5. SERVERUL MySQL

5.1. Prezentare generala si arhitectura MySQL

5.1.1. Ce este MySQL

MySQL este un soft de gestiune a bazelor de date (RDBMS – Relational Database Management System). O baza de date reprezinta o colectie structurata de informatie; RDBMS este softul care:

- realizeaza memorarea efectiva a datelor, in diferite formate posibile
- permite definirea si modificarea structurii datelor
- permite manipularea datelor (introducere, stergere, modificare, extragere)
- permite gestionarea accesului la date (definirea de politici de acces)

MySQL lucreaza intr-o arhitectura client-server. Serverul gestioneaza datele, iar clientii se conecteaza la server si ii transmit acestuia operatiile dorite asupra datelor sau structurii lor; clientii nu opereaza niciodata direct asupra datelor. Pentru fiecare operatie, clientul primeste de la server un rezultat care indica succesul sau esecul operatiei solicitate si care contine eventualele date cerute de client in caz de succes.

Intr-un DBMS, datele sunt memorate in unitati logice numite *tabele*. Fiecare tabela reprezinta un ansamblu de inregistrari (randuri). Introducerea datelor se face inregistrare cu inregistrare. Toate inregistrarile sunt identice ca structura, ele continand un set de coloane (numite si *atribute* sau *campuri*) specificat in definitia tabelei in momentul crearii acesteia. O definitie de tabela cuprinde numele tabelei si lista de definitii de coloane, unde fiecare coloana are nume, tip de date impus si eventuale atribute suplimentare.

Tabelele dintr-un DBMS sunt continute in *baze de date*. O baza de date indeplineste un rol asemanator cu cel al directorului/folderului din sistemul de fisiere, ea necontinand direct datele ci doar cuprinzand una sau mai multe tabele. Pentru a memora date intr-un DBMS este necesar sa cream cel putin o baza de date, iar in interiorul sau cel putin o tabela.

5.1.2. Clientii MySQL

Clientii SQL pot fi:

- aplicatii de administrare a serverului de baze de date – ele ofera utilizatorului uman o modalitate de a interfata cu serverul SQL, fie sub forma unei linii de comanda, fie sub forma unei aplicatii cu interfata grafica
- aplicatii scrise in diferite limbaje, care acceseaza sau modifica datele aflate pe server (ex: programe Java, C, PHP care folosesc date memorate in SQL etc)

Iata cateva exemple de clienti MySQL uzuali:

- aplicatii incluse in distributia MySQL
  - **mysql** – aplicatie ce ofera utilizatorului o linie de comanda cu serverul MySQL
  - **mysqladmin** – utilitar pentru realizarea de operatii administrative (creare/stergere baze de date, golire cache-uri, vizualizare si oprire procese, vizualizare setari si status)
- aplicatii aditionale pt administrare sau pt manipularea structurii/datelor
  - **MySQL Query Browser** – aplicatie grafica ce permite manipularea structurii si continutului bazelor de date aflate pe un server MySQL
  - **MySQL Administrator** – aplicatie grafica dedicata administrarii serverului (backup/restaurare, gestionare politici de acces, analiza/ajustare setari si performante etc)
  - **phpmyAdmin** – aplicatie web scrisa in PHP ce permite aproape toate tipurile de operatii cu serverul de baze de date (atat administrative cat si de manipulare a datelor)

5.1.3. Modalitati de conectare a clientilor la server

Clientii se pot afla pe aceeasi masina cu serverul sau pe o alta statie din retea. Ei se pot conecta la server prin diferite mecanisme:

1. **port TCP** – conectarea clientului la server se efectueaza prin intermediul stivei de protocoale TCP/IP. Aceasta modalitate de conectare poate fi folosita si daca clientul se afla pe aceeasi masina cu serverul, folosind adresa 127.0.0.1 sau una dintre adresele definite pe interfetele de retea ale serverului. Portul default folosit de MySQL este 3306. Aceasta modalitate de conectare poate fi dezactivata din fisierul de configurare al serverului

2. **socket** (Unix/Linux) – aplicabil doar in cazul in care clientul si serverul se afla pe aceeasi masina. Conectarea clientului la server se efectueaza prin intermediul unui fisier special de tip socket, aflat in sistemul de fisiere al serverului. Conectarea prin socket poate fi folosita atunci cand solutia TCP/IP nu este disponibila (spre exemplu, daca, din motive de securitate, aceasta din urma a fost dezactivata). In plus, conexiunea prin socket ofera performante mai bune decat cea TCP/IP, motiv pentru care ea este de preferat atunci cand clientul se afla pe aceeasi masina cu serverul. Locatia si numele fisierului socket difera usor de la o distributie Linux la alta, fiind oricum setabile din fisierul de configurare al serverului. Numele tipice ale fisierului socket sunt mysql.sock sau mysqld.sock, iar locatia sa este de obicei /var/run/mysqld/ sau /var/run/mysql/.

3. **named pipe** (Windows) – solutie aplicabila atunci cand clientul si serverul se afla pe aceeasi masina si varianta TCP/IP nu este disponibila/indicata. Ofera performante mai slabe decat TCP/IP.

4. **shared memory** (Windows) – solutie aplicabila pentru client si server aflate pe aceeasi statie Windows cand solutia TCP/IP este indisponibila/nedorita

Oricare ar fi modalitatea de conectare la server, clientul trebuie de obicei sa furnizeze un username si o parola pentru a i se permite accesul. In cazul clientilor grafici, cele doua sunt configurate folosind optiuni din meniurile acestora; pentru clientii de shell, ele se specifica sub forma de optiuni in linia de comanda. Iata exemple care utilizeaza clientul de shell *mysql*:

```
# conectare la serverul MySQL de pe statia locala, folosind contul root fara parola:
mysql -u root

# conectare la serverul MySQL de pe statia locala, folosind contul root cu parola;
# parola este ceruta la executarea comenzii:
mysql -u root -p

# conectare la serverul MySQL ce ruleaza pe statia 10.0.0.100, cu contul sql si parola
mysql -h 10.0.0.100 -u root -p
```

5.1.4. Motoare de stocare a datelor MySQL

Un motor de stocare reprezinta un modul al serverului MySQL care efectueaza memorarea datelor pe hard-disk intr-o anumita forma (ex: unul sau mai multe fisiere sau partitii) si cu anumite caracteristici (ex: facilitati tranzactionale etc). Fiecare tabela are asociat un anume storage engine, ales la crearea tabelei. Alegerea facuta va afecta, printre altele:

- viteza diverselor operatii de manipulare a datelor ce vizeaza acel tabel
- modalitatea de stocare in sistemul de fisiere a datelor corespunzatoare tabelei (date, indecsi etc.)
- posibilitatea de a folosi tranzactii[1]

1 Tranzactie – un set de operatii care actioneaza ca o singura unitate: fie reusesc in integralitatea lor, producand modificarile corespunzatoare, fie, daca una dintre operatii esueaza, nu este produsa nicio modificare. Ex: daca o tranzactie contine 4 operatii, din care primele doua se executa cu succes si a treia esueaza, eventualele modificari produse de primele doua vor fi anulate

- posibilitatea de a asigura integritatea referentiala a datelor, prin declararea explicita a cheilor externe
- mecanismele de locking (serializare a accesului la date), acestea putand fi aplicate la nivel de intreaga tabela sau simpla inregistrare

