data yang dikirimkan oleh klien.
- dan **SQL-Translator** yang berperan untuk mengubah data yang diterima dari klien dalam bentuk JSON ke dalam bahasa query yang siap dijalankan oleh database. Dan juga sebaliknya mengubah hasil query dari database kedalam bentuk JSON untuk dapat dikirimkan ke aplikasi klien.

3.2 Database Sisi Server

Database yang digunakan pada sisi server adalah RDBMS yang sering digunakan pada aplikasi *client-server* seperti MySQL, Oracle, MsSQL, PostgreSQL, dan lain sebagainya. RDBMS mempunyai kelebihan manajemen konsistensi data dengan menggunakan normalisasi dan juga pengaksesan data yang relatif mudah dengan menggunakan bahasa query terstruktur (SQL).

Database sisi server berperan sebagai media penyimpanan sentral untuk semua data Learning Management System, termasuk data pengguna, manajemen kelas, mata pelajaran, group, dan lain sebagainya. Klien juga akan melakukan update dan sinkronisasi pada database sentral ini.

3.3 Embedded Mobile Database

Perangkat bergerak mempunyai keterbatasan pada kapasitas perangkat penyimpanan, sehingga kita tidak bisa menggunakan DBMS sebagaimana pada sisi server. Sebagai gantinya kita menggunakan embedded yang mempunyai *small footprint.* Maksudnya adalah database ini tidak memerlukan server khusus yang terpisah untuk menjalankan fungsinya. Fungsionalitas database sudah di-*embed*-kan pada aplikasi yang berjalan sehingga tidak memerlukan database server lagi.

Kandidat terbaik *embedded relational database* untuk mengimplementasikan framework ini adalah SQLite. Database ini mempunyai *footprint* yang sangat kecil yaitu kurang dari 324KB untuk fungsional penuh, atau kurang dari 190KB dengan menghapuskan fungsional yang kurang diperlukan [9]. Selain itu SQLite juga sudah banyak terintegrasi dengan sistem operasi *mobile device* modern, seperti Symbian, iPhone, Android, dan Windows Mobile. Sayangnya J2ME masih belum dapat mensupport SQLite, sehingga untuk aplikasi yang dibangun dengan menggunakan J2ME harus mencari alternatif lain.

Salah satu solusi embedded database untuk J2ME adalah Java J2MEMicroDB yang dikembangkan oleh Alier. J2MEMicroDB merupakan persistence framework yang diterapkan pada J2ME, database ini memungkinkan aplikasi J2ME untuk menyimpan data pada perangkat bergerak dan dukungan terhadap operasi relational database, dan juga terdapat mekanisme untuk Backup/Restore [10]. Solusi yang lain adalah menggunakan JavaDB (sebelumnya dikenal dengan Derby), sayangnya database ini mempunyai footprint yang cukup besar untuk ukuran perangkat bergerak, yaitu sekitar 2,5MB.

Di samping itu ada juga mobile database seperti OracleLite [11], SysbaseAnywhere [12], dan Microsoft SQL CE (*Compact Edition*) [13]. Selain mempunyai *small footprint*, database tersebut juga sudah dilengkapi dengan fungsi sinkronisasi otomatis. Sayangnya produk database tersebut tidak gratis dan juga bersifat sumber tertutup, sehingga kurang cocok untuk diimplementasikan pada LMS yang bersifat *Free* dan *Open Source* seperti Moodle.

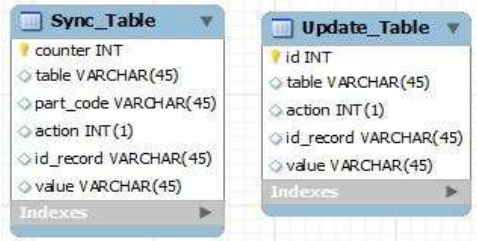
4. ALGORITHMA SINKRONISASI

Proses sinkronisasi mempunyai peranan yang sangat penting dalam framework ini, dengan adanya sinkronisasi data, MLMS dapat diakses secara offline dengan menggunakan embedded database. Ada beberapa algorithma sinkronisasi yang dapat digunakan pada database terdistribusi, diantaranya [14] dan [15]. Pada paper ini, akan dijelaskan model sinkronisasi sederhana yang dapat diimplementasikan.

