Universitatea “Politehnica” din București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

**Studiul bazelor de date distribuite in sistemul de gestiune al bazelor de date MariaDB**

**Proiect de diplomă**

prezentat ca cerință parțială pentru obținerea titlului de

Inginer în domeniul Calculatoare și Tehnologia

Informației

programul de studii de licență Ingineria

Informației (CTI – INF)

Conducător științific Absolvent

S.l. Ing. Valentin PUPEZESCU Bogdan - Petru MATRAGOCIU

Anul

2014

Cuprins

1. Introducere
2. Baze de date

Introducere

* 1. Scurt istoric
  2. Noțiuni generale
  3. Tipuri de date

1. MariaDB
   1. Introducere
   2. Scurt istoric
   3. Concepte
   4. MariaDB vs. MySQL
   5. MariaDB vs. NoSQL
   6. Replicarea
   7. Avantaje si dezavantaje
2. Java
   1. Introducere
   2. Scurt Istoric
   3. Sintaxa Java
   4. OOP
   5. Integrarea cu baza de date
3. Tehnologii user-interface
   1. HTML
      1. Introducere
      2. Structura unei pagini HTML
      3. Tag-uri, elemente si atribute
   2. CSS
      1. Introducere
      2. Selectori in CSS
      3. Proprietăți CSS
   3. JavaScript
   4. jQuery
4. Proiectarea aplicației
   1. Prezentare generala a interfeței
      1. Secțiunea X
      2. Secțiunea Y
   2. Structura bazei de date
   3. Testarea replicării
5. Concluzii
6. Bibliografie
7. Anexe

**Replicarea**

Replicarea in MariaDB se realizează la fel a in MySQL. Replicarea este fundația pentru a construi aplicații mari de performanta înalta folosind MariaDB. Replicarea permite configurarea unui server sau a mai multor servere, replici a altui server. Acest lucru nu este valabil doar pentru aplicațiile de înalta performanta dar si pentru rezolvarea altor problem cu mar fi, sharing-ul de date cu mașini aflate la distanta, păstrarea unei copii de rezerva sau păstrarea unei copii pentru testare sau învățare (training).

In continuare for fi analizate aspectele toate aspectele replicării. Pentru început va fi prezentat modul de funcționare, apoi configurarea unui server, proiectarea unor configurații mai avansate de replicare, management-ul si optimizarea serverelor replicate.

**Problema replicării**

Problema replicării se rezolva prin punerea la dispoziție a unui server care sa fie sincronizat cu un altul. Mai multe salve-uri se pot conecta la un singur master, iar un salve se poate comporta la rândul sau ca un master. Master-ii si salve-urile pot fi aranjați in diferite topologii. Se poate replica întregul server, doar anumite baze de date sau se pot alege doar anumite tabele pentru replicare.

MariaDB suporta 2 tipuri de replicare : replicare „statement-based” sau replicare „row-based. „Statement-based” sau replicare logica este disponibila de la MySQL 3.23 si este cea mai folosita in acest moment. Replicare „row-based” este valabila de la MySQL 5.1. Ambele tipuri de replicare salvează schimbările efectuate pe baza de date in așa numitul „binary log” iar acest log este rulat pe slave. Ambele tipuri de replicare sunt asincrone, asta însemnând ca datele copiate pe slave nu au certitudinea ca sunt actualizate la un anumit moment de timp. Nu exista nicio garanție asupra timpului de actualizare a datelor pe slave. Slave-ul poate poate garanta actualizarea datelor după secunde, minute sau chiar ore daca query-urile au dimensiuni mari.

Replicarea MySQL este ca mai compatibila cu versiunile anterioare. Asta înseamnă ca un server cu o versiune mai noua poate fi slave pentru un server cu o versiune mai veche fără probleme. Totuși versiunile mai vechi in principiu nu pot fi slave pentru un server cu o versiune mai noua (nu pot înțelege funcționalități noi apărute si pot fi diferențe intre formatul de fișier folosit pentru replicare). Replicare nu presupune un overhead mare pe master, presupune doar activarea funcției de „binary logging”. Fiecare slave poate adăuga o încărcare suplimentara dar infima datorita operațiilor I/O de rețea.

Replicarea este in general folositoare pentru a scala citiri care pot fi direcționate către un slave, dar nu este o modalitate buna de a scala scrieri, doar in cazul in care este făcută așa cum trebuie. Atașarea multor slave-uri la un master presupune scrierea datelor pe fiecare din mașini. Fiecare sistem este limitata la viteza de scriere pe care o are cea mai slaba mașina.