Printre motoarele de stocare disponibile se numara:

- **MyISAM** – motorul default pentru MySQL. Dezavantaje: nu suporta tranzactii si foloseste locking la nivel de tabela pentru serializarea accesului. Avantaje: poate fi folosit in crearea tabelelor de tip MERGE si in plus suporta indecsi de tip FULLTEXT
- **InnoDB** – motor de stocare tranzactional, ce permite declararea explicita a cheilor externe si impunerea integritatii referentiale. Locking-ul se face la nivel de inregistrare, ceea ce il face foarte potrivit pentru scenarii cu update-uri multiple concurente
- **MEMORY** – motor in cazul caruia toate datele si indecsii tabelelor stau in memorie. Are avantajul de a fi cea mai rapida forma de stocare. Tabelele ce folosesc acest motor nu pot avea coloane de tip TEXT sau BLOB. **Atentie! Datele dintr-o astfel de tabela se pierd la restart de server, ramanand memorata numai definitia tabelei!**
- **FEDERATED** – motor nou-introdus in MySQL 5, ce permite unui server MySQL sa acceseze tabele aflate pe alte servere si sa le prezinte clientilor sai ca si cum ar fi ale sale, astfel incat clientii sa nu mai aiba nevoie sa se conecteze direct la acele servere. Nu suporta tranzactii.
- **MERGE –** reprezinta o uniune de tabele cu structura identica, ce actioneaza ca o interogare de tip UNION. In acest fel se poate crea o tabela ce depaseste dimensiunea maxima admisibila pentru MyISAM
- **NDBCluster** – permite crearea unei baze de date distribuite pe mai multe noduri MySQL ce actioneaza ca un tot unitar. Datele sunt stocate in memoria RAM a statiilor ce compun clusterul, insa, spre deosebire de motorul MEMORY, modificarile sunt scrise pe HDD pentru a nu se pierde in caz de repornire a clusterului. Reprezinta o solutie scalabila, redundanta, ce asigura disponiblitate maxima a datelor.

Suportul pentru motorul MyISAM este inclus intotdeauna in MySQL, insa celelalte pot lipsi - fie din cauza ca nu au fost incluse in distributie la compilarea serverului, fie fiindca au fost dezactivate din fisierul de configurare.

Nota: *lista de motoare suportate se poate obtine cu comanda SQL* SHOW ENGINES;

5.1.5. Locatia datelor si fisiere de control

Datele memorate in bazele de date ale unui server MySQL se afla in locatii diferite in functie de motorul de stocare folosit pentru tabela din care ele fac parte. La configurarea serverului se specifica directorul ce va constitui radacina datelor; fiecare baza de date creata va determina aparitia aici a unui subdirector. Inauntrul acestuia va fi creat cate un fisier .frm pentru fiecare tabela din baza de date respectiva, fisier ce poarta numele tabelei si contine datele despre ea (motor de stocare folosit si diverse alte informatii generale). In cazul tabelelor de tip MERGE este creat un fisier .mrg.

In functie de motor, datele din tabela se pot gasi:

- **MEMORY**: datele sunt stocate in memoria RAM statiei
- **MyISAM**: datele sunt memorate intr-un fisier .myd aflat in acelasi director cu cel .frm. In acelasi director poate aparea si un fisier .myi, ce contine indecsii atasati acelei tabele, daca acestia exista
- **InnoDB**: exista doua variante de configurare:
    a. datele tuturor tabelelor InnoDB stau intr-un spatiu de stocare comun (asa-numitul “tablespace”), care poate fi format dintr-unul sau mai multe fisiere sau partitii. Numarul,dimensiunea si locatia fisierelor sunt stabilite din fisierul de configurare. Tablespace-ul nu contine numai datele, ci si indecsii, modificarile produse de tranzactiile in desfasurare care inca nu au facut commit etc.

b. fiecare tabela are propriul fisier cu extensia .ibd alaturi de fisierul .frm corespunzator. Atentie insa! Chiar daca serverul a fost configurat sa creeze cate un fisier/tabel InnoDB, tablespace-ul comun este in continuare necesar pentru memorarea altor date (dictionarul datelor, segmentul de rollback etc)

5.2. Instalare MySQL si programe conexe

5.2.1. Versiuni de server in varianta precompilata

Pana la versiunea 5.0 existau doua distributii de server: MySQL Standard si MySQL Max. Prima cuprindea cele mai des intalnite facilitati si motoare de stocare, pe cand cea de-a doua includea si motoare de stocare mai putin folosite. Incepand cu MySQL 5.1 site-ul mysql.com ofera o distributie unica, ce corespunde fostului MySQL Max. Unele distributii Linux continua insa sa ofere pachete separate mysql si mysql-max, iar unele plaseaza in pachete separate si suportul pentru NDB (noua tehnologie de clustering introdusa odata cu MySQL 5). De aceea este necesara analizarea listei de pachete disponibile si instalarea celor potrivite pentru scopul propus.

5.2.2. Modalitati de instalare MySQL

MySQL poate fi instalat in 3 moduri:

- **prin compilare din surse.** Sursele mysql sunt disponibile pe www.mysql.com
- **folosind o varianta precompilata**
  - **cea disponibila pe www.mysql.com**. Aceasta varianta are avantajul de a fi fost compilata folosind setari de finete si compilatoare de nivel comercial, dar poate avea incompatibilitati cu versiunea de Linux utilizata (mysql.com ofera pachete precompilate numai pentru cateva distributii mai cunoscute)
  - **cea inclusa in distributia Linux utilizata**, sub forma de pachete precompilate. In functie de distributie, MySQL poate fi compus din unul sau mai multe pachete. Lista de pachete al caror nume contine fragmentul *mysql* poate fi determinata cu:

```
apt-cache search mysql          # Debian si distributii derivate
urpmq -y mysql                  # Mandriva si alte distributii ce folosesc urpm*
```

Spre exemplu, in Debian Lenny este necesara instalarea pachetelor mysql-server, mysql-common, mysql-client, libmysqlclient; in Mandriva, pachetele sunt mysql, mysql-common, mysql-client, fiind disponibile in plus pachetul mysql-max si pachete mysql-ndb-*.

5.2.3. Instalarea software-ului aditional

Serverul MySQL include utilitare pentru administrarea serverului si manipularea structurii si informatiei din baza de date, insa ele functioneaza din linia de comanda. De aceea este deseori mai comod si eficient sa se foloseasca softuri aditionale, cu interfata grafica, precum MySQL Query Browser, MySQL Administrator, phpMyAdmin. In plus, ne putem folosi de aplicatii aditionale pentru analiza performantelor serverului si optimizarea acestuia (ex: mysqltuner sau mysql_performance_tuning_primer).

In Debian, pachetele necesare sunt mysql-gui-tools-common , mysql-admin, mysql-query-browser, mysqltuner. In Mandriva avem mysql-gui-tools, mysql-administrator,mysql-query-browser,mysqltuner, mysql_performance_tuning_primer.

5.3. Configurarea de baza a serverului

5.3.1. Fisiere de configurare

Serverul MySQL poate fi configurat pasandu-i-se la pornire toate optiunile necesare. Acest mod de lucru nu este insa intotdeauna de dorit sau optim, de aceea se prefera deseori varianta unui fisier de configurare in care sunt salvate optiunile dorite.

