1. REPLIKACJA

Replikacja polega na powielaniu informacji pomiędzy różnymi serwerami baz danych. Replikacja pozwala nam na:

- **Skalowalność** – rozkłada obciążenie pomiędzy wiele serwerów. Operacje zapisu i aktualizacji rekordów odbywają się na jednym serwerze, a pobieranie i przeszukiwanie danych na drugim.
- **Bezpieczeństwo** – replikacja pozwala na sklonowanie istniejącej bazy produkcyjnej. Nie uchroni to bazy przed wykonaniem operacji typu „DROP TABLE”, ale może pomóc w przypadku awarii sprzętowej głównego serwera. Zreplikowana baza nadaje się do wykonywania tradycyjnego backupu, bez konieczności wstrzymywania pracy bazy głównej.
- **Analiza** - różnego rodzaju przeliczenia i analizy statystyczne mogą być wykonywane na osobnym serwerze bez konieczności obciążania głównej bazy.
- **Separacja** - możemy udostępnić sklonowaną bazę produkcyjną dla developerów lub testerów, będą oni wtedy wykonywali operacje na kopii bazy produkcyjnej.

1.1 Replikacja w MySQL

Replikacja w bazach typu MySQL zrealizowana jest na następującej zasadzie: serwer główny nazywany serwerem master, zapisuje on wszystkie wykonywane na nim czynności do pliku nazywanego dziennikiem. Wykorzystywane są do tego celu bin-logi (pliki binarne zawierające instrukcje jakie wykonał serwer master). Rola serwera zapasowego (slave) polega na odczytywaniu tych danych i kolejnym wykonywaniu zapytań w nich zawartych. W ten sposób zapełnia on bazę rekordami. Efektem takiej pracy są dwie identyczne bazy danych.

1.2 Rodzaje replikacji

W MySQL-u dostępne są trzy metody replikacji, przekłada się to na format danych zapisywanych do bin-logów. W celu określenia rodzaju replikacji należy ustawić odpowiednią wartość zmiennej „binlog_format”, która może przyjąć wartość: row, statement lub mixed. Metody replikacji:

- **SBR** (statement-base replication) - serwer zapisuje do pliku tylko zapytania jakie wykonał.
- **RBR** (rew-based replication) - do bin-logów zapisywane są wyniki działań zapytań na serwerze master. Zapisywana jest informacja jaki rekord został w jaki sposób zmieniony.
- **MFL** (mixed format logging) - jest to połączenie dwóch powyższych typów replikacji.

Najszybszą metodą replikacji jest wykorzystanie techniki **SBR**. Serwer główny zapisuje do pliku zapytanie jakie wykonał, następnie serwer zapasowy je odczytuje i wykonuje. Przykładem takiego zapytania może być:

```
DELETE FROM product WHERE product_id = 49598;
```

Metoda ta jest bardzo szybka i wydajna. Do pliku logów zapisywane są tylko zapytania SQL.

W metodzie **RBR** do bin-logów zapisywane są zmiany jakie zaszły po wykonaniu polecenia. Logowane są informacje na temat sposobu modyfikacji konkretnych rekordów. Metoda ta jest znacznie wolniejsza niż **SBR,** powoduje również znaczne zwiększenie ilości wysyłanych danych pomiędzy replikującymi się serwerami. W metodzie **MFL**, w większości przypadków, logowane są zapytania SQL tak jak w przypadku **SBR**, natomiast dla zapytań, których wynik nie jest przewidywalny,

włączana jest replikacja **RBR**.

1.3 Konfiguracja replikacji

Na samym początku zajmiemy się konfiguracją serwera master. W tym celu musimy poddać edycji plik konfiguracyjny serwera MySQL. Najczęściej jest to plik /etc/my.cnf. Powinny znaleźć się w nim takie opcje jak:

```
[mysqld]
log-bin = /tmp/mysql-bin
binlog_format = mixed
max_binlog_size = 50M
server-id = 1
```

Każda z tych opcji odpowiada za konkretny parametr dotyczący replikacji:

- **log-bin = /tmp/mysql-bin** – uruchamia mechanizm logowania zmian w bazie.
- **binlog_format = mixed** - ustawia metodę replikacji.
- **max_binlog_size = 50M** - maksymalny rozmiar bin-logów.
- **server-id** – unikatowy numer serwera w obrębie replikacji.

Opcjonalną opcja pozwalającą na replikację tylko wybranej bazy jest opcja binlog-do-db=databse.

Po restarcie serwera MySQL, logujemy się do niego (mysql -u root -p) i wpisujemy następujące zapytanie SQL. Zwróci ono stan serwera master.

```
mysql> SHOW MASTER STATUS\G
*************************** 1. row ***************************
        File: mysql-bin.000003
        Position: 106
        Binlog_Do_DB:
        Binlog_Ignore_DB:
        1 row in set (0.00 sec)

mysql>
```

Kolejnym krokiem jest stworzenie użytkownika odpowiedzialnego za autoryzację serwerów slave. W tym celu wpisujemy zapytanie SQL:

```
CREATE USER 'repuser'@'%' IDENTIFIED BY 'haslo';
GRANT REPLICATION SLAVE ON *.* TO 'repuser'@'%' IDENTIFIED BY 'haslo';
```

Kolejnym krokiem jest konfiguracja serwera slave. W jego pliku konfiguracyjnym (najczęściej /etc/my.cnf) należy umieścić poniższe opcje:

```
server-id=2
master-host=192.168.0.127
master-user=repuser
master-password=haslo
master-connect-retry=30
```

Dla serwerów MySQL 5.5 i nowszych:

```
CHANGE MASTER TO MASTER_HOST='192.168.0.127',
MASTER_USER='repuser',
MASTER_PASSWORD='haslo';
```

