

Memfaatkan Replikasi MySQL 5.0

31 Mei 2007

Daftar Isi

1	Pengantar	2
1.1	Tujuan	2
1.2	Latar belakang	3
2	Setup Replikasi Master Slave	3
2.1	Setup server Master	3
2.1.1	Instal mysql server di mesin master	3
2.1.2	File konfigurasi	4
2.1.3	Menjalankan server master	5
2.1.4	Menset hak akses untuk replikasi	6
2.1.5	Membuat arsip base backup	6
2.2	Setup server Slave	7
2.2.1	Instal mysql server di mesin slave	7
2.2.2	Setup datadir dari arsip base backup.	7
2.2.3	File konfigurasi	8
2.2.4	Menjalankan server slave	8
2.2.5	Setup parameter replikasi	9
2.2.6	Menjalankan dan memeriksa thread slave	9
2.3	Menguji replikasi dan troubleshooting	9
2.4	Mengoptimalkan replikasi	9
2.4.1	Mereplikasi database tertentu	9
2.4.2	Menyimpan binary log di direktori tertentu	10
2.4.3	Menyediakan disk space yang cukup besar untuk file temporary di slave	10
2.5	Catatan bagi database administrator	11
2.5.1	Memeriksa status master dan slave	11
2.5.2	Ketika mesin server slave restart	11
2.5.3	Memperhatikan disk space untuk binary log	11
2.5.4	Berhati-hati dalam menshutdown server slave	11

1	PENGANTAR	2
3	Ketika Master crash	11
3.1	Mempromosikan slave menjadi master	11
3.2	Membuat slave baru	12
3.3	Alternatif lain: mebuat master baru	12
4	Membuat base backup secara periodik	12
4.1	Mengapa perlu backup lagi?	12
4.1.1	Sebagai backup sekunder.	13
4.1.2	Diperlukan untuk memulihkan kesalahan eksekusi update SQL di master secara cepat	13
4.2	Apa saja dan bagaimana	13
4.3	Script untuk otomatisasi	13
5	Memulihkan dari kesalahan eksekusi update SQL di master	15
5.1	Mengunci master dari user	15
5.2	Mencari posisi binlog terakhir	16
5.3	Menggunakan base backup terakhir	16
5.4	Menjalankan thread slave sampai posisi tertentu	16
5.5	Mengganti data master	17
5.6	Menjalankan thread slave seperti biasa	17
6	Keterbatasan-keterbatasan Replikasi	17
7	Penutup	19

1 Pengantar

1.1 Tujuan

Setelah membaca tulisan ini, pembaca diharapkan mengetahui apa yang perlu dipersiapkan agar dapat memanfaatkan replikasi secara efektif, sehingga pada akhirnya ia dapat :

1. Mensetup replikasi master-slave dengan MySQL 5
2. Melakukan recovery jika master crash atau jika terjadi kesalahan eksekusi perintah di master
3. Mengetahui potensi-potensi dari replikasi sekaligus menghindari kondisi-kondisi yang dapat menyebabkan masalah ketika menggunakan replikasi akibat keterbatasan-keterbatasan dalam sistem replikasi tersebut.

1.2 Latar belakang

Aset paling berharga dan paling panjang umur pakainya dari sebuah sistem informasi adalah data. Sayangnya, upaya menjaga integritas data dan availability data tidaklah mudah, terlebih jika volumenya semakin besar. Replikasi adalah salah satu diantara sedikit cara untuk mencapai integritas dan availability data.

Replikasi memungkinkan tersedianya database lain yang keadaannya mendekati sama dengan database tempat pertamakali terjadinya update data. Database lain itu disebut slave dan database tempat pertamakali terjadinya update disebut master.

Hal ini memberikan keuntungan-keuntungan sebagai berikut:

1. Tersedia backup yang setiap saat hampir sama dengan database asli.
2. Database lain tersebut dapat diakses oleh aplikasi secara terjaga integritasnya, selama aplikasi hanya melakukan update ke database master.
3. Kondisi (2) memungkinkan pendistribusian beban akses read-only ke beberapa database slave. Sehingga dapat meningkatkan kinerja sistem aplikasi secara keseluruhan.