4.1 Penambahan Skema Database

Untuk melakukan sinkronisasi perlu ditambahkan sebuah tabel pencatat perubahan pada sisi server yang menyimpan semua aktivitas perubahan yang ada pada database. Sedangkan pada sisi klien, ditambahkan 2 tabel. Sync_Table untuk menyimpan data sinkronisasi yang sukses dijalankan, dan Update_Table untuk menyimpan perubahan yang dilakukan pada tabel klien selama MLMS berjalan pada offline mode. Skema tabel dapat dilihat pada Gambar 2.

Gambar 2: Tabel bantu untuk sinkronisasi dan update



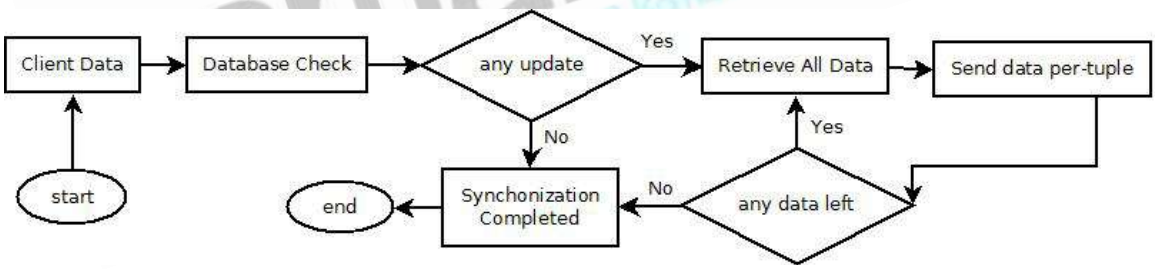
Pada Gambar 2 terlihat ada 6 field yang dibutuhkan untuk menyimpan data perubahan.

- Counter berfungsi sebagai penanda perubahan yang terakhir di jalankan
- Table menunjukkan pada tabel apa perubahan terjadi
- Part_code dapat digunakan untuk mengecek apakah klien tertentu benar memerlukan update perubahan tersebut. Misalkan klien A hanya enroll pada mata pelajaran dengan kode AB dan AC, maka semua perubahan pada tabel dengan part_code selain itu akan diabaikan untuk menghemat bandwith dan juga mempercepat proses.
- Action memiliki tiga kemungkinan nilai yaitu 1 untuk penanda penambahan record (insert), 2 untuk penghapusan record (delete), dan 3 untuk perubahan record (update).
- Id_record merupakan id unik pada tiap tabel tempat di mana perubahan terjadi.
- Value adalah konten perubahan dengan skema sebagaimana ditampilkan Tabel 1.

Tabel 1: Pola penulisan value untuk insert update dan delete

Action	Value
Insert	value 1||value 2||value 3.....||value n
Update	field 1=value 1||field 2=value 2||field n = value n
Delete	NULL

Kemudian dalam melakukan update data dan sinkronisasi, MLMS menggunakan algorithma sebagaimana ditampilkan pada Gambar 3. Pertama Listener menerima data dari klien yang berisi counter juga tabel dan part_code yang diperlukan klien. Selanjutnya checker akan melakukan checking pada database apakah ada counter yang lebih tinggi pada database dengan nama tabel dan part_code yang sama, yang berarti ada perubahan data terbaru di database sisi server. Jika tidak ada maka proses sinkronisasi selesai. Jika ada maka service layer akan mengirimkan data perubahan satu persatu pada klien sampai data habis, sehingga jika pada pada pertengahan proses ada gangguan maka proses sinkronisasi tidak harus mengulan dari awal.



Gambar 3: Algorithma sinkronisasi

Sedangkan untuk update perubahan yang dilakukan pada sisi klien relatif lebih sederhana, karena tidak ada proses checking pada saat sinkronisasi. Cukup mengirimkan data ke server, kemudian listener akan meneruskan pada SQL-Translator, dan setelah diubah dalam bentuk SQL langsung dapat dieksekusi oleh *main* database pada server.

