JURNAL TEKNOLOGI TECHNOSCIENTIA ISSN: 1979-8415
Vol. 5 No. 1 Agustus 2012

REPLIKASI UNTUK MENINGKATKAN KINERJA DAN KETERSEDIAAN DATA (STUDY KASUS SISTEM INFORMASI AKADEMIK)

Joko Triyono[1]

[1]Jurusan Teknik Informatika, Institut Sains & Teknologi AKPRIND, Yogyakarta

Masuk: 3 Maret 2012, revisi masuk : 15 Juni 2012, diterima: 2 Juli 2012

ABSTRACT

This study aims to find the algorithms for designing a distributed database architecture with replication method that will be implemented on an academic database. From this study it is hoped that will get a distributed database architecture that can cope with the availability of data on academic information system. By using a distributed database an application can provide information to its end users to retrieve data from the sources closest to the network from the end user transactions.This design was developed and built using the MySQL database type InnoDB. The design was focused in managing a distributed database using the replication method. Transaction processing was only done to the master, while the slave was only a mirror. Data that has been transacted by the slave will be synchronized with the two methods of automatic and manual by following the pattern of events in the academic system. Master was put on the intranet so that the area will ensure the safety of its data, while the slave was spread to multiple database servers, both intranet and public. Web applications that use the data base was placed disuatu domains on the Internet. The results of this study indicate that by doing the exact distribution of the database. With this method an information system is not necessarily connected to a database, but sufficiently connected to the nearest slave database of network users, so in addition to reducing the server load will also further increase the speed in obtaining data and will impact on the confidence level stack holder.

Keywords: *database, innoDB, distribution, replication*

INTISARI

Penelitian ini bertujuan mencari algoritma-algoritma untuk merancang arsitektur database terdistribusi dengan metode *replikasi* yang akan diimplementasikan pada database akademik dan diharapkan didapatkan sebuah arsitektur database terdistribusi yang bisa mengatasi ketersediaan data pada sistem informasi akademik. Dengan menggunakan database terdistribusi tersebut maka sebuah aplikasi bisa memberikan informasi kepada *end user*-nya dengan mengambil dari sumber data yang terdekat dengan *network* dari melakukan transaksi. Rancangan ini dikembangkan dan dibangun menggunakan basis data MySQL berjenis InnoDB. Rancangan difokuskan dalam mengelola database terdistribusi menggunakan metode Replikasi. Proses transaksi hanya dilakukan terhadap master, sedangkan slave hanya akan menjadi sebuah mirror. Data yang telah ditransaksikan akan dilakukan sinkronisasi oleh slave dengan dua metode yaitu otomatis dan manual dengan mengikuti pola kejadian dalam sistem akademik. Master diletakkan pada area intranet sehingga akan menjamin keamanan datanya, sedangkan slave disebarkan ke beberapa server database, baik intranet maupun public. Aplikasi Web base yang menggunakan data tersebut diletakkan di suatu domain di internet. Dengan metode ini sistem informasi tidak harus selalu terkoneksi ke sebuah database, tetapi cukup dikoneksikan ke database slave terdekat dari network pengguna, sehingga selain mengurangi beban server juga akan lebih meningkatkan kecepatan dalam memperoleh data dan akan berimbas pada tingkat kepercayaan stack holder.

Kata kunci: basis data, innoDB, terdistribusi, replikasi

[1]zainjack@gmail.com

PENDAHULUAN

Kemudahan dalam mendapatkan pelayanan dan atau informasi merupakan salah satu faktor yang sangat penting dewasa ini. Terutama informasi dan atau pelayanan terhadap konsumen. Dalam dunia perguruan tinggi, konsumen yang sangat berkompeten dalam hal ini adalah mahasiswa dan orang tua mahasiswa.

Dengan kemajuan Teknologi komunikasi dan informasi yang bisa dikatakan telah merata diseluruh pelosok negri, sudah menjadi hal yang wajib bagi pengelola perguruan tinggi untuk menyediakan informasi dan layanan yang bisa diakses menggunakan teknologi komunikasi dan informasi yang ada tanpa memperhatikan letak geografis dan teknologi yang dipakai. Faktor keterbatasan teknologi yang dimiliki sering dijadikan alasan untuk meminta kebijaksanaan pada penyelenggara pendidikan, selain itu faktor keamanan data juga sering dijadikan alasan pihak penyelenggara pendidikan untuk tidak mem*publish* data yang dimiliki. Yang akan berimbas kepada ketidak konsistenan antara pelaku dan kebijakan yang telah ditetapkan yang akhirnya akan menimbulkan ketidak percayaan *stake holder* terhadap penyelenggara pendidikan tersebut dan akan berimbas kepada menurunnya minat *stake holder* terhadap penyelenggaraan pendidikan. Lokasi dari *stake holder* saat melakukan transaksi melalui media internet akan terpetakan *networking*-nya pada beberapa kelompok *network*, seperti *intranet*, *internet iix, internet international*, *internet inherent, internet jardik-nas* dan lain-lain.