Serverul MySQL cauta fisierul de configurare in cateva locatii prestabilite, in aceasta ordine:
1. **/etc/my.cnf** – locatia principala a fisierului cu optiuni
2. **/etc/mysql/my.cnf** – locatie inclusa incepand cu Mysql 5.1.15
3. **$MYSQL_HOME/my.cnf** – fisierul este cautat aici numai daca variabila MYSQL_HOME este definita
4. **~/.my.cnf** – fisier de configurare aflat in directorul personal al utilizatorului care ruleaza serverul. Desi suportata de server, aceasta modalitate de configurare este de evitat, deoarece setarile serverului nu ar trebui sa depinda in vreun fel de utilizatorul care il ruleaza

O buna parte a optiunilor utilizabile in aceste fisiere se aplica numai pentru anumite motoare de stocare folosite pentru tabelele memorate pe server.

5.3.2. Formatul general al fisierului de configurare

Fisierul *my.cnf* este format directive de configurare, care pot avea doua forme:

```
# directiva ca primeste o valoare; corespunde unei optiuni cu parametru la rularea serverului
directiva = valoare

# cazul unei optiuni fara parametru
directiva
```

Directivele sunt impartite in sectiuni. Fiecare sectiune incepe printr-un titlu de forma *[titlu_sectiune]*:

```
[nume_sectiune]
directiva1 = valoare1
directiva2 = valoare2
directiva3
...
```

Numele de sectiuni corespund cu cele ale diferitelor utilitare mysql care se folosesc de acest fisier. Fiecare utilitar isi cauta propria sectiune si isi citeste optiunile de acolo. Sectiuni tipice sunt:
- [mysqld] – sectiunea principala, de configurare a serverului
- [client] – sectiune ce se aplica tuturor programelor client
- [mysql] – sectiunea clientului mysql (cel care ofera linie de comanda cu serverul)
- [manager] – sectiunea dedicata lui mysqlmanager (programul de manipulare a instantelor MySQL)

Fiecare directiva din fisierul de configurare are de obicei o optiune corespondenta, cu acelasi nume, in linia de comanda. Spre exemplu, pentru a determina serverul sa asculte numai pe adresa IP de loopback, il putem porni apeland direct executabilul *mysqld* cu optiunea *--bind-address=127.0.0.1*, sau putem crea o sectiune *[mysqld]* in fisierul de configurare in care sa plasam directiva *bind-address=127.0.0.1*.

5.3.3. Configurarea variantelor de conectare la server

Variantele de conectare la server pot fi gestionate cu ajutorul urmatoarelor directive:

```
# locatia si numele fisierului socket
socket              = /var/run/mysqld/mysqld.sock

# conectare TCP/IP: adresa si portul pe care asculta serverul
bind-address = 127.0.0.1
port         = 3306

# ...sau, pentru dezactivare TCP/IP
skip-networking
```

5.3.4. Contul folosit pentru rularea serverului

Ca orice alt server, MySQL trebuie configurat sa ruleze cu un cont neprivilegiat. Contul din oficiu folosit este *mysql* dar administratorul poate alege orice alt cont, atata timp cat seteaza permisiunile corecte pe fisierele folosite de server. Oricare ar fi contul utilizat, acesta trebuie creat inaintea pornirii serverului; specificarea sa in fisierul de configurare de realizeaza in sectiunea [mysqld] sau [mysqld_safe] astfel:

```
user   = mysql
```

5.3.5. Locatia datelor

Dupa cum s-a explicat anterior, modul in care serverul va stoca datele diverselor tabele depinde de storage engine-ul folosit pentru fiecare tabela in parte; MyISAM creeaza cate unul sau mai multe fisiere pentru fiecare tabela, pe cand InnoDB poate folosi un spatiu comun pentru toate tabelele sale si indecsii acestora.

In cazul InnoDB, spatiul comun (tablespace-ul) poate fi compus din:

- unul sau mai multe fisiere. Fiecare fisier poate avea dimensiune fixa sau variabila. Atunci cand fisierul are dimensiune variabila, se poate preciza dimensiunea initiala si cea maxima. Trebuie insa constientizat faptul ca un fisier a carui dimensiune creste in timp poate fi subiect de fragmentare in multe sisteme de fisiere; in plus, fiecare sistem de fisiere are o dimensiune maxima de fisier care este in general inferioara dimensiunii maxime de tabela suportata de MySQL. Ultima restrictie poate fi evitata compunand tablespace-ul din mai multe fisiere (vezi exemplul de mai jos)
- una sau mai multe partitii raw (neformatate). Folosirea de partitii elimina restrictiile legate de dimensiunea maxima a fisierelor, existente in sistemele de fisiere uzuale.

```
# locatia instalarii de MySQL (constituie prefixul pentru caile relative ce apar in restul
fisierului)
basedir      = /usr

# directorul ce contine fisierele corespunzatoare bazelor de date
datadir             = /var/lib/mysql

# directorul in care se creeaza tabelele temporare
tmpdir       = /tmp

# directorul in care vor sta toate fisierele de date InnoDB (default: valoarea lui datadir)
innodb_data_home_dir = /var/lib/mysql/innodb

# locatia fisierelor ce compun tablespace-ul, relativa la innodb_data_home_dir
# primul fisier va fi creat cu dimensiune initiala de 10MB, putandu-se expanda in functie de
# necesitati pana la maxim 2GB; al doilea are dimensiune fixa
innodb_data_file_path = innodb1:10M:autoextend:max:2G;innodb2:500G

# cantitatea alocata suplimentar la fiecare largire a tablespace-ului
innodb_autoextend_increment = 8M
```

5.3.6. Configurare logging

MySQL poate genera urmatoarele tipuri de loguri:

- **general log** – inregistreaza toate conectarile clientilor la server si interogarile trimise de catre acestia. Este cel mai cuprinzator log, insa in acelasi timp si cel care capata proportii cel mai rapid
- **binary log** – inregistreaza doar interogarile care modifica date sau structura. Poate fi folosit pentru a restaura bazele de date dupa un crash, sau pentru replicarea datelor intre mai multe servere. Fisierele generate sunt intr-un format binar, ele putand fi analizate cu ajutorul utilitarului *mysqlbinlog*.
- **error log** – inregistreaza mesajele legate de pornirea/oprirea serverului si eventualele erori aparute
- **slow query log** – inregistreaza interogarile a caror durata de executie depaseste un anumit prag (configurabil). Permite reperarea interogarilor ce necesita optimizare. Se vizualizeaza cu utilitarul *mysqldumpslow*.
- **relay log** – in cazul unui server care joaca rolul de replication slave, modificarile primite de la serverul master si care asteapta sa fie efectuate sunt memorate temporar in relay log. Relay log-ul este un log binar, care poate fi vizualizat tot cu ajutorul comenzii *mysqlbinlog*

Logurile sunt ocazional sau periodic *rotite*: logul curent este redenumit si se deschide in locul sau in fisier nou, gol. Rotirea de loguri se poate face:

- automat – conditiile de rotire putand fi specificate din fisierul de configurare mysql. Este cazul directivei max_binlog_size din exemplul de mai jos
- manual - la comanda administratorului. Exista cel putin doua modalitati de a determina rotirea logurilor:
  - comanda *mysqladmin flush-logs*
  - instructiunea SQL *FLUSH LOGS*

```
# ***** GENERAL LOG ********

log    = /var/log/mysql/mysql.log

# ****** BINARY LOG **********

# locatie fisier log
log-bin      = /var/log/mysql/mysql-bin
# fisierele create vor fi de forma mysql-bin.CCCCCC (C=cifra)

# dimensiunea maxima a fiecarui fisier; daca logul atinge aceasta dimensiune este rotit
max_binlog_size = 500M     # maximul este de 1G, care constituie in acelasi timp si default-ul

# ******* ERROR LOG **********

log-error    = /var/log/mysql/mysql.err

# ******* SLOW QUERY LOG **********

log-slow-queries    = /var/log/mysql/mysql.slow      # inainte de MySQL 5.1.29
slow_query_log_file = /var/log/mysql/mysql.slow      # dupa 5.1.29

# cat trebuie sa dureze o interogare pt a fi considerata "slow"? (in secunde)
long_query_time     = 20
```

5.4. Operatii administrative

5.4.1. Pornirea si oprirea serverului

Exista mai multe modalitati de a porni serverul MySQL:

– prin rularea directa a executabilului mysqld. Este o solutie potrivita pentru diagnosticare, insa este preferabil ca in regim de functionare normal serverul sa fie pornit printr-una din celelalte metode

- prin intermediul scriptului mysqld_safe. Acesta citeste optiunile din sectiunile [mysqld],[mysqld_safe] si [server] ale fisierului my.cnf, si impreuna cu optiunile primite la apelare le paseaza executabilului mysqld. Scriptul mysqld_safe adauga masuri de siguranta precum restartarea serverului in caz de eroare
- prin intermediul scriptului de rulare a serviciului. Este cazul instalarii serverului din pachete precompilate, cand se instaleaza in /etc/init.d scriptul de management al serviciului MySQL. Acesta accepta ca argumente start, stop sau restart:

```
/etc/init.d/mysqld start

# alternativ, in distributii care dispun de comanda service:
service mysqld start
```

Pentru oprirea serverului:
- daca exista script de management al serviciului, poate fi apelat acesta cu argumentul stop:

```
/etc/init.d/mysqld stop
```

- daca se doreste oprirea manuala a serverului, se poate efectua acest lucru prin trimiterea unui semnal. Nu uitati insa sa opriti intai mysqld_safe, deoarece acesta restarteaza serverul mysqld in caz de oprire
- daca este disponibil utilitarul mysqladmin, acesta poate fi de asemenea folosit pentru oprirea serverului:

```
mysqladmin shutdown
```

5.4.2. Backup si restaurare

Tipuri de backup

In MySQL administratorul poate utiliza urmatoarele tipuri de backup:

- **backup binar** – presupune copierea ca atare a fisierelor care compun bazele de date si tabelele serverului. Solutia are avantajul de a fi rapida in comparatie cu celelalte variante, insa nu functioneaza pentru toate storage engine-urile pentru ca nu toate creeaza fisiere al caror format este independent de platforma
- **backup in format text**
  - crearea unui script de restaurare a datelor – fisierul text creat va contine instructiunile SQL care reconstituie bazele de date si tabelele dorite
  - prin exportarea datelor – solutie nerecomandata, deoarece nu salveaza si informatiile legate de structura sau relatiile dintre diversele tabele
- **replicarea datelor pe mai multe servere SQL** – reprezinta o solutie prin care datele unui server master pot fi duplicate pe unul sau mai multe servere slave, mentinand serverele slave permanent sincronizate cu master-ul prin intermediul unui protocol de comunicatie in retea. Aceasta solutie are dezavantajul configurarii mai speciale necesare si a resurselor suplimentare folosite, pe de alta parte insa serverele slave pot asigura atat redundanta, cat si load sharing pentru serverul master

Utilitare folosite pentru backup si restore si utilizarea acestora

MySQL pune la dispozitia administratorului urmatoarele utilitare:

- **mysqldump** – folosit pentru exportarea uneia sau mai multor baze de date – fie exclusiv a datelor, fie a intregii structuri si a datelor memorate in aceasta. In ultimul caz, mysqldump produce un script SQL care, prin rularea pe un alt server, reconstituie datele exportate

- **mysqlimport** – **NU este complementul lui mysqldump!** Utilitarul mysqlimport poate doar sa importe date din fisiere text, insa nu este utilizat pentru rularea de scripturi generate cu mysqldump!
- **mysql** – utilitarul mysql, prezentat pana acum ca modalitate de a obtine o linie de comanda interactiva cu serverul, are si capabilitatea de a rula in mod non-interactiv – si anume pentru rularea de scripturi SQL.

Exemple:

```
# exportarea bazei de date produse si plasarea scriptuklui rezultat in fisierul /tmp/script.sql
mysqldump -u root -p produse > /tmp/script.sql

# rularea aceluiasi script pe serverul aflat pe statia 10.0.0.5
# (urmatoarele doua comenzi sunt echivalente)
cat /tmp/script.sql | mysql -u root -p
mysql -h 10.0.0.5 -u root -p < /tmp/script.sql
```

5.4.3. Analiza si modificare variabile si informatii de stare

Categorii de variabile

Serverul MySQL mentine o serie de variabile, vizualizabile (si unele modificabile) de catre administrator. Acestea se impart in:

- **variabile de sistem** – reprezinta setari ale serverului. Unele pot fi modificate la nivel de intreg server (efectul aplicandu-se astfel tuturor clientilor), altele pot fi setate per-sesiune, aplicandu-se numai clientului ce a operat modificarea
- **variabile de stare** – reprezinta diferite valori/statistici ce tin de functionarea serverului, pe care administratorul le poate utiliza pentru a analiza performantele serverului si a determina eventualele optimizari necesare. Aceste variabile sunt mai degraba informative, nefiind gandite pentru modificare

Variabile de sistem

Variabilele de sistem se impart in doua categorii:

- variabile de sistem **statice** - se pot seta ca optiuni date serverului la pornire sau incluse in fisierul de configurare, neputand fi modificate ulterior. Exemplu: datadir, bind-address, locatie loguri etc.
- variabile de sistem **dinamice** - se pot modifica in timpul rularii, prin instructiuni SQL. Ele pot fi la randul lor globale sau locale. Exista variabile care se pot seta atat la nivel global, cat si local, caz in care valoarea globala joaca rolul de default pentru cea de sesiune
  - globale = se aplica la nivel de intreg server. Modificarea unei astfel de variabile afecteaza toti clientii
  - locale = per sesiune (conexiune a unui client). Modificarea unei astfel de variabile afecteaza doar acea sesiune a acelui client.

Vizualizarea variabilelor de sistem de poate efectua folosind:

- MySQL Administrator → Health → Server variables
- instructiuni SQL:
  - variabile de sesiune: *SHOW <LOCAL | SESSION> VARIABLES <LIKE '...%fragment_nume%...'>*
  - variabile globale: *SHOW GLOBAL VARIABLES <LIKE '...%fragment_nume%...'>*
- mysqladmin variables (comanda de shell)

Variabile de stare

Variabilele de stare ofera informatii si statistici despre functionarea serverului. Pe baza lor evaluam performanta serverului si decidem daca e cazul sa schimbam variabilele de sistem.