Dokumen ini membahas replikasi dengan single master dan single slave.

2 Setup Replikasi Master Slave

2.1 Setup server Master

Bagian ini membahas cara instalasi mysql 5, dan mensetupnya sebagai server master.

2.1.1 Instal mysql server di mesin master

- Mesin master adalah linux server
- IP address 192.168.23.9
- Direktori instalasi /usr/local/mysql5
- Direktori data di /data/mysqldev

Berikut ini adalah cara instalasinya:

1. Download mysql5 dari www.mysql.com, misalkan di dir /home/yan
2. Uraikan file distribusi .tar.gz di /usr/local sebagai berikut:
% cd /usr/local
% tar zxvf /home/yan/mysql-standard-5.0.15-linux-i686-glibc23.tar.gz

3. Buat simbolik link:


```
% cd /usr/local
% ln -s mysql-standard-5.0.15-linux-i686-glibc23 mysql5
```
4. Buat user mysql:


```
% groupadd mysql
% useradd mysql -g mysql
```
5. Setup PATH, agar semua user dapat mengakses program-program mysql, edit `/etc/profile`, tambahkan :


```
export PATH=/usr/local/mysql5/bin:$PATH
```
6. Inisialisasi direktori data (`datadir`) , di `/data/mysqldev`:


```
% cd /usr/local/mysql5
% ./scripts/mysql_install_db --datadir=/data/mysqldev
```
7. Selesai

2.1.2 File konfigurasi

File konfigurasi disimpan sebagai `/etc/my5.cnf`, berikut ini parameter konfigurasi umum (bukan spesifik master):

1. `basedir=/usr/local/mysql5`
2. `datadir=/data/mysqldev`
3. `socket=/var/lib/mysql/mysql5.sock`
4. `port=3333`
5. `log-error=/var/log/mysqld/mysql5.log`

Selanjutnya, parameter berikut ini diperlukan untuk server bertindak sebagai master:

1. `log-bin=mysql-binlog`
Digunakan untuk mengaktifkan binary log
2. `server-id=1`
3. `sync-binlog=1`
synchronize setiap database update dengan binary log sesering mungkin.

Berikut ini file konfigurasi selengkapnya:

```
[mysqld]
basedir=/usr/local/mysql5
datadir=/mnt/data/mydata
socket=/var/lib/mysql/mysql.sock
slave-load-tmpdir=/tmp/
tmpdir=/tmp/

#_setup_master
log-bin=mysql-binlog
server-id=1
sync-binlog=1

[mysql.server]
user=mysql

[mysqld_safe]
log-error=/var/log/mysqld/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
ledir=/usr/local/mysql5/bin
```

2.1.3 Menjalankan server master

Selanjutnya anda perlu menjalankan server dan memeriksanya:

1. Agar server beroperasi di background, jalankan :

```
% mysqld_safe --defaults-file=/etc/my5.cnf &
```
2. Login ke mysql sebagai root di mysql (bukan root di system):

```
% mysql -u root -S /var/lib/mysql/mysql5.sock
```
3. Periksa apakah server sudah berjalan sebagai master, jalankan:

```
mysql>SHOW MASTER STATUS
```

Seharusnya muncul tampilan seperti ini:

```
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-binlog.000003 |      98 |              |                  |
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

2.1.4 Menset hak akses untuk replikasi

Berikut ini akan didefinisikan setup agar user 'replikator' dapat mengakses master untuk melakukan replikasi. Dalam hal ini slave akan dioperasikan dari mesin dengan IP address 192.168.23.10.

Jalankan perintah berikut:

1. Login ke mysql sebagai mysql root
2. Jalankan perintah berikut di prompt mysql