Melihat kenyataan diatas, maka penelitian ini bertujuan mengembangkan pemanfaatan Replikasi untuk dapat meningkatkan kinerja dan ketersediaan data pada Sistem Informasi Akademik dan menemukan algoritma-algoritma baru yang bisa diimplementasikan.

Siklus transaksi dalam sistem informasi akademik dari tahun ke tahun adalah selalu mirip atau bahkan bisa dikatakan sama, yaitu dari mahasiswa mendaftar, melakukan pengisian krs, melaksanakan kuliah dan praktikum, mendapatkan nilai, akhirnya yudisium akhir untuk kelulusan dan menjadi alumni. Siklus tersebut akan selalu berulang-ulang untuk periode tertentu. Dengan kejadian tersebut maka sebetulnya perubahan data akan mengikuti pola tertentu untuk masing-masing proses dan tidak selalu *real-time*, artinya ada beberapa proses yang memiliki usia 1 semester, 1 tahun, bahkan ada proses yang memiliki usia 1 kali selama mahasiswa kuliah.

Batasan Masalah dalam penelitian ini adalah merancang sebuah arsitektur database terdistribusi dengan metode menggunakan *replikasi* untuk dapat meningkatkan Kinerja dan Ketersediaan Data pada Sistem Informasi Akademik di Institut Sains dan Teknologi AKPRIND Yogyakarta baik dari sisi penyelenggara pendidikan maupun dari sisi *stake holder*. Dengan kondisi kampus yang terbagi menjadi 3 lokasi, yaitu kampus 1 (pusat), kampus 2 dan kampus 3, dengan sentral transaksi di lakukan di kampus 1 (pusat), serta pelayanan terhadap user dari luar (Internet). Sehingga jika seseorang melakukan akses terhadap sistem akademik melalui jalur intranet, maka akan diberikan database yang ada di intranet terdekat (dalam satu group network), yang artinya akan memberikan kenyamanan kepada *end user* tersebut karena data akan lebih cepat terakses, sedangkan *end user* dari internet akan dilayani database yang ada di Server Public.

Triyono (2010) tentang Proses Implementasi Sistem Database Terdistribusi Dengan Metode *Partial Replica* (Studi Kasus : Pelaporan Hasil Penghitungan Suara di DPW PKS DIY) yang telah menghasilkan beberapa kesimpulan antara lain : hasil penelitian ini menunjukkan bahwa distribusi basis data menggunakan metode Partial Replica akan meningkatkan kinerja dan ketersediaan data pada slave, yang berdampak kepada kebutuhan hardware yang harus disediakan. Dengan metode ini suatu sistem informasi tidak harus selalu *online*, cukup menggunakan *semi-online* saja.

Gilfillan (2004) pada *Database Journal* menyatakan bahwa replikasi memungkinkan Anda untuk mengambil satu *database*, membuat salinan persis di *server* lain, dan menetapkan satu dari mereka (*slave*) untuk mengambil semua

update dari yang lain (*master*). *Slave* membaca *log biner master*, yang menyimpan semua pernyataan yang mengubah *database*, dan mengulangi ini di database-nya, menjaga dua sinkron yang tepat. Karena *database* replikasi hanya mengulangi pernyataan, *database* belum tentu persis di *sync*, dan pengguna yang mahir dapat mengambil keuntungan dari ini.

Diehl (2010) menyatakan pada *Linux Journal* bahwa pada dasarnya *Mysql* menggunakan *model Master-slave* dimana *master* menyimpan *log* dari semua pembaruan *database* yang telah dilakukan. Kemudian, satu atau lebih *slave* terhubung ke *master*, membaca setiap *entri log*, dan melakukan *update* ditunjukkan. *Server master* melacak masalah internal seperti *rotasi log* dan kontrol akses. Setiap *server slave* harus memelihara posisinya saat itu dalam *log* transaksi *server*. Jika transaksi baru terjadi di *server*, mereka mendapatkan *login* pada *server master* dan di *download* oleh setiap *slave*. Setelah transaksi telah dilakukan oleh setiap *slave*, *slave* memperbarui posisi mereka dalam transaksi *server log* dan menunggu untuk transaksi berikutnya. Ini semua dilakukan secara *asynchronous*, yang berarti bahwa *server master* tidak harus menunggu *slave* Ini, juga berarti bahwa jika *slave* tidak dapat terhubung ke *master* untuk periode waktu tertentu, itu hanya bisa "mengejar ketinggalan." *download* semua transaksi tertunda ketika konektivitas terhubung kembali.

Silberschatz et.all (2002) menyatakan bahwa sebuah sistem basis data terdistribusi terdiri dari kumpulan *site-site*, masing-masing *site* ini dapat berpartisipasi dalam pemrosesan transaksi yang mengkases data pada suatu *site* atau beberapa *site*. Beberapa alasan untuk membangun basis data terdistribusi, seperti pemakaian bersama (*share*), kehandalan (*reliability*), ketersediaan (*availability*) dan kecepatan pemrosesan *query*. Keuntungan utama dari basis data terdistribusi adalah kemampuan untuk pemakaian dan pengaksesan data secara bersama dengan cara yang handal dan efisien.