Vizualizarea variabilelor de stare se poate realiza astfel:
- utilitare
  - GUI-based: MySQL Administrator → Health → Status variables
  - shell: *mysqladmin extended-status*
- instructiuni MySQL
  - *SHOW STATUS <LIKE '...%fragment_nume_variabila%...'>*

Iata cateva categorii de variabile de stare de interes:
- **com_*** - statistici legate de tipurile de comenzi SQL primite de server
- **innodb_*** - statistici legate de motorul innodb. Complementeaza informatiile aratate de comanda SHOW ENGINE INNODB STATUS.
- **qcache_*** - statistici legate de query cache (cache-ul in care sunt mentinute rezultatele interogarilor anterioare)
- **key_*** - statistici legate de utilizarea bufferului pentru indecsi MyISAM
- **sort_*** - statistici legate de operatii de ordonare
- **threads_*** - informatii legate de conexiuni (MySQL porneste cate un thread pentru fiecare client care se conecteaza)

5.4.4. Tuning si optimizare

Directii de actiune

Optimizarea functionarii unei baze de date se indreapta in cel putin doua directii:

1. **Optimizarea structurii bazei de date**. Cade in primul rand in sarcina developer-ului si presupune aspecte precum:
   - un design corect al structurii bazei de date
     - folosirea tipurilor de date cele mai potrivite pentru datele memorate
     - eliminarea redundantei prin aducerea tabelelor in formele normale
     - folosirea de indecsi pentru sporirea vitezei operatiilor de extragere a datelor (si nu numai)
   - optimizarea interogarilor
     - formularea unor interogari care sa foloseasca pe cat posibil indecsii
     - gasirea echilibrului intre operatiile efectuate de catre serverul de baze de date si cele efectuate de catre aplicatia client

2. **Optimizarea setarilor serverului** – se concentreaza in primul rand pe reglarea diverselor buffere astfel incat sa se minimizeze lucrul cu hard disk-ul. Setarile difera in general de la un storage engine la altul, deoarece acestea administreaza in mod diferit stocarea fizica a datelor pe hard-disk. Setarile uzuale cuprind:
   - reglarea bufferelor pentru indecsi. Indecsii unei tabele isi arata adevarata eficienta in masura in care sunt stocati, in cat mai mare proportie, in memoria RAM
   - reglarea cache-ului de interogari – rezultatele interogarilor efectuate des pot fi memorate, partial sau complet, intr-un cache, astfel incat la repetarea lor timpul si resursele consumate sa fie reduse la un minim
   - reglarea bufferelor folosite pentru optimizarea diferitelor alte operatii cu tabele (deschiderea tabelelor, citirea lor secventiala, ordonarea datelor unei interogari, join-uri etc)

In afara de analiza directa a variabilelor de stare ale serverului, exista utilitare care ajuta administratorul in a-si forma o imagine asupra corectitudinii configurarii, unele dintre ele oferind si sfaturi de ajustare a unora dintre parametrii de configurare:

- Mandriva: mysql_performance_tuning_primer (nume pachetului si al utilitarului)
- Debian, Mandriva: mysqltuner (nume pachet&utilitar)

Reglaje MyISAM

Principala setare a motorului MyISAM este optiunea/variabila **key_buffer_size**. Aceasta seteaza dimensiunea bufferului RAM folosit pentru pastrarea indecsilor tabelelor. In mod normal, indecsii stau in fisierele .myi atasate unei tabele MyISAM; prin copierea lor in RAM se obtine un semnificativ spor de viteza al operatiilor care implica acesti indecsi.

Nota: *versiunile mai noi de MySQL ofera posibilitatea definirii a mai multe buffere independente, administratorul putand stabili la nivel de fiecare index al unei tabele care dintre buffere va fi folosit.*

Dimensiunea key buffer-ului se alege in functie de eficienta si gradul de ocupare a cache-ului, putand merge pana la mai mult de 50% din memoria serverului pentru o statie dedicata. Pentru a face o alegere corecta se pot analiza urmatoarele variabile de stare/de sistem:

- variabile de sistem
  - **key_cache_block_size** – dimensiunea unui bloc de date al bufferului de indecsi. Alocarea spatiului in acest buffer se efectueaza bloc cu bloc
- variabile de stare
  - **key_read_requests** – numarul de cereri de citire a unui bloc de date din cache-ul de indecsi efectuate de server de la pornire pana in momentul curent
  - **key_reads** – numarul de citiri ale unui bloc de date al unui index direct de pe hard-disk, in urma absentei in buffer a acelui bloc
  - **key_blocks_unused** – numarul de blocuri de date nefolosite ale bufferului de indecsi

Exista doua calcule simple care pot fi facute pentru a evalua eficienta si gradul de folosire al key buffer-ului:

- **eficienta bufferului** - ne dorim ca numarul de citiri fizice sa fie cat mai mic in raport cu numarul de cereri de citire – altfel spus, majoritatea cererilor sa fie servite din buffer, fara accesarea hard-disk-ului. Eficienta cache-ului (exprimata in procente) se calculeaza ca:

```
eficienta = (1 - (key_reads / key_read_requests))*100
```

Valoarea obtinuta ar trebui in mod normal sa se situeze peste 90-95%. Daca eficienta este joasa este necesara cresterea valorii lui key_buffer_size. Acesta nu trebuie insa crescut excesiv, deoarece:

- daca serverul epuizeaza memoria RAM disponibila va incepe sa foloseasca swap, ceea ce duce la o degradare semnificativa a performantelor
- trebuie sa raman memorie RAM disponibila atat pentru alte storage engine-uri, cat si pentru sistemul de operare si celelalte procese care ruleaza pe statia in cauza

- **gradul de ocupare a bufferului** – ne dorim ca bufferul sa fie acoperitor pentru nevoile serverului, dar sa nu stea nefolosit. Este indicat un grad de ocupare de peste 80%. Gradul de ocupare al bufferului (in procente) se poate calcula ca:

```
grad_ocupare = (1 - (Key_blocks_unused * key_cache_block_size)/key_buffer_size) * 100
```

Reglaje InnoDB

Motorul InnoDB, spre deosebire de MyISAM, foloseste un cache comun pentru indecsi si pentru datele citite din tabele. Avand in vedere acest aspect, este necesara o anvergura mai mare a acestui buffer, putand merge pana la 80% din memoria RAM a statiei pentru un server dedicat (cu aceleasi avertizari insa ca in cazul MyISAM!).

In plus, InnoDB este un motor tranzactional; modificarile sunt pastrate in memorie si nu sunt efectuate decat daca toate operatiile ce compun tranzactia se incheie cu succes. Operatiile intermediare sunt memorate la randul lor intr-un buffer; daca acest buffer se umple, este necesara scrierea unei parti a informatiei pe hard-disk, lucru de care trebuie sa ne ferim pe cat posibil.

Variabilele de interes sunt:
- variabile de sistem
  - **innodb_buffer_pool_size** – valoarea sa stabileste dimensiunea bufferului amintit
  - **innodb_log_buffer_size** – reprezinta dimensiunea bufferului in care sunt memorate operatiile corespunzatoare tranzactiilor in desfasurare
- variabile de stare
  - **Innodb_page_size** – dimensiunea blocului de date folosit la alocarea spatiului in bufferul InnoDB
  - **innodb_buffer_pool_pages_free** – numarul de pagini ale bufferului nefolosite la momentul curent
  - **innodb_buffer_pool_pages_total** – numarul total de pagini ale bufferului. Inmultind aceasta valoare cu cea a lui Innodb_page_size trebuie sa obtinem valoarea lui innodb_buffer_pool_size

Nota: *statistici despre functionarea motorului InnoDB pot fi obtinute cu instructiunea* SHOW ENGINE INNODB STATUS.