Replicare este ineficienta când există mai mult de câteva slave-uri deoarece se duplica date fără rost. De exemplu un singur master cu 10 slave-uri va stoca 11 copii a acelorași date si va duplica mare parte din date in 11 cache-uri diferite. Acesta este analog cu „11 way RAID to 1” la nivel de server care specifica faptul ca acest timp de arhitectura nu este un mod de a utiliza eficient aparatura hardware. Totuși acest tip de arhitectura poate fi întâlnit destul de des. In continuare vor fi discutate moduri de a evita aceasta problema.

**Probleme rezolvate de replicare**

Cele mai întâlnite utilizări ale replicării:

* Distribuția datelor (Data distribution)

Replicare in MariaDB si MySQl nu folosește de obicei foarte mult din lungimea de banda a rețelei si poate fi oprita sau pornita la cerere. Prin urmare este recomandată păstrarea unei copii a datelor într-un loc distant din punct de vedere geografic cum ar fi alt centru de date. Slave-ul de la distanta poate lucra chiar cu o conexiune intermitenta (intenționată sau nu). Totuși daca este necesara o replicare fără timp de răspuns mare va fi nevoie de o conexiune stabila.

* Distribuirea încărcării (Load balancing)

Replicare poate ajuta la distribuția query-urilor de citire pe ,ai multe servere, lucru ce funcționează foarte bine pe aplicații cu operații intensive de citire. Se poate efectua „load balancing” doar cu câteva linii de cod. Pe scala mica se poate utiliza abordarea simplista cum ar fi nume de host „hardcoded” sau DNS round-robin (care face ca mai multe IP uri sa pointeze către aceeași gazdă). Exista si abordări mai sofisticate. Soluții standard mai costisitoare sunt produsele pentru distribuție a încărcării pe rețea care funcționează bine împreună cu serverele MySQL. Proiectul server virtual Linux (LVS) se pliază pe aceasta situație.

* Rezerva (Backup)

Replicarea este o tehnica buna pentru a păstra o copie de rezerva a datelor. Însă un slave nu este prin definiție un backup.

* Disponibilitate mare si failover

Replicarea ajuta la evitarea problemei ce presupune prăbușirea aplicației daca exista un singur punct care daca pica poate face aplicația inutilizabila (defectare hardware a mașinii ce tine serverul de baze de date). Un sistem bun de failover este deținerea de slave-uri replicate ce pot lua locul master-ului in caz de defectare.

* Testarea actualizărilor (updates) aduse sistemului de gestiune de baze de date

Este o practica des întâlnită setarea unui slave care are o versiune actualizata de MariaDB/MySQL si folosirea ei pentru a testa buna funcționalitate înainte de a actualiza toate sistemele.

**Cum funcționează replicarea**

Modul in care sunt replicate datele este prezentat in continuare. Văzută de la un nivel înalt replicarea presupune 3 pași:

1. Masterul înregistrează schimbările in fișierul de „binary log”. (Aceste modificări se numesc evenimente ale „binary log”)
2. Slave-ul copiază fișierul „binary log” de la master in log-ul sau „relay log”.
3. Slave-ul reia evenimentele din „relay log”, aplicând modificările pe datele sale.

Aceasta este vederea de ansamblu. Fiecare pas este destul de complex. In continuare va fi prezentata replicarea mai in detaliu.

Prima parte a procesului este cea de „binary logging” care are log pe master. Înainte ca orice tranzacție sa înceapă sa modifice date pe master, master-ul înregistrează schimbările in „binary log”. Tranzacțiile sunt scrise serial chiar daca ele au fost intercalate in momentul execuției lor. După această scriere in log, masterul indică motorului de stocare (storage engine) să salveze (to commit) tranzacțiile.

Cum functioneaza replicarea:



Urmatorul pas este ca slave-ul sa copieze „binary log”-ul de pe master pe hard-disk-ul sau inasa numitul „relay log”. Pentru inceput porneste un fir de executie „worker” numit fitul de executie I/O al slave-ului. Firul de execute I/O deschide o conexiune obisnuita de la client catre master, apoi incepe o operatirtie speciala numita „binlog dump” (nu exista o comanda SQL corespondenta). Acest proces citeste evenimentele din „binary log”-ul masterului. El nu interogheaza master-ul in vederea schimbarilor ci dupa ce este adus la curent cu masterul intra in „sleep” si asteapta ca masterul sa semnaleze noi evenimente. Acest fir de executie scrie evenimentele in „relay log”.

Fitul de executie slave SQL se ocupa de ultima parte a integului proces. Acest fir citeste si reruleaza evenimentele din „relay log” astfel incat datele de pe slave dor fi la curent cu cele de pe master. Atat timp cat acest fir de executie tine paasul cu firul I/O, „relay log”-ul sta de obicei in cache-ul sistemului de operare, prin urmare „relay log”-urile au un overhead foarte mic. Evenimentele pe care firul de executie SQL le executa pot ajunge optional in „binary log”-ul slave-ului, lucru folositor pentru scenariile de replicare ce vor fi prezentate ulterior.