Arsitektur *desentralisasi* merupakan konsep dari pemrosesan data tersebar (atau terdistribusi). Sistem pemrosesan data terdistribusi (atau biasa disebut sebagai komputasi tersebar) sebagai sistem yang terdiri atas sejumlah komputer yang tersebar pada berbagai lokasi yang dihubungkan dengan sarana telekomunikasi dengan masing-masing komputer yang mampu melakukan pemrosesan yang serupa secara mandiri, tetapi bisa saling berinteraksi dalam pertukaran data.

Tiga pendekatan arsitektur alternatif yang dipilih untuk memisahkan fungsionalitas melalui proses yang berkaitan dengan DBMS yang berbeda; Arsitektur DBMS terdistribusi alternatif ini disebut dengan *Client/Server, Collaboration Server dan MiddleWare (Ramakrishnan, 2003).*

Sistem *client-server* mempunyai satu proses klien atau lebih dan satu proses *server* atau lebih, dan proses klien dapat mengirim sebuah *query* pada proses *server* manapun. *Client* bertanggungjawab terhadap proses *user-interface* dan *server* mengatur data dan mengeksekusi transaksi. Jadi, *proses* klien dapat menjalankan komputer personal dan mengirim *query* pada *server* yang berjalan pada kerangka utama.

Arsitektur ini menjadi sangat popular untuk beberapa alasan. Pertama, Arsitektur ini relatif sederhana untuk diimplementasikan berkenaan dengan pemisahan fungsionalitas yang bersih dan servernya disentralisasi. Kedua mesin server yang mahal tidak digunakan berkaitan dengan interaksi pengguna biasa, yang kini dipindahkan ke mesin klien yang murah. Ketiga, pengguna dapat menjalankan *user-interface* grafis yang mereka kenali, daripada *user interface* pada *server* (kemungkinan tidak familiar).

Arsitektur *client-server* tidak mengijinkan *query t*unggal untuk menjangkau banyak *server* karena proses *client* harus mampu memecahkan sebuah *query* ke dalam beberapa *subquery* yang tepat untuk dieksekusi pada tempat yang berbeda dan kemudian membagi jawaban ke *subquery.* Oleh karena itu, Proses *client* cukup komplek dan kemampuannya akan mulai overlap dengan server; sehingga perbedaan antara *client* dan *server* menjadi sulit. Untuk me-

ngurangi perbedaan digunakan alternatif arsitektur *client-server* yaitu sistem *Collaboration Server*. Pada sistem ini terdapat sekumpulan *server database*, masing-masing mampu menjalankan transaksi melalui data lokal, yang secara bekerjasama mengeksekusi transaksi yang menjangkau banyak *server*.

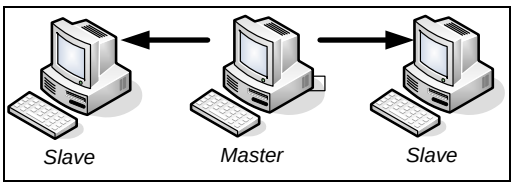
Saat *server* menerima *query* yang membutuhkan akses ke data pada *server* lain, server menghasilkan *sub-query* yang tepat untuk dieksekusi *server* lain dan menempatkan hasilnya bersama-sama untuk menggabungkan jawaban menjadi *query* asal *(Ramakrishnan, 2003).*

Arsitektur *middleware* didesain untuk memungkinkan sebuah *query* tunggal menjangkau banyak *server,* tanpa membutuhkan semua *server database* untuk mampu mengatur strategi eksekusi *multi-situs*. Arsitektur ini menjadi sangat menarik saat berusaha untuk mengintegrasikan beberapa sistem warisan, yang kemampuan dasarnya tidak dapat diperluas.

Gagasannya hanya membutuhkan satu *server database* yang mampu mengatur *query* dan transaksi yang menjangkau banyak *server; server* yang tersisa hanya perlu mengendalikan *query* dan transaksi lokal. *Server* khusus ini sebagai lapisan perangkat lunak yang mengkoordinasikan eksekusi *query* dan transaksi melalui satu *server database* yang *independent* atau lebih; perangkat lunak tersebut sering disebut *Middleware*. Lapisan *middleware* mampu mengeksekusi *join* dan operasi relasional lain pada data yang diperoleh dari *server* lain, secara khusus tidak sendirinya menggunakan data *(Ramakrishnan, 2003).*

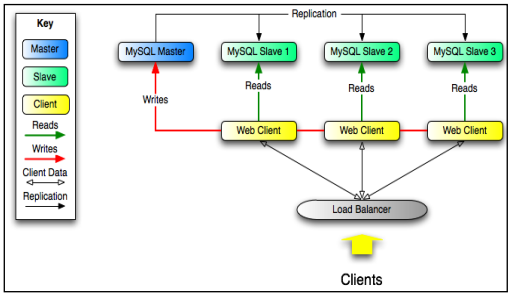
(Silberschatsz A., 2002), menyatakan bahwa jika relasi r direplikasi, maka sebuah salinan dari relasi disimpan pada dua atau lebih lokasi. Dala kasus yang ekstrim, kita bisa juga memiliki replikasi penuh (*Full Replication*), dimana sebuah salinan dari relasi r dapat disimpan pada setiap lokasi. Dalam sistem replikasi terdapat sejumlah kelebihan dan kekurangan pada antara lain: *Availibility*, jika salah satu dari lokasi yang berisi relasi r mengalami kegagalan, maka relasi r dapat ditemukan pada lokasi lain, sehingga sistem dapat melanjutkan untuk memproses *query* yang berisi r walaupun terjadi kegagalan sebelumnya.