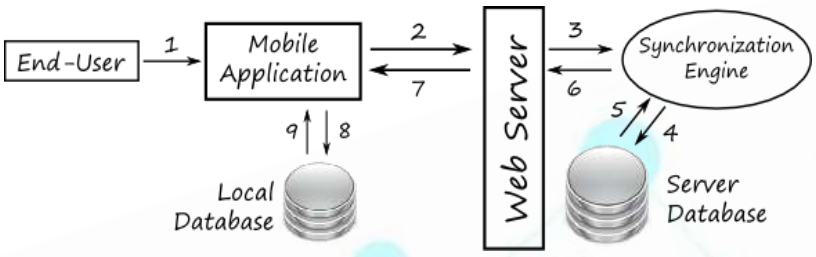
4.2 Mengatasi Konflik Data

Kim menjelaskan untuk menanggulangi duplikasi data pada proses singkronisasi terdapat 4 langkah yaitu klasifikasi konflik, identifikasi perubahan, pendeteksian konflik, dan penanganan konflik. Ada tiga kemungkinan konflik yang terjadi yaitu penginputan data dengan index yang sama (*insertion conflict*), pengubahan *record* yang sama oleh dua pengguna atau lebih (*update conflict*), dan penghapusan *record* yang hendak diakses (di-*update*) oleh user yang berbeda (*deletion conflict*). Saat ini belum ada penanganan konflik yang cukup efektif kecuali dengan menggagalkan perubahan data yang menimbulkan inkonsistensi

[15]. Untungnya pada konteks MLMS ini kemungkinan terjadi konflik sangatlah minim, karena masing-masing pengguna hanya dapat mengubah data mereka sendiri. Kemungkinan terjadi konflik hanya pada MLMS yang menyediakan fitur wiki atau glosarium yang dapat diperbaharui oleh semua pengguna. Dalam kasus tersebut, maka metode penggagalan query yang menimbulkan inkonsistensi data diterapkan.

5. ALUR KERJA

Gambar 4 menunjukkan alur kerja dari system yang ada pada aplikasi MLMS, point pertama menjelaskan tentang interaksi dari *End User* dengan aplikasi MLMS. Pada point ini pengguna hanya berinteraksi dengan aplikasi MLMS secara lokal. Selanjutnya, ketika MLMS melakukan koneksi ke server sebagaimana ditunjukkan point kedua, aplikasi MLMS akan melakukan *request* ke web server untuk menjalankan proses sinkronisasi antara database lokal dan database *server*. Ketika reply dari server didapat, aplikasi MLMS langsung mengirimkan data berupa data – data apa saja yang berubah ketika aplikasi tersebut offline ke database server (terlihat pada point 3). Disini proses sinkronisasi terjadi, data–data pada database lokal dicocokkan dan disinkronisasikan dengan database server.



Gambar 4: Alur kerja sistem

Setelah proses sinkronisasi terjadi, data hasil dari proses sinkronisasi dimasukkan ke database server sebagai acuan data yang baru dari user (point 4). Setelah itu server mengirimkan data berupa update data terbaru yang dibutuhkan oleh user (point 5). Sebelum dikembalikan ke aplikasi MLMS, data dari server di sinkronisasi lagi untuk proses pencocokan dengan database lokal. Setelah itu, hasil dari proses sinkronisasi dikirimkan ke aplikasi MLMS sebagai sumber acuan yang baru pada database lokal (point 6). Sesampainya di sisi lokal (point 7), aplikasi menyimpan data yang didapat dari server ke database lokal (point 8). Setelah itu database lokal menerapkan data – data yang didapat dari database server untuk digunakan pada aplikasi MLMS (point 9). Pada keseluruhan proses tersebut, terlihat bahwa koneksi terjadi hanya ketika awal dan akhir dari keseluruhan proses tersebut, sehingga hal ini menghemat pemakaian bandwith yang menjadi kendala di kebanyakan aplikasi mobile. Dilain sisi, data yang dikirimkan oleh server sudah dalam bentuk data yang sesuai dengan database lokal atau telah disinkronisasikan dan diterjemahkan dengan format database lokal, sehingga hal ini mempermudah aplikasi MLMS untuk menerima data dari server dan menerapkannya pada system lokal, hal ini juga merupakan pengoptimasian agar pada sisi lokal tidak terlalu berat dalam melakukan proses ketika aplikasi tersebut terhubung secara online.