5.5. Administrarea conturilor si a privilegiilor acestora

5.5.1. Sistemul de privilegii MySQL

Serverul MySQL distinge clientii conectati prin doua carateristici:
- username-ul furnizat. Acesta poate fi o secventa alfanumerica, sau, daca serverul o permite, sirul vid. In acest ultim caz avem de a face cu accesul anonim. Fiecare cont de client poate avea atasata o parola, cu ajutorul careia serverul se poate asigura de identitatea acelui client (autentificare)
- statia de pe care se conecteaza aplicatia client. Aceasta poate fi specificata sub forma de adresa IP sau nume DNS

Aceste doua ingrediente formeaza asa-numitul cont al clientului; odata clientul autentificat, setul de privilegii care i se aplica va depinde atat de username-ul furnizat, cat si de statia de pe care clientul se conecteaza.

Fiecarui cont i se pot atribui privilegii:
- globale – se aplica la nivel de intreg server - implicit tuturor bazelor de date, tabelelor si coloanelor acestora
- per baza de date, per tabela sau chiar per coloana – fiecarui cont i se pot atasa privilegii la oricare dintre aceste niveluri.

5.5.2. Cum se memoreaza conturile si privilegiile

Conturile MySQL si privilegiile atasate acestora se memoreaza in tabele din baza de date *mysql*; iata principalele tabele utilizate:

- **user** – contine caracteristicile conturilor de client (identificate prin coloanele Host, User si Password) si privilegiile globale atasate fiecarui cont
- **db** – contine privilegiile per baza de date atasate clientilor
- **tables_priv** – contine privilegiile la nivel de tabela atasate clientilor
- **columns_priv** – idem pentru privilegii la nivel de coloana

5.5.3. Cum se administreaza conturile si privilegiile

Toate aspectele legate de administrarea de conturi pot fi gestionate in doua moduri:

- folosind un utilitar grafic sau web-based, care permite administratorului o modalitate facila si familiara de specificare a conturilor si privilegiilor dorite. Un astfel de utilitar actioneaza pe post de client al serverului MySQL, trimitandu-i acestuia instructiunile SQL corespunzatoare operatiilor cerute de catre utilizatorul uman. Exemple de astfel de clienti: MySQL Administrator, phpMyAdmin
- folosind un utilitar de tip client MySQL care permite executarea de instructiuni MySQL in mod interactiv pe server. Exemplu: mysql, MySQl Query Browser, etc.

Instructiunile SQL folosite pentru administrarea conturilor si privilegiilor sunt:

- CREATE USER, DROP USER, RENAME USER – folosite pentru creare/stergere/redenumire cont
- GRANT – folosit pentru atribuirea de privilegii unui cont, putand si crea contul automat daca este cazul
- REVOKE – elimina privilegii ale unui cont existent
- SET PASSWORD – atribuie/schimba parola unui cont existent
- SHOW GRANTS – afiseaza privilegiile unui cont existent

5.5.4. Modificarea parolei unui cont existent si securizarea conturilor din oficiu

Instalarea din oficiu a serverului MySQL contine deseori cateva conturi deja create, gandite sa permita conectarea initiala la server si administrarea acestuia. Analizand exemplul (real!) din tabela alaturata, observam ca niciunul dintre aceste conturi nu are parola. Aceasta reprezinta un risc de securitate moderat: pe de o parte nu este permisa conectarea decat de pe statia locala, insa pe de alta parte orice utilizator al acelei statii se va putea conecta la serverul MySQL cu contul root, care are privilegii depline! De aceea problema trebuie remediata cat mai curand dupa instalare, prin stabilirea de parole pentru contul de root si prin eventuala desfiintare a contului cu acces anonim.

User	Host	Password
root	localhost	
root	127.0.0.1	
	localhost	

Exista diferite modalitati de a schimba parola unui cont:

- dintr-un client MySQL grafic sau web-based
- din linia de comanda Linux:
  - folosirea comenzii mysqladmin cu subcomanda password, ca in exemplul urmator:

```
mysqladmin -u root  password parolanoua
```

- dintr-un client MySQL care ne ofera posibilitatea de a trimite comenzi serverului in mod interactiv (ex: clientul *mysql*):
  - folosind instructiunea SET PASSWORD
  - folosind instructiunea GRANT – nu este intotdeauna solutia ideala, deoarece GRANT este folosit in general pentru acordarea de privilegii, posibilitatea de schimbare a parolei fiind doar un efect colateral
  - folosind instructiunea UPDATE si modificand direct tabela User (contraindicat)

Exemple:

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('mypass');
GRANT USAGE ON *.* TO 'root'@'127.0.0.1' IDENTIFIED BY 'mypass';
```

5.5.5. Crearea conturilor MySQL

Un cont MySQL este reprezentat in instructiunile SQL care-l administreaza sub forma *'username'@'statie'*. Daca username-ul lipseste, clientii se vor putea conecta de pe statia specificata fara a furniza username. Statia poate contine wildcard-ul % care inlocuieste 0 sau mai multe caractere, sau poate fi un intreg subnet, fiind astfel posibile urmatoarele forme:

- **'sql'@'10.0.0.4'** – clientii care se conecteaza de pe statia 10.0.0.4 folosind username-ul *sql*
- **'sql'@'10.0.0.%'** - clientii care se conecteaza de pe orice statie a carei adresa IP incepe cu 10.0.0. (echivalentul notatiei 10.0.0.0/24 din retelistica)
- **'sql'@'10.0.0.0/255.255.255.0'** – acelasi efect ca exemplul anterior

Crearea conturilor SQL se poate realiza:

- prin intermediul unui client MySQL grafic sau web-based
- prin instructiuni SQL adresate serverului de pe un client. Instructiunea folosita este CREATE USER, cu forma generala *CREATE USER 'username'@'statie' IDENTIFIED BY 'parola'*, ca in exemplele urmatoare:

```
CREATE USER 'ana'@'192.168.0.3' IDENTIFIED BY 'mypass';
CREATE USER 'vlad'@'192.168.0.%' IDENTIFIED BY 'the_impaler';
CREATE USER 'brad'@'192.168.0.0/255.255.255.0' IDENTIFIED BY 'pitt';
```

Nota: *clientul trebuie sa se conecteze la server folosind numele exact al statiei specificat in contul sau! Spre exemplu, daca un server MySQL are doar contul 'root'@'localhost', conectarea cu clientul mysql folosind comanda mysql -u root -h 127.0.0.1 va esua!*

5.5.6. Acordarea de privilegii unui cont MySQL

Acordarea privilegiilor pentru un cont se realizeaza fie folosind utilitare grafice sau web-based, fie interactionand direct cu serverul si transmitandu-i instructiunile SQL corespunzatoare. Instructiunea SQL pentru acordarea de privilegii este GRANT, cu sintaxa urmatoare:

```
GRANT privilegii (coloane)
  ON resurse
  TO 'user'@'statie' [IDENTIFIED BY 'password']
[REQUIRE restrictii_criptare]
[WITH optiuni suplimentare grant sau limite pe resurse consumate];
```

Daca contul nu exista va fi creat si apoi i se vor acorda privilegiile cerute.

Nota: *atunci cand clauza IDENTIFIED BY lipseste, instructiunea GRANT va crea un cont fara parola. Pentru a evita aceasta situatie se poate activa in sql_mode optiunea NO_AUTO_CREATE_USER, care nu permite lui GRANT sa creeze automat conturi decat in prezenta clauzei IDENTIFIED BY.*

Resursele carora li se aplica privilegiile se specifica sub forma bazadedate.tabela, unde ambele portiuni pot fi inlocuite cu * care semnifica “orice baza de date/tabela” (ex: produse.* semnifica toate tabelele din baza de date produse, *.* inseamna toate tabelele din toate bazele de date de pe server).

Din setul de privilegii posibile (care este foarte bogat) amintim cateva:

- CREATE – permite clientului sa creeze baze de date si tabele

- DROP – permite clientului sa stearga elemente de structura (baze de date, tabele, coloane etc)
- INSERT – clientul poate introduce noi inregistrari
- UPDATE – clientul poate modifica inregistrari existente
- DELETE – clientul poate sterge inregistrari existente
- SELECT – clientul poate extrage date din tabelele existente

Exista si doua privilegii speciale:
- ALL PRIVILEGES – folosit pentru a desemna setul complet de privilegii posibile
- USAGE – cuvant cheie folosit pentru a desemna lipsa oricaror privilegii

Exemple:

```
# acordare de privilegii de extragere si modificare de date pe toate tabelele din produse:
GRANT SELECT,UPDATE ON produse.* TO 'vlad'@'localhost' IDENTIFIED BY 'mypass';

# acordare de privilegii depline pe toata informatia pentru userul root:
GRANT ALL PRIVILEGES ON *.* to 'root'@'localhost' IDENTIFIED by 'pass';
```

5.5.7. Stabilirea de limite pentru un cont

In tabela User, fiecare cont poate avea un set de restrictii atasate, care cuprind aspecte precum:
- numarul maxim de interogari pe ora: clauza MAX_QUERIES_PER_HOUR
- numarul maxim de interogari UPDATE pe ora: clauza MAX_UPDATES_PER_HOUR
- numarul maxim de conectari pe ora: clauza MAX_CONNECTIONS_PER_HOUR
- numarul maxim de conexiuni simultane ale unui cont: clauza MAX_USER_CONNECTIONS

Toate aceste clauze au valoare default 0, care inseamna ca nu sunt impuse niciun fel de limite.

```
CREATE USER 'sql'@'localhost' IDENTIFIED BY 'mypass';
GRANT ALL ON produse.* TO 'sql'@'localhost' WITH MAX_CONNECTIONS_PER_HOUR 10
      MAX_QUERIES_PER_HOUR 200 MAX_UPDATES_PER_HOUR 50;
```

5.5.8. Afisarea privilegiilor unui cont

Afisarea setului de privilegii atasate unui anumit cont se realizeaza cu instructiunea SQL SHOW GRANTS, ca in exemplul urmator:

```
SHOW GRANTS FOR 'sql'@'localhost';
```

Pentru a afisa privilegiile contului curent logat, putem folosi:

```
# privilegii proprii
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER();
```

5.6. Replicarea datelor intre servere MySQL

5.6.1. Descrierea mecanismului

Sincronizarea datelor intre doua servere MySQL functioneaza intr-o relatie de tip master-slave:
- serverul master este locul in care sta copia primara a datelor. Modificarile asupra datelor se efectueaza numai pe acest server, deoarece sunt sincronizate numai in sensul master --> slave, nu si invers

- serverul slave se conecteaza la master, obtine lista de modificari efectuate de la ultima sincronizare si le aplica pe copia sa a datelor pentru a o aduce la zi

Nota: *un server master poate avea mai multe servere slave. De asemenea, un server slave isi poate sincroniza datele folosind ca sursa un alt slave.*

Pe serverul slave ruleaza doua thread-uri ce participa in replicarea datelor:

- **I/O thread** – este firul de executie responsabil cu procesarea evenimentelor primite de la master. Evenimentele sunt inregistrate in asa-numitul relay log
- **SQL thread** – este firul de executie care proceseaza relay log-urile, stergand fiecare fisier pe masura ce modificarile consemnate in el au fost efectuate

Cele doua thread-uri actioneaza independent, putand fi activate sau dezactivate individual, dupa nevoie.

5.6.2. Configurarea replicarii intre doua servere MySQL

Pentru a putea configura replicarea intre doua servere MySQL este mai intai necesara indeplinirea urmatoarelor conditii:

- pentru serverul master
  - binary log-ul trebuie sa fie activat. Acesta este log-ul in care se salveaza instructiunile care produc modificari asupra bazelor de date, iar pe baza sa serverele slave determina ce modificari au de efectuat pe copia lor de date
  - trebuie sa fie permisa conectarea clientilor prin TCP/IP. Replicarea nu functioneaza prin socket
  - serverul trebuie sa aiba asignat un server ID diferit de al oricarui slave (vezi mai jos parametrii de configurare)
- pentru serverul slave
  - inainte de a porni procesul de replicare, pe slave trebuie importate datele de pe master, in starea lor curenta. Acesta va constitui punctul de plecare la care se vor raporta modificarile ulterioare. Aceasta sincronizare initiala se poate realiza prin mijloacele obisnuite de export/import date (ex: mysqldump)
  - serverul trebuie sa aiba atasat un server ID diferit de al master-ului si de al celorlalte slave-uri

Etapele de configurare sunt urmatoarele:

1. **Configurarea serverului master**. Configurarea ID-ului se efectueaza folosind directiva de configurare *server-id*:

```
# SERVER MASTER
[mysqld]
server-id = 11
# nu uitam sa activam binary log
log-bin = mysql-bin
```

Nota: *adaugarea acestor directive impune restart de server (valabil si mai jos pentru serverul slave).*

2. **Configurarea serverului slave**. Pentru acest server este suficienta configurarea ID-ului; activarea lui binary log nu este obligatorie decat in conditiile in care serverul slave joaca rol de master pentru un alt server sau daca necesitati externe impun activarea acestui tip de log.

```
# SERVER SLAVE
[mysqld]
server-id = 21
```

Nota: *daca nu se doreste replicarea integrala, ci numai a unora dintre bazele de date de pe master, putem folosi pe serverul slave directivele de configurare* replicate-do-db *si* replicate-ignore-db.

3. **Crearea unui cont dedicat replicarii pe serverul master**. Acest cont va avea nevoie doar de privilegiul REPLICATION SLAVE:

```
CREATE USER 'replic8'@'adresa.server.slave' IDENTIFIED BY 'thou_shalt_not_pass';
GRANT REPLICATION SLAVE ON *.* TO 'replic8'@'adresa.server.slave';
```

4. **Identificarea pozitiei curente din logurile serverului master**, pentru a putea comunica apoi slave-ului de unde sa inceapa sa aplice schimbari. Se realizeaza pe serverul master prin intermediul instructiunii SQL *SHOW MASTER STATUS.* Pentru a ne asigura ca toate modificarile au fost salvate inaintea obtinerii informatiei precedam comanda amintita cu un FLUSH TABLES:

```
mysql> flush tables; show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000243 |       98 |              |                  |
+------------------+----------+--------------+------------------+
```

5. Se efectueaza **sincronizarea initiala a slave-ului**, prin metode manuale (mysqldump etc). *Atentie! Un dump complet include si baza de date mysql, suprascriind astfel toate conturile si privilegiile definite anterior pe slave! Includeti in dump numai bazele de date dorite!*
6. **Se configureaza parametrii serverului master pe serverul slave**. Acestia sunt stabiliti folosind instructiunea SQL CHANGE MASTER si se salveaza automat intr-un fisier numit *master.info* din data directory.