Replikasi memungkinkan membangun data dari satu *server* basis data *mysql* (disebut *master*) direplikasikan ke satu atau lebih server basis data *mysql* (disebut *slaves*). Replikasi *mysql* ini adalah *asynchronous*, yaitu *slave* tidak harus terhubung secara permanen untuk menerima *update* dari *master*. Replikasi pada umumnya digunakan atas sebuah *master* dan sebuah *slave* seperti terlihat pada Gambar 1.



Gambar 1. *Master Multi Slave*

(Pachev, 2008) menjelaskan tentang tujuan digunakannya replikasi pada *MySQL* meliputi: *Scale-out solutions,* pada Gambar 2 membuat banyak *slave* untuk meningkatkan *performa*. Dalam hal ini semua transaksi *write* dan *update* harus dilakukan pada master.



Gambar 2 Replika untuk meningkatkan performa selama *scale-out* (Pachev, 2008)

Penelitian ini bertujuan untuk mengkaji secara lebih rinci, hal-hal yang akan dilakukan dan menjadi tujuan dalam penelitian ini adalah mencari algoritma-algoritma untuk merancang arsitektur database terdistribusi dengan metode *replikasi* yang akan diimplementasikan pada *database* akademik. Sehingga akan didapatkan sebuah arsitektur *database* terdistribusi yang bisa mengatasi ketersediaan data pada sistem informasi aka-

demik. Dengan database terdistribusi tersebut maka sebuah aplikasi bisa memberikan informasi kepada *end user*-nya dengan mengambil dari sumber data yang terdekat dengan *network* dari *end user* tersebut melakukan transaksi.

METODE

Pada penelitian yang dilakukan menggunakan bahan berupa perangkat keras dan perangkat lunak. Perangkat Keras meliputi: perangkat *networking (Infrastruktur Jaringan intranet* dan atau *internet), Server (PC server untuk Web dan Database Server)* dan *Client (PC terminal* untuk pengujian *proses. Perangkat Lunak / Software* terdiri dari *Ubuntu Server, Windows, DBMS MySQL, Apache Web Server dan Client.*

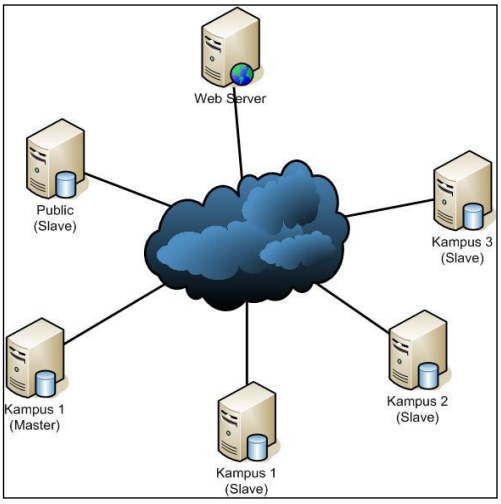
Proses pada penelitian ini meliputi langkah-langkah *instalasi networking system* baik *hardware* maupun *software* serta aplikasi-aplikasi pendukung, *instalasi master database, instalasi slave database* dan instalasi aplikasi *web base*.

Langkah pemasangan *database* akademik pada *master, slave, pembuatan user replikasi pada master database*, proses *sinkronisasi database* dari *master* ke *slave*, serta pengujian menggunakan aplikasi dari beberapa kelas *network*.

PEMBAHASAN

Prototipe perancangan dan instalasi networking terlihat pada Gambar 3 menjelaskan perancangan networking dari permasalahan, dengan mengacu Gambar 2 tentang *Replika* untuk meningkatkan *performa selama scale-out* maka terbentuklah rancangan ini.

Fungsi dari masing-masing node pada Gambar 3 dapat dijelaskan sebagai berikut: Web Server, sebagai tempat untuk aplikasi berbasis web yang didukung oleh Apache dan diletakkan pada sebuah domain di area public (Internet), dengan cara ini semua *stake-holder* akan berinteraksi dengan sistem database. Kampus 1 (*Master)*, sebuah database server yang diletakkan di area Intranet, mesin ini digunakan sebagai *MySQL master* dari replikasi, semua transaksi dipusatkan di mesin ini. Kampus 1 *(Slave*), sebuah database server yang diletakkan di area intranet, mesin ini sebagai slave yang akan melayani semua *request* yang terletak di area Kampus 1. Kampus 2 (*Slave*), sebuah database server yang diletakkan di area intranet, mesin ini sebagai slave yang akan melayani semua *request* yang terletak di area Kampus 2. Kampus 3 (*Slave)*, sebuah database server yang diletakkan di area intranet, mesin ini sebagai slave yang akan melayani semua *request* yang terletak di area Kampus 3. Public (*Slave*), sebuah database server yang diletakkan di area public, mesin ini sebagai slave yang akan melayani semua *request* yang datang dari *public/internet*.