```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replikator'@'192.168.23.10'
IDENTIFIED BY 'b1smill4h';
```

2.1.5 Membuat arsip base backup

Base backup atau backup dasar adalah seluruh isi file system dari data yang akan dibackup, file system bisa mencakup seluruh direktori data (datadir) atau bisa juga hanya satu atau lebih direktori database tertentu, yang ada dibawah datadir.

Anda dapat membuat base backup tanpa harus menshutdown server master. Secara teori, base backup dapat dibuat sesering yang diinginkan, tetapi pada kenyataannya seperti akan kita lihat nanti, frekuensi base backup akan dibatasi oleh waktu akses read-only yang bisa ditolerir oleh aplikasi yang menggunakan database master tersebut.

Target replikasi pada setup ini adalah seluruh database di master, jadi perlu dibuat base backup dari isi direktori data (datadir) di /data/mysqldev. Berikut ini caranya:

1. Pastikan database dalam keadaan read-only. Login sebagai mysql root lalu jalankan:


```
mysql> FLUSH TABLES WITH READ LOCK
```

 Perhatikan bahwa setelah anda menjalankan perintah ini, pengguna tidak dapat melakukan insert, update dan delete ke database.
2. Lihat posisi binary log, catat parameter File dan Position:


```
mysql> SHOW MASTER STATUS
```

 Misalkan tampil sebagai berikut:

```
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-binlog.000003 |      98 |              |                   |
```

```
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

- Gunakan tar untuk membuat file arsip base backup, nama file arsip sebaiknya menggunakan parameter File dan Position, supaya *self-documented*, misal backup_mysql-binlog.000003_98.tgz. Jika datadir adalah /data/mysqldev, maka jalankan:

```
% cd /data
% tar zcvf /home/yan/backup_mysql-binlog.000003_98.tgz mysqldev
```

- Setelah file arsip selesai dibuat, buka kunci database dari akses read-only:

```
mysql>UNLOCK TABLES
```

2.2 Setup server Slave

2.2.1 Instal mysql server di mesin slave

- Mesin master adalah linux server
- IP address 192.168.23.10
- Direktori instalasi /usr/local/mysql5
- Direktori data di /data/slave

Tahap instalasi hampir sama dengan instalasi master dibagian sebelumnya, perbedaannya adalah langkah ke 6 (enam), yaitu inisialisasi datadir tidak diperlukan karena kita akan menggunakan base backup sebagai datadir.

2.2.2 Setup datadir dari arsip base backup.

Alih-alih melakukan inisialisasi dengan script, datadir dibuat dengan mende-kompresi file base backup dan mengubah nama direktori, jalankan:

- Copy file arsip base backup ke slave, dapat menggunakan scp atau ftp.
- Dekompres file arsip
- Rename direktori, langkah ini sebenarnya tidak mutlak diperlukan, kami hanya ingin nama direktori yang memberikan petunjuk bahwa yang tersimpan adalah data dari slave dalam proses replikasi.
- Ubah kepemilikan datadir untuk user mysql, tentunya user mysql ini sudah harus ada di server slave.

2.2.3 File konfigurasi

File konfigurasi disimpan sebagai `/etc/slave_my.cnf`, berikut ini parameter konfigurasi umum (bukan spesifik slave):

1. `basedir=/usr/local/mysql5`
2. `datadir=/data/slave`
3. `socket=/tmp/slavemysql.sock`
4. `port=4444`
5. `log-error=/var/log/mysqld/mysql5.log`

Selanjutnya, parameter berikut ini disarankan untuk server bertindak sebagai slave:

1. `skip-slave-start`
mencegah slave start otomatis jika mesin server restart.
2. `read-only`
mencegah slave diupdate dari luar proses replikasi, karena dapat menyebabkan ketidaksinkronan database.
3. `log-warnings`
mencatat warning dalam proses replikasi
4. `server-id=2`
identitas server harus berbeda dengan master

2.2.4 Menjalankan server slave

Selanjutnya anda perlu menjalankan server dan memeriksanya:

1. Agar server beroperasi di background, jalankan :
`% mysqld_safe -defaults-file=/etc/slave_my.cnf &`
2. Login ke mysql sebagai root di mysql (bukan root di system):
`% mysql -u root -S /tmp/slavemysql.sock`
3. Periksa apakah server sudah berjalan sebagai master, jalankan:
`mysql>SHOW SLAVE STATUS`

Pada kolom `Slave_IO_State` anda akan mendapati nilai kosong, karena thread slave belum berjalan.