In figura sunt prezente 2 fire de executie pe slave dar conexiunea pe care slave-ul o deschide pe master porneste un nou fir de executie si pe master.

Acest proces de replicare decupleaza procesele de detectare si rerulare a evenimentelor pe slave ceea ce ofera posibilitatea ca ele sa fie asincrone. Prin urmare, firul de executei I/O poate lucra independent de firul de executie SQL. De asemenea sunt palciate constrangeri pe procesul de replicare, cea mai importanta fiind ca replicarea este serializata pe slave. Asta inseamna ca actualizari ce ar fi putut rula in paralel pe master nu for fi paralelizate pe slave. Dat fiind acestui fapt pot exista bottle-neck-uri cand exista multe date de prelucrat.

**Setarea replicarii**

Setarea replicareii este un proces relativ simplu, dar exista multe variatiuni ale procesului i nfunctie de scenariu. Cel mai simplu scenariu este atunci cand avem un master si un slave. Procesul presupune urmatoare etape in mare:

1. Configurarea contrurilor de replicare pe fiecare server;
2. Configurarea masterului si a slave-ului;
3. Insruirea serverului sa se conecteze si sa replice din master.

In principiu multe setari standard (default) nu vor trebui modificate, atat timp cat cele doua instante sunt proaspat instalate si contin aceleasi date. In continuare vor presupune ca cele doua instante se numesc server1 (cu adresa IP 192.168.0.1) si server2 (cu adresa IP 192.168.0.2). Apoi va fi explicata intitializarea slave-ului .

Crearea conturilor pentru replicare

Exista cateva privilegii ce trebuie setate penru ca replicarea sa functioneze. Firul de executie I/O de pe slave se conecteaza la master prin TCP/IP. Asta inseman ca va trebui creat un cont pe master si setarea privilegiilor corespunzartoare pentru ca firul I/O sa se conectez3e ca si acel usersi sa citeazsca „binary log”-ul masterului.cum se crea contrul pontru userul *repl:*

**mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\***

**-> TO repl@'192.168.0.%' IDENTIFIED BY 'p4ssword';**

Vom crea acest cont user si pe master si pe slave. Replicarea este constransa la reteaua locala curenta deoarece contul de replicare nu este sigur.

Replicarea presupune doar setarea contului doar pe master dar s-a facut setarea lui si ple client deoarece este mai usoara monitorizarea ulterioara a schimbarilor de pe client si eventuala comportarea a slave-ului ca master pentru un nou slave.

**Configurarea masterului si slave-ului**

Urmatorul pas consta in activarea unor optiuni pe master, pe care il numim server1. Trebuie activat „binary logging” si specificat un ID pentru server. In fisierul my.cnf trebuie adaugate sau modificate urmatoarele linii :

**log\_bin = mysql-bin**

**server\_id = 10**

Valorile sunt la alegerea utilizatorului. Ele au fost alese pentru simplitate.

ID-ul de server trebuie sa fie unic. De aceea a fost ales 10 in loc de 1. 1 este faloarea standard (default). Prin urmarea utilizarea numarului 1 poate crea confuzii sau conflicte intre servere careau acelasi ID sau nu il au setat. Un obicei comun este utilizarea ultimului octet din adresa IP a serverului presupunand ca nu se va schimba si ca este unica.

Daca optiunea „binary logging” nu era specificata in configurare este necesara restartarea servereului. Pentru a verifica faptul ca fisierul a fost creat pe master se poate rula comanda **SHOW MASTER STATUS** si se va obtine un output similar cu urmatorul:

**mysql> SHOW MASTER STATUS;**

**+------------------+----------+--------------+------------------+**

**| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |**

**+------------------+----------+--------------+------------------+**

**| mysql-bin.000001 | 98 | | |**

**+------------------+----------+--------------+------------------+**

**1 row in set (0.00 sec)**

Slave-ul necesita si el o configurare similara cu masterul in fisierul my.cnf si o restartare.

**log\_bin = mysql-bin**

**server\_id = 2**

**relay\_log = mysql-relay-bin**

**log\_slave\_updates = 1**

**read\_only = 1**

Cateva din aceste optiuni nu sunt strict necesare, sunt doar enuntate cele standard. Pe salve doar server ID-ul este necesar dar a fost activat si „binary logging” cu un nume explicit. In principu vrem aceeasi configurare pe toate masinile astfel incat promovarea de al slave la master sa fie facila. La fel cum au fost creeate acealeasi conturi si pe master si pe slave, vrem si aceleasi setari.