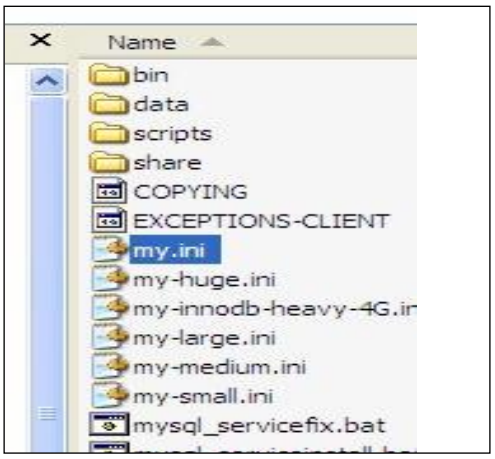


Gambar 3. Instalasi Networking

Proses konfigurasi *master*, file konfigurasi pada file *my.ini* yang terletak di folder MySQL seperti pada Gambar 4. *Mapping network system* yang digunakan dalam penelitian ini dapat ditunjukkan pada Tabel 1.

Tabel 1. Peta IP Address Network

Lokasi	IP-Server	IP-Range
Kampus 1 (Master)	15.74.0.1	15.74.0.1 – 15.74.0.255
Kampus 1 (Slave)	15.74.1.1	15.74.1.1 – 15.74.1.255
Kampus 2 (Slave)	15.74.2.1	15.74.1.1 – 15.74.1.255
Kampus 3 (Slave)	15.74.3.1	15.74.3.1 – 15.74.3.255
Public (Slave)	202.91.10 .214	202.91.10.209 – 202.91.10.222



Gambar 4 Lokasi my.ini

File my.ini sebagai konfigurasi dari database MySQL perlu dilakukan penambahan *script* untuk mendefinisikan sebagai *master* dan atau *slave* dengan script konfigurasi sebagai berikut:

Master:
[mysqld]
#sebagai server
log-bin=istaakdk-log
server-id=1574
binlog-do-db=akademik

Sedangkan untuk melihat file log digunakan *show binlog events*

```
mysql> show binlog events\G
*************************** 1. row
***************************
   Log_name: istaakd-log.000001
        Pos: 4
 Event_type: Format_desc
  Server_id: 1574
End_log_pos: 98
       Info: Server ver: 5.0.24a-
community-nt-log, Binlog ver: 4
1 row in set (0.00 sec)
```

Dari tampilan diatas terlihat bahwa *master* berjalan dan mencatat kegiatan pada file log *istaakd-log.000001* mulai posisi 98, sedangkan pada *events* terlihat bahwa ada satu kejadian (*1. row*) pada log *istaakd-log.000001* posisi awal 4 pada *server-id* 1574 dan posisi akhir log adalah 98.

Agar *master* ini bisa di koneksikan ke *slave*, maka pada Site *master* harus dibuatkan user dengan hak *Replication Slave* sebagai berikut:

```
Mysql>CREATE USER 'ista'@ '%'
IDENTIFIED BY '****';
mysql>GRANT REPLICATION SLAVE ON *.*
TO 'ista'@ '%' IDENTIFIED BY '****';
```

Pada knfigurasi ini dilakukan di semua slave (Kampus 1, 2 dan 3 serta Public), Mesin ini akan bertindak sebagai *slave* dan hanya akan melakukan pembacaan log terhadap basisdata *akademik,* sehingga semua kejadian pada *master* terhadap basisdata *akademik* tersebut akan dikirimkan ke *slave*.

File my.ini sebagai konfigurasi dari database MySQL perlu dilakukan penambahan *script* untuk mendefinisikan sebagai *slave* dengan konfigurasi sebagai berikut :

```
[mysqld]
server-id= 15741
#sebagai slave
master-host=15.74.0.1
master-user=ista
master-password=ista
#slave hanya mengakses dB pileg
dan tabel-tabel tertentu
replicate-do-db=akademik
```

Dengan penjelasan sebagai berikut :

server-id=15741, digunakan untuk mengidentifikasi server MySQL, dalam hal ini masing-masing *slave* memiliki id yang berbeda.

master-host=15.74.0.1, digunakan untuk menunjukkan bahwa *master* basisdata yang ditunjuk adalah nomor *ip address* 15.74.0.1.

master-user=ista, digunakan untuk mendefinisikan user MySQL *master* yang diberi hak sebagai *replication slave*.

master-password=ista, digunakan untuk mendefinisikan password dari user.

replicate-do-db=akademik,digunakan untuk mendefinisikan bahwa database yang direplikasi adalah *akademik*.