2.2.5 Setup parameter replikasi

Slave perlu diberi informasi tentang alamat (ip address), file binary log, posisi file master dan informasi otentikasi untuk mengakses master, caranya:

1. Login ke server slave sebagai mysql root.
2. Jalankan perintah :

```
mysql>CHANGE MASTER TO MASTER_HOST='192.168.23.9', MASTER_PORT=3333,  
MASTER_USER='replikator',MASTER_PASSWORD='b1smil4h', MASTER_LOG_FILE='mysql-  
binlog.000003', MASTER_LOG_POS=98;
```

Parameter replikasi akan tersimpan persistent di server, sampai anda mengubahnya lagi dengan perintah yang sama.

2.2.6 Menjalankan dan memeriksa thread slave

Proses replikasi di slave baru akan berjalan jika ada perintah START SLAVE :

1. Login ke server slave sebagai mysql root.
2. Jalankan perintah START SLAVE
3. Periksa status slave

Pada kolom Slave_IO_State anda akan mendapati teks 'Waiting for master to send event', artinya thread server sudah aktif dan menunggu update event dari master.

2.3 Menguji replikasi dan troubleshooting

1. Menguji replikasi
2. Troubleshooting

2.4 Mengoptimalkan replikasi

Hal-hal yang dapat dilakukan untuk supaya replikasi dapat berjalan lebih efisien dan andal.

2.4.1 Mereplikasi database tertentu

Adakalanya anda tidak ingin mereplikasi semua database, tapi hanya database tertentu saja. Misalkan anda ingin mereplikasi database garuda dan gsmdev7 saja, tambahkan option berikut di konfigurasi master:

```
binlog-do-db=garuda
```

```
binlog-do-db=mysql
```

Tapi perhatikan bahwa jika anda tidak mereplikasi database mysql, ada kemungkinan anda menghadapi masalah ketika aplikasi menggunakan data dari slave (setelah melakukan restore), jika sebelumnya terjadi perubahan grant di master.

2.4.2 Menyimpan binary log di direktori tertentu

Binary log akan terus bertambah, maka sebaiknya binary log ada di partisi yang berbeda dengan datadir.

Gunakan parameter log-bin berikut ini di konfigurasi master, untuk memindahkan lokasi binary log ke partisi /blog:

```
log-bin=/blog/mysql-binlog
```

Perhatikan: jika anda memindahkan binary log ke lokasi baru, anda tidak cukup hanya mengedit konfigurasi master, lalu merestart database, cara paling aman adalah:

1. Stop master
2. Edit konfigurasi
3. Copy (bukan pindahkan) binary log lama ke lokasi baru, termasuk binlog .index
4. Startup master
5. Buat base backup terkini.
6. Setup ulang slave dari base backup yang paling baru.
7. Start up slave
8. Selesai

2.4.3 Menyediakan disk space yang cukup besar untuk file temporary di slave

Jika slave mereplikasi perintah LOAD DATA INFILE, dan file yang di load di master berukuran besar, maka slave membutuhkan ruang cukup besar untuk menyimpan file ini, untuk sementara, sebelum statemen ini selesai dieksekusi.

Parameter yang perlu diset adalah slave-load-tmpdir=/partisi/yang/cukup/besar.

2.5 Catatan bagi database administrator

2.5.1 Memeriksa status master dan slave

Administrator perlu memeriksa status master dan slave secara teratur, karena tidak ada jaminan kedua server akan berjalan tanpa masalah sepanjang masa. Jika ada statement dari master yang tidak dapat dieksekusi karena suatu sebab (misal: tidak ada temporary table, grant permission tidak memadai)

Perhatikan bahwa replikasi pada MySQL 5.0 memiliki keterbatasan-keterbatasan yang mungkin saja dilampaui oleh aplikasi anda. Baca manual mysql Bab 6 bagian 6.7 “Replication Features and Known Problems”. Pada bagian akhir tulisan ini kami merangkum beberapa keterbatasan yang kami anggap relevan.