Exista 2 parametri optionali: relay\_log – care specifica locatia si numele „relay log”-ului si log\_slave\_updates – care specifica slave-ului sa adauge evenimentele primite si in „binary log”ul propriu. Ultima optiune cauzeaza lucru in plus pentru slave dar in continuare va fi explicata de ce adaugarea ei este un practice bun.

**Pornitrea slave-ului**Urmatorul pas spune slave-ului cum sa se conecteze la master si sa incapea rerularea „binary logg”-urilor primite. Pentru acest scop nu trebuie folost fisierul my.cnf ci comanda CHANGE MASTER TO. Aceasta comanda inlocuieste setarile corespondente din my.cnf in totalitate. I n viitor permite si schimbarea masterului fara oprirea serverului. Uramtorul ocd trebuie rulat pe slave:

**mysql> CHANGE MASTER TO MASTER\_HOST='server1',**

**-> MASTER\_USER='repl',**

**-> MASTER\_PASSWORD='p4ssword',**

**-> MASTER\_LOG\_FILE='mysql-bin.000001',**

**-> MASTER\_LOG\_POS=0;**

Campul MASTER\_LOG\_POS este setat cu valoarea 0 deoarece indica inceputul log-ului. Dupa ce se ruleaza aceasta comanda se poate verifica setarea cu comanda SHOW SLAVE STATUS ce produce uramtorul output:

**mysql> SHOW SLAVE STATUS\G**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Slave\_IO\_State:**

**Master\_Host: server1**

**Master\_User: repl**

**Master\_Port: 3306**

**Connect\_Retry: 60**

**Master\_Log\_File: mysql-bin.000001**

**Read\_Master\_Log\_Pos: 4**

**Relay\_Log\_File: mysql-relay-bin.000001**

**Relay\_Log\_Pos: 4**

**Relay\_Master\_Log\_File: mysql-bin.000001**

**Slave\_IO\_Running: No**

**Slave\_SQL\_Running: No**

**...omitted...**

**Seconds\_Behind\_Master: NULL**

Coloanele Slave\_IO\_State, Slave\_IO\_Running, si Slave\_SQL\_Running arata ca procesele de pe slave nu ruleaza. Se observa ca pozitia log-ului este 4. Prin uramare 0 setata anterior nu este o pozitie ci indica doar inceputul fisierului de log. Primul eveniment este defapt la pozitia 4.

Pentru a rula aplicatia se executa comanda :

**mysql> START SLAVE;**

Aceasta comnada nu ar trebui sa produca erori sau output. Se poate verifica cu comanda SHOW SLAVE STATUS:

**mysql> SHOW SLAVE STATUS\G**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Slave\_IO\_State: Waiting for master to send event**

**Master\_Host: server1**

**Master\_User: repl**

**Master\_Port: 3306**

**Connect\_Retry: 60**

**Master\_Log\_File: mysql-bin.000001**

**Read\_Master\_Log\_Pos: 164**

**Relay\_Log\_File: mysql-relay-bin.000001**

**Relay\_Log\_Pos: 164**

**Relay\_Master\_Log\_File: mysql-bin.000001**

**Slave\_IO\_Running: Yes**

**Slave\_SQL\_Running: Yes**

**...omitted...**

**Seconds\_Behind\_Master: 0**

Se observa ca ambele fire de executie I?O si SQL ruleaza si campul Seconds\_Behind\_

Master nu mai este NULL. Firul de executie I/O asteapta evenimente de la master ceea ce inseamna ca a preluat toate fisierele „bianry log”de la master. Pozitiile de log s-au incrementat ceea ce inseamna ca exista evemnimente ce au fost preluate si executate. Modificare masterului va face schimbari pe slave.

Se pot vizualiza si firele de executie pentru replicare in lsita de procese atat pe amaster cat si pe slave. Pe master va fi o conexiune creata de threadul I/O al slave-ului:

**mysql> SHOW PROCESSLIST\G**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Id: 55**

**User: repl**

**Host: slave1.webcluster\_1:54813**

**db: NULL**

**Command: Binlog Dump**

**Time: 610237**

**State: Has sent all binlog to slave; waiting for binlog to be updated**

**Info: NULL**

Pe slave se pot observa 2 fire de executie, cel I/O si cel SQL:

**mysql> SHOW PROCESSLIST\G**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Id: 1**

**User: system user**

**Host:**

**db: NULL**

**Command: Connect**

**Time: 611116**

**State: Waiting for master to send event**

**Info: NULL**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Id: 2**

**User: system user**

**Host:**

**db: NULL**

**Command: Connect**

**Time: 33**

**State: Has read all relay log; waiting for the slave I/O thread to update it**

**Info: NULL**

**Initializarea unui alt slave de pe alt server**