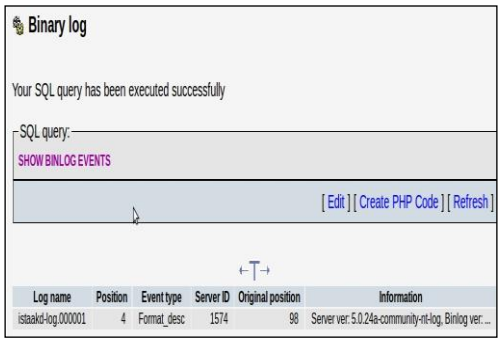
Setelah konfigurasi *master* dan *slave* tersebut di terapkan, maka kedua sisi database server sudah akan bertindak sebagai *master* dan *slave.*

Konfigurasi pada aplikasi web lebih difokuskan pada penanganan saat membuka database, yaitu dengan mendeteksi dari manakah client itu mengakses.. Script ini digunakan untuk mengatur semua hal yang terkait untuk menggunakan basisdata. Script memperlihatkan penanganan tersebut yaitu:

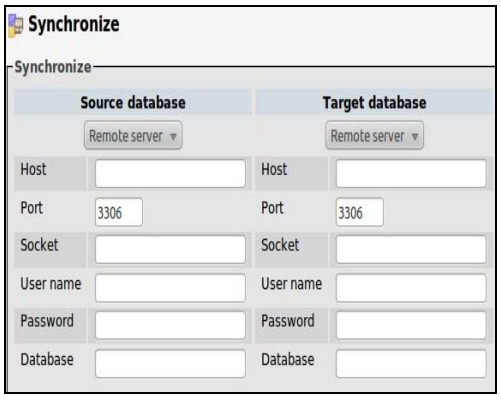
```
<?php
$asal=$_SERVER['REMOTE_ADDR'];
$client=substr($client, 0, 7);
```

```
If($client=='15.74.0.') {
       $dbs='15.74.0.1';
} ElseIf($client=='15.74.1.') {
       $dbs='15.74.1.1';
} ElseIf($client=='15.74.2.') {
       $dbs='15.74.2.1';
} ElseIf($client=='15.74.3.') {
       $dbs='15.74.3.1';
} Else { $dbs='202.91.10.214'; }
$con=mysql_connect($dbs,'ista','15
74');
if(!$con) { die("Mysql belum
diaktifkan, silahkan aktifkan
dulu"); }
if(!mysql_select_db('akademik',$co
n)) die("database salah ");
```

Proses Sinkronisasi**,** pada tahap ini memantau transaksi antara *master* dan *slave*, *Binary Log Master* pada Gambar 5, untuk melihat *status master*.

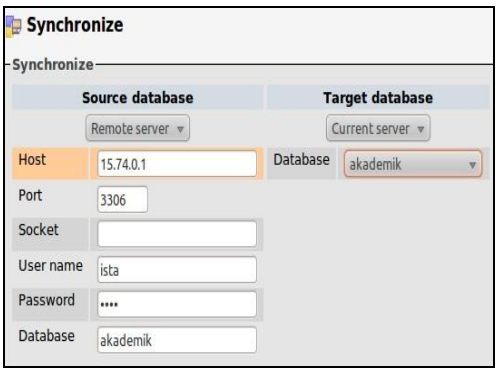


Gambar 5. Status *Master*



Gambar 6. Dialog Sinkronisasi

Pada Gambar 6 memperlihatkan dialog untuk melakukan proses *sinkronisasi*, *source database* adalah sumber data dalam hal ini adalah master dan *target database* adalah tujuan *slave*. Proses sinkronisasi dengan memasukkan data master ke form *source database*.



Gambar 7. Proses sinkronisasi

Dan Slave ke *Target Database* seperti terlihat pada Gambar 7.Setelah terhubung deng- an master akan didapat informasi yang di tampilkan pada gambar 8. Legend S menginformasikan jika ada perbedaan atau ada struktur data yang baru atau berubah, sedangkan legend D menunjuk-kan jika ada perubahan pada rekord di tabel tersebut.

Tabel 2. Peta Transaksi Sistem Akademik

Nama Kegiatan	Modul Terkait	Waktu
Pendaftaran Mahasiswa	Data Induk Calon Mahasiswa	Tiap periode penerimaan mahasiswa
Her-Registrasi Mahasiswa	Data Induk Mahasiswa	Tiap periode Her-Registrasi
Pengisian KRS	Data Matakuliah, Dosen, Kelas dan KRS	Tiap semester
Pengisian Nilai	Data KRS/KHS dan Transkrip	Tiap Semester 2 kali (UTS dan UAS)
Yudisium	Data Transkrip	Tiap Periode Yudisium
Alumni	Data Lulusan dan Alumni	Tiap periode Yudisium