Sebagian dari keterbatasan ini merupakan bawaan dari sifat replikasi yang mereplikasikan statement bukan data.

2.5.2 Ketika mesin server slave restart

Jika mesin slave reboot atau restart, administrator harus menjalankan thread slave secara manual, sehingga dapat memastikan apakah thread slave sukses dijalankan.

2.5.3 Memperhatikan disk space untuk binary log

Semakin tinggi laju transaksi database, ukuran binary log bertambah dengan cepat, sangat disarankan binary log yang sudah terkirim ke slave dipindahkan ke media backup offline, misal cdrom atau tape backup. Sehingga selalu tersedia ruang yang cukup untuk binary log yang baru.

2.5.4 Berhati-hati dalam menshutdown server slave

Pada saat menshutdown server slave, pastikan bahwa thread eksekusi query slave sudah mengeksekusi semua relay log. Jika anda terpaksa menshutdown ketika ada relay log yang belum dieksekusi, usahakan jangan sampai ada tabel temporer yang terbuka, untuk penjelasan lebih detil, lihat bagian 6 poin 5.

3 Ketika Master crash

Ketika server master crash, dan tidak dapat segera dipulihkan, inilah saatnya memanfaatkan server slave untuk menggantikannya.

3.1 Mempromosikan slave menjadi master

Berikut ini prosedur untuk mempromosikan slave menjadi master:

1. Shutdown server slave, jalankan:
`%mysqladmin -u root shutdown`
2. Copy file konfigurasi slave `/etc/slave.cnf` menjadi `/etc/master.cnf`,
3. Edit file `/etc/master.cnf`, hapus semua parameter yang berhubungan dengan setting slave, yaitu: `read-only,skip-slave-start`. Set `server-id=1`
4. Jalankan master baru :
5. Ubah konfigurasi aplikasi, (`config.php`), arahkan ke master baru ini
6. Selesai

3.2 Membuat slave baru

Buat slave baru, agar master baru dapat direplikasi. Prosedurnya seperti yang dijelaskan dibagian sebelumnya.

3.3 Alternatif lain: mebuat master baru

Alternatif ini lebih disukai jika yang crash hanya instalasi database master saja dan tidak dapat segera dipulihkan, sedangkan sistem operasi masih berjalan baik. Prosedurnya:

1. Stop thread slave
2. Backup datadir slave
3. Copy arsip datadir slave ke master
4. Gunakan arsip tersebut sebagai datadir master
5. Setup konfigurasi master baru
6. Jalankan master
7. Jalankan thread slave
8. selesai

4 Membuat base backup secara periodik

4.1 Mengapa perlu backup lagi?

Replikasi dapat digunakan sebagai backup, walau demikian ada sedikitnya 2 alasan perlunya melakukan base backup secara periodik:

4.1.1 Sebagai backup sekunder.

Replikasi memiliki keterbatasan-keterbatasannya sendiri (lihat bagian 6) maka tidaklah bijaksana jika kita hanya mengandalkan replikasi saja. Kita tidak pernah tahu jika keterbatasan tersebut akan kita temui dan apakah kita memiliki pemecahan yang cepat jika itu terjadi.

4.1.2 Diperlukan untuk memulihkan kesalahan eksekusi update SQL di master secara cepat

Proses replikasi mengeksekusi setiap pernyataan yang memodifikasi data dari master ke slave, termasuk perintah yang diinput oleh user secara tidak sengaja.

Jika user melakukan kesalahan update atau delete yang tidak bisa dipulihkan dengan perintah SQL, maka diperlukan prosedur khusus untuk mengembalikan data ke keadaan sebelum dieksekusinya perintah yang salah tersebut. Prosedur ini akan efektif jika kita memiliki base backup yang mutakhir.