6. KESIMPULAN DAN SARAN

Framework yang ditawarkan pada paper ini dapat menghemat bandwith koneksi yang digunakan untuk mengakses MLMS dengan menyimpan data-data yang dibutuhkan oleh aplikasi pada media penyimpanan lokal suatu perangkat bergerak. Sehingga untuk menggunakan aplikasi MLMS pengguna tidak harus selalu terkoneksi dengan server (*online*), cukup mengakses data yang ada pada *mobile database*. Sedangkan cara untuk menjaga data yang ada pada media penyimpanan lokal selalu *up to date* adalah dengan melakukan sinkronisasi dengan data yang ada pada database utama. Selain itu sinkronisasi juga dimaksudkan untuk menyimpan perubahan data yang dilakukan pada perangkat bergerak selama kondisi *offline*, sehingga interaktifitas antar satu pengguna dengan pengguna yang lain tetap terjaga.

Kelanjutan dari penelitian ini adalah implementasi framework dengan memodifikasi LMS yang ada seperti Moodle dan Claroline, dan juga membuat *protoype* aplikasi untuk perangkat bergerak berbasis *native mobile application* dan *platform independent mobile application*. Selain itu karena saat ini kebanyakan

materi yang disimpan dalam LMS masih *desktop oriented* yaitu dalam bentuk slide dan pdf, diperlukan penelitian lebih lanjut tentang jenis dokumen yang tepat untuk perangkat mobile, agar lebih mudah diakses pada perangkat mobile yang mempunyai ukuran tampilan layar yang terbatas.

DAFTAR PUSTAKA

[1] Jianmei Wang and Yongtang Li. *Flexible Teaching and learning to Life-long Education for 21 Century Undergraduates.* IEEE 2nd International Conference on Education Technology and Computer, July 2010.

[2] Nur' Aini, binti Abdul Rashid, Bin Majid Omar, and Chow Shiao yen, *E-learning Management System for Secondary School in Malaysia.* International Conference on The Challenge of Learning & Teaching in a Brave New, Hatyai, Thailand, 2002.

[3] Geddes S.J., "*Mobile Learning in the 21st century: benefit for learners, the knowledge tree", 2004.* http://knowledgetree.flexiblelearning.net.au/edition06/download/Geddes.pdf, (accessed 24 Februari 2011)

[4] Mehta Nirav, "*Mobile Web Development"*, Packt Publishing, Mumbai: 2008.

[5] Tauber Daniel A., What's J2ME? http://tim.oreilly.com/pub/a/onjava/2001/03/08/J2ME.html diakses pada 20 Maret 2011

[6] http://www.adobe.com/ap/products/flashlite/ diakses pada 20 Maret 2011

[7] Sultana Afroza and Sultana Ishrat, "*E-School: A Web-Service Oriented Resource Based E-Iearning System*", proceeding of International Conference on Networking and Information Technology, 2010.

[8] Forment Marc Alier, Guerrero Mª José Casany, "*Moodlbile: Extending Moodle To The Mobile On/Offline Scenario*" proceeding of El departament d'Enginyeria de Serveis i Sistemes d'Informació , 2008.

[9] Junyan Lv, Shiguo Xu, Yijie Li, "Application Research of Embedded Database SQLite", proceeding of International Forum on Information Technology and Applications, 2009.

[10] Alier Marc, Casado Pablo, Casany Maria José, "*J2MEMicroDB: An Open Source Distributed Database Engine for Mobile Applications*", proceeding of the 5th international symposium on Principles and practice of programming in Java, New York:2007.

[11] http://www.oracle.com/technetwork/database/database-lite/overview/index.html diakses pada 5 Maret 2011

[12] http://www.sybase.com/products/databasemanagement/sqlanywhere diakses pada 5 Maret 2011

[13] http://www.microsoft.com/sqlserver/2005/en/us/overview.aspx diakses pada 5 Maret 2011

[14] Choi Mi-Young, Cho Eun-Ae, Park Dae-Ha, Moon Chang-Joo, Baik Doo-Kwon, "*A Database Synchronization Algorithm for Mobile Devices*", journal of Consumer Electronics, IEEE Transactions, vol 56 , issue 2, pp 392 - 398 , 2010.

[15] Kim Sang-Wook, "*Synchronization in an Embedded DBMS Environment*", journal of International Journal of Computer Science and Network Security, VOL.6 No.7A, 2006.