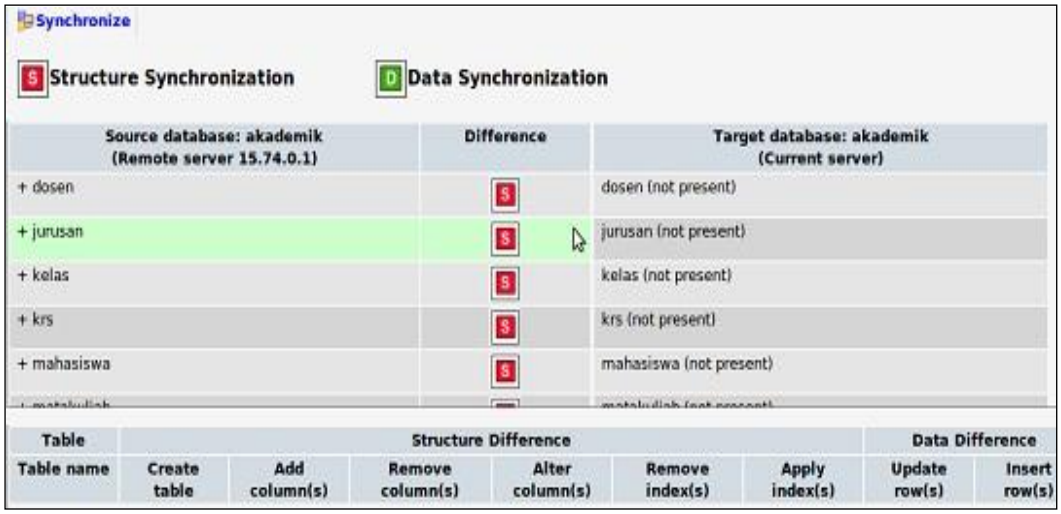
Setelah memilih tabel yang akan di sinkronisasikan maka akan terlihat pada gambar 9, kejadian apa yang terjadi pada object tersebut. Pada gambar 10 memperlihatkan proses sinkronisasi telah dilakukan, dan diperlihatkan *query* yang

terjadi. Pada Gambar 11 memperlihatkan perubahan data pada tabel yang dilakukan sinkronisasi yaitu melakukan transaksi Insert atau Update pada tabel tersebut. Setelah proses ini selesai dilakukan, maka database slave telah terupdate dengan data terkini. Dan bisa dipertanggungjawabkan sebagai sumber data bagi *stake holder* yang mengakses Sistem Informasi Akademik.

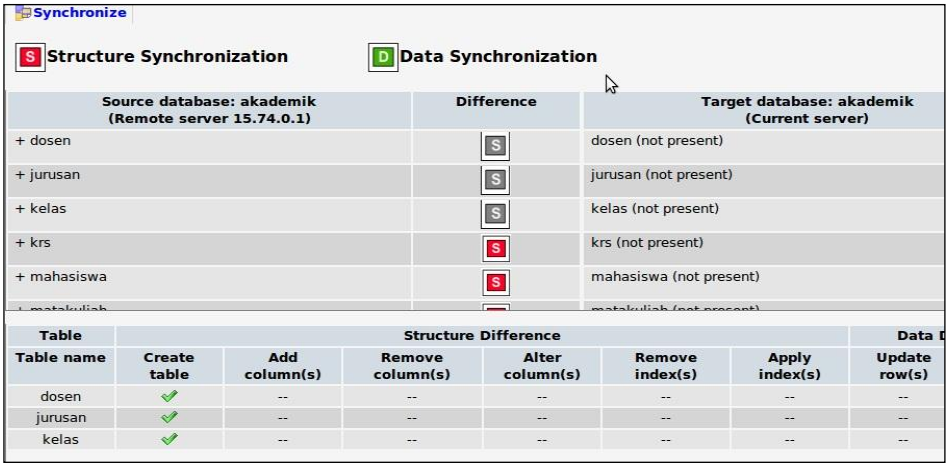
Dalam penelitian ini database Slave merupakan data mirror yang tidak pernah dilakukan transaksi, karena sifatnya mirror maka jika terjadi suatu kerusakan baik disengaja maupun tidak, maka kita bisa melakukan perbaikan dengan cara mengambil lagi dari master untuk di sinkronisasikan ke slave tersebut.

Proses sinkronisasi data bisa dilakukan secara otomatis dengan pola waktu tertentu, ataupun dengan cara terjadwal. Dalam penelitian ini, karena proses perubahan data yang terjadi pada sistem informasi akademik tidak begitu padat yaitu hanya terjadi pada waktu-waktu tertentu, maka untuk efisiensi proses sinkronisasi dilakukan menurut *schedule* dengan mengikuti pola seperti pada tabel 2.