4.2 Apa saja dan bagaimana

Prosedur base backup sama dengan yang sudah dijelaskan dibagian 2.1.5. Hanya saja prosedur tersebut dijalankan secara periodik.

4.3 Script untuk otomatisasi

Berikut ini script yang dapat digunakan untuk mengotomatisasi base backup secara periodik.

Perhatikan bahwa anda perlu mengedit pattern 'mysql-binlog' jika nama binlog yang anda gunakan berbeda.

```
#!/bin/sh

# otomatisasi base backup
# $Id: 3_0_basebackup.sh,v 1.3 2006/07/26 08:14:47 yanf Exp $
# WARN masih menggunakan nama binlog yg hardcoded: mysql-binlog
# harus diedit jika berbeda dg nama binlog anda

# dir - edit bila perlu
datadir=/mnt/data/mydata
destdir=/mnt/download/files
prefix=basebkp

# auth *sensitive* info
```

```
# harus user yg dapat lock tables - edit bila perlu
user=root
pass=1

# list dbname yg akan dibackup - edit bila perlu
dbnames='testdb mysql gsmdev5_proven'

# lokasi mysql bin - edit bila perlu
mysqlbin=/usr/local/mysql5/bin

dir='dirname $datadir'
name='basename $datadir'

export PATH=$mysqlbin:$PATH

lockcmd='FLUSH TABLES WITH READ LOCK'
unlockcmd='UNLOCK TABLES'
statuscmd='SHOW MASTER STATUS'
logincmd="mysql -u${user} -p${pass}"

# flush and lock
echo $lockcmd|$logincmd

# master status
x='echo $statuscmd|$logincmd'

# define nama archive b'dasar nama & posisi binlog
# nama binlog
binlog='echo $x| awk '{ if ($0 ~ /mysql-binlog/) {
    printf "%s", $5
} }'
# posisi binlog
pos='echo $x| awk '{ if ($0 ~ /mysql-binlog/) {
    printf "%d", $6
} }'
archive=${prefix}_${binlog}_${pos}.tar

compressed="$destdir/$archive.gz"
if [ -e $compressed ]
then
```

5 MEMULIHKAN DARI KESALAHAN EKSEKUSI UPDATE SQL DI MASTER15

```
# jangan lupa unlock
echo $unlockcmd|$logincmd
echo "sudah ada $compressed .. exit"
exit 0
fi

# archive db
cd $datadir
tar cvf $destdir/$archive $dbnames

# unlock
echo $unlockcmd|$logincmd

# compress
cd $destdir
gzip $archive
```

Perhatikan bahwa script ini berisi nama user dan password untuk database, sebaiknya set ownership file ini dengan 700.

5 Memulihkan dari kesalahan eksekusi update SQL di master

Suatu sore seorang programmer PHP yang sudah lebih dari 24 jam tidak tidur, secara tidak sengaja mengeksekusi perintah UPDATE pada tabel transaksi berisi ratusan ribu record tanpa menuliskan klausa WHERE, di server master. Bagian ini menjelaskan cara menyelamatkan diri anda sebagai DBA dan dirinya sebagai programmer, dari kemurkaan users dan atasan – tanpa harus mencari pekerjaan baru.

5.1 Mengunci master dari user

Database master pada saat ini berada dalam kondisi error, jadi perlu mencegah user melakukan modifikasi data. Perintah berikut di mysql, akan mencegah user melakukan update, sekaligus mem-flush semua perubahan yang sudah terjadi di tabel ke disk:

```
mysql>FLUSH TABLES WITH READ LOCK
```

5.2 Mencari posisi binlog terakhir

Misalkan user tersebut menjalankan perintah:

```
UPDATE transaksi_jual SET status=2
```

Maka kita perlu mencari perintah tersebut dimuali dari binlog yang sedang aktif, misal mysql-binlog.000006, gunakan utilitas mysqlbinlog sbb:

```
% mysqlbinlog mysql-binlog.000006
```