```
CHANGE MASTER TO
  MASTER_HOST = 'adresa.server.master',
  MASTER_USER = 'replic8',
  MASTER_PASSWORD = 'thou_shalt_not_pass',
  MASTER_LOG_FILE = 'mysql-bin.000243',
  MASTER_LOG_POS = 98;
```

7. **Se porneste replicarea** folosind pe serverul slave instructiunea SQL:

```
START SLAVE
```

Pentru oprirea ulterioara exista STOP SLAVE. Daca este nevoie, in cadrul acestei instructiuni se poate preciza care dintre cele doua thread-uri se doreste oprit:

```
# oprirea thread-ului care obtine lista de modificari de la master si o scrie in relay log
STOP SLAVE IO_THREAD

# oprirea thread-ului care aplica modifcarile din relay log
STOP SLAVE SQL_THREAD
```

8. **Se urmareste functionarea replicarii** folosind pe serverul slave instructiunea SQL SHOW SLAVE STATUS. Daca in acest scop se foloseste clientul mysql, instructiunea trebuie urmata de \G in loc de ; pentru ca output-ul sa fie lizibil, ca in exemplul urmator:

```
mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
             Slave_IO_State: Waiting for master to send event
                Master_Host: 192.168.1.3
                Master_User: repl
                Master_Port: 3306
              Connect_Retry: 60
            Master_Log_File: mysql-bin.000244
        Read_Master_Log_Pos: 2414
             Relay_Log_File: mysqld-relay-bin.000005
              Relay_Log_Pos: 879
      Relay_Master_Log_File: mysql-bin.000244
           Slave_IO_Running: Yes
          Slave_SQL_Running: Yes
            Replicate_Do_DB:
        Replicate_Ignore_DB:
         Replicate_Do_Table:
     Replicate_Ignore_Table:
    Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
                 Last_Errno: 0
                 Last_Error:
               Skip_Counter: 0
        Exec_Master_Log_Pos: 2414
            Relay_Log_Space: 879
            Until_Condition: None
             Until_Log_File:
              Until_Log_Pos: 0
         Master_SSL_Allowed: No
         Master_SSL_CA_File:
         Master_SSL_CA_Path:
            Master_SSL_Cert:
          Master_SSL_Cipher:
             Master_SSL_Key:
      Seconds_Behind_Master: 0
```

In output-ul de mai sus au fost evidentiate liniile de interes sporit: cele care indica corecta conectare a slave-ului la master, cele care arata starea celor doua thread-uri ale slave-ului, pozitia curenta in binlog-ul master-ului si erorile aparute la executarea relay log-ului.

Atentie! *In caz de aparitie a unei erori la executarea unei instructiuni din relay log (ex: incercarea de stergere a unei baze de date care a fost deja stearsa de pe slave, etc.) thread-ul SQL se opreste si asteapta administratorul sa ia masuri. Eroarea aparuta este afisata pe linia Last_Error din output-ul de mai sus. Daca analizam eroarea si decidem ca evenimentul care o cauzeaza poate fi sarit, putem proceda astfel:*

```
mysql> STOP SLAVE;
mysql> SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;
mysql> START SLAVE;
```

5.7. BIBLIOGRAFIE

- MySQL Reference Manual: http://dev.mysql.com/doc/refman/5.1/en/
- Lista variabilelor de stare MySQL: http://dev.mysql.com/doc/refman/5.1/en/dynindex-statvar.html
- Lista variabilelor de sistem MySQL: http://dev.mysql.com/doc/refman/5.1/en/dynindex-sysvar.html
- Optimizare MySQL: http://www.databasejournal.com/features/mysql/article.php/3367871/Optimizing-the-mysqld-variables.htm
- Optimizare InnoDB: http://www.mysqlperformanceblog.com/2007/11/01/innodb-performance-optimization-basics/
- InnoDB Buffer Pool: http://dev.mysql.com/doc/refman/5.1/en/innodb-buffer-pool.html

- MySQL Instance Manager: http://www.lenzg.net/archives/17-Enabling-and-using-the-MySQL-Instance-Manager-IM.html

5.8. ANEXA 1. Instructiuni SQL uzuale si tipuri de date MySQL

Manipulare structura

Scop	Forma generala	Exemplu
Creare baza de date	CREATE DATABASE numebaza	CREATE DATABASE magazin
Stergere baza de date	DROP DATABASE numebaza	DROP DATABASE magazin
Vizualizare baze de date	SHOW DATABASES	SHOW DATABASES
Stabilire b.d. default	USE numebaza	USE magazin
Creare definitie tabela	CREATE TABLE numetabela(definitie coloana 1, definitie coloana 2, ...)	CREATE TABLE produse(denumire VARCHAR(100), pret FLOAT(5,2), cantitate INT)
Stergere tabela	DROP TABLE numetabela	DROP TABLE produse
Vizualizare tabele	SHOW TABLES	SHOW TABLES
Vizualizare definitie tabela	DESCRIBE numetabela	DESCRIBE produse
Adaugare coloana	ALTER TABLE numetabela ADD COLUMN definitie_coloana	ALTER TABLE produse ADD COLUMN furnizor VARCHAR(200)
Stergere coloana	ALTER TABLE numetabela DROP COLUMN numecoloana	ALTER TABLE produse DROP COLUMN furnizor

Manipulare date

Scop	Forma generala	Exemplu
Introducere inregistrare	INSERT INTO tabela(col1,col2,...) VALUES(val1,val2,...)	INSERT INTO produse(denumire,pret,cantitate) VALUES('mere',5,10)
Extragere date	SELECT expr1,expr2,... FROM tabela WHERE conditie1 AND conditie2 OR....	SELECT denumire, pret*1.19 FROM produse WHERE pret<100 AND cantitate=3
Stergere inregistrari	DELETE FROM tabela WHERE conditii	DELETE FROM produse WHERE pret>1000
Modificare inregistrari	UPDATE tabela SET col1=val1, col2=val2... WHERE conditii	UPDATE produse SET pret=pret*0.9 WHERE cantitate<100 AND denumire='rosii'

Tipuri de date MySQL

Categorie			Tipuri de date MySQL	Exemplu de definitie de coloana
numerice	intregi		TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT	CantitateProdus MEDIUMINT
	fractionare	virgula fixa	DECIMAL	PretProdus DECIMAL (6,2)
		virgula mobila	FLOAT, DOUBLE	Greutate FLOAT(4,1)
siruri	de octeti		BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB	Semnatura VARBINARY(250) Thumbnail BLOB
	de caractere		CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT	NumeProdus VARCHAR(200) NrTel CHAR(10) Comentariu TEXT
temporale			DATE, DATETIME, TIME, TIMESTAMP, YEAR	DataNasterii DATE DataOraIntrare DATETIME
enumerate			ENUM, SET	Opinie ENUM('pro','contra')