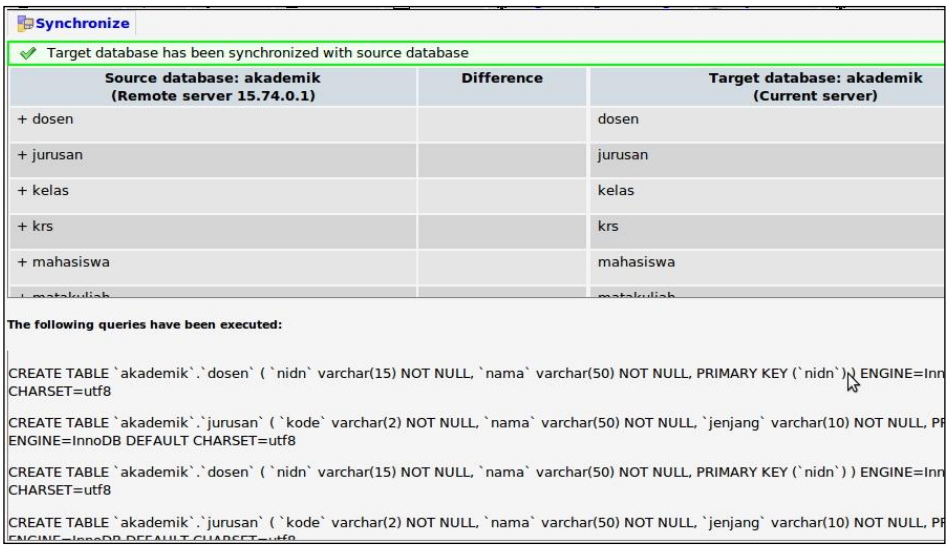
Dari peta pada tabel 2 terlihat bahwa pada sistem informasi akademik proses sinkronisasi bisa dilakukan mengikuti pola kejadian tersebut. Misalkan Pengisian Nilai, dengan data yang terkait adalah KRS/KHS dan Transkrip, sinkronisasi cukup dilakukan tiap Semester 2 kali, yaitu pada periode UTS dan periode UAS.



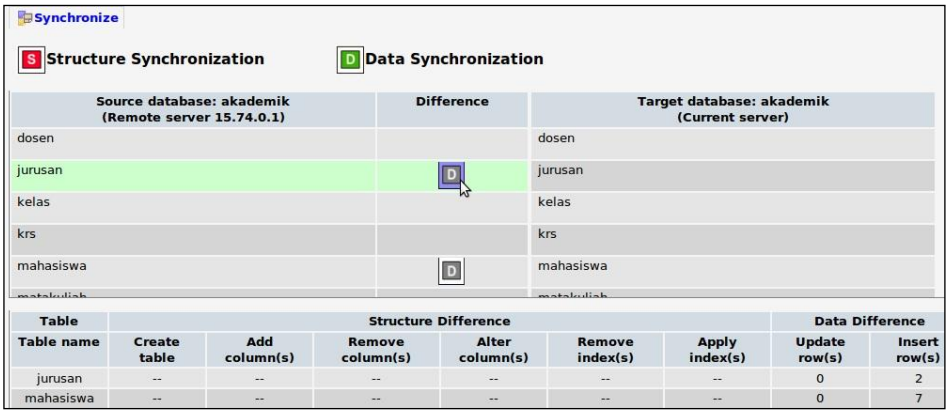
Gambar 8. Perubahan Data



Gambar 9. Struktur data yang dipilih



Gambar 10. Data terpilih di proses



Gambar 11. Perubahan isi tabel

KESIMPULAN

Dari pembahasan dan peng-ujian yang telah dilakukan maka dapat disimpulkan yaitu metode yang di-bangun ini cocok untuk sistem yang mengacu kepada proses transaksi terpusat, yaitu hanya di site *master* saja terjadinya transaksi, sedangkan site *slave* hanya melayani untuk proses baca atau *view* data sehingga tidak akan terjadi konfik data. Dengan metode ini, maka *slave* bisa disebarkan ke tempat-tempat atau *node-node* strategis di *area public* atau bahkan di *intranet* yang terpisah, proses *sinkronisasi* dilakukan secara *temporal.* Sehingga walaupun koneksi ke *master* terputus sistem informasi yang mengacu ke *slave* akan tetap bisa bekerja secara sempurna.

Slave pada penelitian ini ber-fungsi sebagai hanya sebagai mirror, maka jika terjadi kerusakan pada slave tidak akan mengganggu sistem secara keseluruhan, dan pengembalian data cukup mudah dilakukan dengan sin-kronisasi ulang. Keamanan data *master* akan terjaga, karena *master* di tem-patkan pada *intranet*, yang hanya dikoneksikan ke area *public* untuk periode tertentu saja.

DAFTAR PUSTAKA

Diehl, M. (2010, May 25). *Database Replication with Mysql.* Dipetik Maret 8, 2012, dari Linux Journal: http://www.linuxjournal.com/content/database-replication-mysql

Gilfillan, I. (2004, Mai 18). *Database Replication in MySQL.* Dipetik

Maret 8, 2012, dari Database Journal: http://www.databasejournal.com/features/mysql/article.php/3355201/Database-Replication-in-MySQL.htm

Pachev, S. (2008, 10 28). *MySQL 5.0 Manual Reference*. Dipetik 10 28, 2009, dari MySQL Manual: http://www.mysql.com/doc/

Ramakrishnan, R. G. (2003). Database Management System, Third Edition. The McGraw-Hill Companies, Inc.

Silberschatsz A., K. H. (2002). Database System Consepts, 3nd ed. Singapore: McGrow-Hill.

Triyono, J. (2010). *Implementasi Sistem Database Terdistribusi Dengan Metode Partial Replica.* Yogyakarta: FMIPA UGM.