Maka akan muncul baris-baris sebagai berikut:

```
# at 232
#060725 18:15:44 server id 1  end_log_pos 260  Intvar
SET INSERT_ID=202;
# at 260
#060725 18:15:44 server id 1  end_log_pos 366  Query   thread_id=7      exec_time=0      er
SET TIMESTAMP=1153826144;
INSERT INTO t1 values (default,'zzzz...');
# at 394
#060725 18:15:44 server id 1  end_log_pos 500  Query   thread_id=8      exec_time=0      er
SET TIMESTAMP=1153826144;
UPDATE transaksi_jual SET status=2;
```

Dari file tersebut dapat kita ketahui posisi log (end_log_pos), tepat sebelum perintah UPDATE tersebut adalah 366.

5.3 Menggunakan base backup terakhir

Jika thread slave sedang berjalan, kemungkinan besar slave yang ada sekarang berada pada kondisi yang sama dengan master, yaitu mengalami kesalahan update. Jadi perlu mensetup slave baru, dengan kondisi data seperti masa lampau, sebelum terjadinya kesalahan update.

Sekarang saatnya memanfaatkan base backup yang dibuat sebelum dijalan-
kannya perintah UPDATE tersebut.

Setup slave baru dengan datadir dari base backup tersebut, seperti dijelaskan dibagian sebelumnya.

5.4 Menjalankan thread slave sampai posisi tertentu

Sekarang kita perlu menjalankan thread slave, dengan menambahkan posisi ter-
akhir sebelum thread slave tersebut berhenti, jalankan:

```
mysql>START SLAVE UNTIL master_log_file='mysql-binlog.000006',master_log_pos=366
```

5.5 Mengganti data master

Selanjutnya anda bisa memperlakukan situasi ini seperti bagian 3 “Ketika Master Crash”. Data dari slave yang sudah disinkronkan sampai posisi tertentu (366 pada contoh ini) digunakan sebagai data master.

5.6 Menjalankan thread slave seperti biasa

Setelah master beroperasi normal, jalankan thread slave seperti biasa:

1. Stop thread slave:
mysql>STOP SLAVE;
2. Start thread slave:
mysql>START SLAVE

6 Keterbatasan-keterbatasan Replikasi

Replikasi sekalipun bermanfaat dan berdaya guna, tetapi juga memiliki keterbatasan. Sebagian besar keterbatasan ini adalah inherent, dikarenakan replikasi mysql menggunakan query-level logging, yaitu master menuliskan query yang berhasil dieksekusi ke binary log, untuk selanjutnya dieksekusi oleh slave.

Berikut ini keterbatasan-keterbatasan yang kami anggap perlu diketahui, yang diambil dari Manual MySQL 5.0, bab 6, bagian 6.7. “Replication Features and Known Problems”.

1. Menambahkan kolom yang AUTO_INCREMENT ke sebuah tabel dengan perintah ALTER TABLE mungkin saja menghasilkan urutan baris yang tidak sama antara master dan slave, karena urutan bergantung pada storage engine yang digunakan dan urutan baris yang diinsert.
2. Fungsi USER(), UUID(), and LOAD_FILE() direplikasikan tanpa perubahan, sehingga tidak dapat diandalkan di slave.
3. User privilege hanya direplikasikan jika database mysql juga direplikasikan. Artinya, pernyataan GRANT, REVOKE, SET PASSWORD, CREATE USER, and DROP USER, hanya berpengaruh di slave, jika replikasi menyertakan database mysql.
4. Perbedaan data master dan slave sangat mungkin terjadi, jika pernyataan (sql) yang digunakan menghasilkan modifikasi data yang non-deterministic, sehingga hasilnya bergantung pada query optimizer. Contoh:

- (a) Pernyataan `CREATE ... SELECT` atau `INSERT ... SELECT` yang meng-insert nilai nol atau `NULL` ke sebuah kolom `AUTO_INCREMENT`.
 - (b) Pernyataan `DELETE` jika digunakan menghapus baris pada sebuah table yang memiliki foreign key dengan properti `ON DELETE CASCADE`.
 - (c) Pernyataan `REPLACE ... SELECT`, `INSERT IGNORE ... SELECT` jika ada duplikasi nilai-nilai pada data yang di-insert.
5. Walaupun table temporer direplikasikan, jika terjadi shutdown pada server slave (bukan hanya thread slave), dan tabel temporer sudah tereplikasi, tapi pernyataan update yang menggunakan replikasi tersebut belum sempat dieksekusi di slave. Maka, ketika slave direstart, tabel temporer sudah tidak ada lagi (karena 'temporer') sehingga perintah update yang menggunakan tabel temporer tersebut dipastikan akan gagal dan menghentikan thread eksekusi SQL di slave. Untuk mencegah hal ini, hindari menshutdown slave ketika tabel temporer masih open. Berikut ini prosedurnya:
 - (a) Jalankan perintah `STOP SLAVE`
 - (b) Gunakan `SHOW STATUS` untuk memeriksa nilai variabel `Slave_open_temp_tables`.
 - (c) Jika nilainya 0, berarti tidak ada tabel temporer yang terbuka, jalankan perintah `mysqladmin shutdown` untuk menshutdown server slave.
 - (d) Jika nilainya tidak 0, berarti ada tabel temporer yang terbuka, jalankan perintah `START SLAVE`.
 - (e) Ulangi prosedur ini sampai variabel `Slave_open_temp_tables` sama dengan 0, baru anda boleh menshutdown slave server.
6. Jika terjadi crash pada master, mungkin sekali terjadi posisi akhir binary log di master berada dibawah posisi terbaca oleh slave, karena binary log di master belum sempat di-*flush*. Hal ini dapat menyebabkan slave tidak dapat mereplikasi, ketika master kembali beroperasi. Untuk mencegahnya gunakan setting `sync-binlog=1`, sehingga master sesering mungkin mem-*flush* ke binary log.
7. Jika sebuah pernyataan di slave menghasilkan error maka thread SQL slave berhenti, dan slave akan menuliskan pesan error ke error log. Anda harus login ke slave secara manual dan memeriksa status slave (`SHOW SLAVE STATUS`). Lalu membereskan masalahnya, baru menjalankan thread slave dengan `START SLAVE`.

8. Nilai-nilai floating point adalah pendekatan (aproksimasi), sehingga perbandingan-perbandingan yang menggunakan nilai-nilai ini tidak pasti juga, hal ini berlaku pada operasi yang secara eksplisit menggunakan floating point atau secara implisit menggunakan nilai-nilai yang dikonversi ke floating point. Perbandingan nilai-nilai floating point mungkin saja menghasilkan perbedaan antara master dan slave, yang disebabkan perbedaan arsitektur mesin komputer, compiler yang digunakan untuk mem-build MySQL dan sebagainya.

7 Penutup

Jika data anda hilang, apapun database anda, anda tidak dapat menuntut manufaktur software database untuk mengembalikan data tersebut – bahkan software komersial bernilai ribuan dollar sekalipun. Silakan periksa di License Agreement.

Apapun database anda, pastikan tersedia prosedur backup dan recovery yang dapat diandalkan. Replikasi adalah salah satu cara membackup data yang dapat diandalkan.

MySQL memiliki sistem replikasi yang cukup andal dan mudah digunakan, jika kita dapat mengenali keterbatasan-keterbatasannya. Tidaklah bijaksana jika hanya mengandalkan replikasi semata untuk menyediakan backup, kita perlu menyediakan backup sekunder sebagai antisipasi jika data replikasi tidak dapat digunakan pada proses pemulihan.

Sebaiknya anda tidak mempelajari replikasi hanya dari tulisan ini saja, penting sekali mendalami dari sumber aslinya yaitu Reference Manual MySQL 5.0, bab 6, sebelum akhirnya memutuskan untuk menggunakan replikasi di server produksi anda.