Sprawdzenie poprawności konfiguracji:

```
mysql> LOAD DATA FROM MASTER;
Query OK, 0 rows affected, 1 warning (0.34 sec)
mysql> SLAVE STOP;
Query OK, 0 rows affected (0.00 sec)
```

Kolejnym krokiem jest ustawienie parametrów replikacji. Wydajemy zapytanie SQL:

```
CHANGE MASTER TO
MASTER_HOST='192.168.0.127',
MASTER_LOG_FILE='mysql-bin.000003',
MASTER_LOG_POS=1274;
```

Wartości MASTER_LOG_FILE oraz MASTER_LOG_POS pobieramy Za pomocą polecenia SHOW MASTER STATUS wydanego po stronie serwera głównego. Teraz pozostaje już tylko włączyć replikację oraz sprawdzić jej status.

```
mysql> START SLAVE;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW SLAVE STATUS\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.0.127
                  Master_User: repuser
                  Master_Port: 3306
                Connect_Retry: 30
              Master_Log_File: mysql-bin.000003
          Read_Master_Log_Pos: 1274
               Relay_Log_File: mysqld-relay-bin.000002
                Relay_Log_Pos: 251
        Relay_Master_Log_File: mysql-bin.000003
             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: 1274
              Relay_Log_Space: 407
              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
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
1 row in set (0.00 sec)

ERROR:
No query specified

mysql>
```

1.4 Replikowanie wybranej tabeli

Mechanizm replikacji MySQLa udostępnia możliwość replikowanie wybranej tabeli, bazy lub wyłączenia jej z replikacji. Odpowiadają za to parametry:

replicate-do-db
replicate-ignore-db
replicate-do-table
replicate-ignore-table
replicate-wild-do-table
replicate-wild-ignore-table
binlog-do-db
binlog-ignore-db

Umożliwiają one filtrowanie tego co jest replikowane. Możemy replikować lub nie wybrane bazy lub tabele. Nie jest zalecanie korzystanie z tego typu funkcjonalności, jeżeli nie jesteśmy pewni jakie zapytania idą do bazy danych. Może się okazać, że ładnie skonfigurowana replikacja nagle się rozsynchronizuje.

1.5 Problemy związane z replikacją

Replikację możemy wykorzystać do wykonywania kopii zapasowych. Dzięki temu backup jest pobierany z jednego z serwerów zapasowych, nie obciążając pracującego serwera głównego.

Niestety replikacja nie uchroni nas przed zapytaniami, które kasują rekordy, tabele czy bazy danych. Aby uniknąć takich sytuacji, wykorzystuje się polecenia CHANGE MASTER TO MASTER _DELAY = N;. Opóźni ono replikowanie danych o **N** sekund.

Replikacja chroni nas przed fizycznym uszkodzeniem serwera głównego. W takim wypadku możemy przełączyć naszą aplikację na bazę zapasową.

Jedną z ważnych cech replikacji MySQL jest jej **asynchroniczność**. Nie daje to jednak żadnej gwarancji, że po wykonaniu dowolnej operacji na głównej bazie danych, zostały one przesłane i zapisane na zapasowych bazach. W wersji MySQL 5.5 został dodany półsynchroniczny mechanizm replikacji, w którym dane wysyłane są do serwera zapasowego po wykonaniu polecenia COMMIT i

przechowywane są w relay-logach. W momencie kiedy serwer zapisze wszystkie otrzymane dane, wysyła potwierdzenie do serwera master. Jeżeli serwer główny otrzyma potwierdzenie informacji od co najmniej jednego serwera slave, transakcja jest uznawana za zakończoną. Rozwiązanie takie oczywiście wypływa na wydajność głównej bazy danych, która musi czekać na odpowiedzi z serwera zapasowego.

2. Duplikacja MySQL

Duplikacja master-master to właściwie dwie replikacje typu master-slave. Pozwala ona na przeprowadzanie transakcji jednoczesie na dwóch serwerach. Propagacja danych następuje dużo szybciej niż w przypadku replikacji master-slave.

Konfiguracja duplikacji:

Pierwsze kroki zrobienia replikacji master-slave zostały opisane już wcześniej. Poniżej znajduje się lista kroków z przykładowymi plikami konfiguracyjnymi, które posłużą za wzór do wykonania drugiej replikacji typu master-slave, czyli duplikacji.

```
mysql> grant replication slave on *.* to 'replication'@192.168.16.4
identified by 'slave';

mysql> start master;

#master-1/slave-1
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
old_passwords=1
log-bin
binlog-do-db=<database name>  # input the database which should be
replicated
binlog-ignore-db=mysql            # input the database that should be ignored
for replication
binlog-ignore-db=test
server-id=2
[mysql.server]
user=mysql
basedir=/var/lib
[mysqld_safe]
err-log=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

#master-2/slave-2

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
old_passwords=1
log-bin
binlog-do-db=<database name>  # input the database which should be
replicated
binlog-ignore-db=mysql            # input the database that should be ignored
for replication
binlog-ignore-db=test
server-id=1
```

```
[mysql.server]
user=mysql
basedir=/var/lib
[mysqld_safe]
err-log=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

mysql> show master status;
+------------------------+----------+--------------+------------------+
| File                   | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------------+----------+--------------+------------------+
|mysqld-bin.000012       |      106 | adam         |                  |
+------------------------+----------+--------------+------------------+
1 row in set (0.00 sec)

mysql> CHANGE MASTER TO MASTER_HOST='192.168.16.5',
MASTER_USER='replication', MASTER_PASSWORD='password',
MASTER_PORT=3306,MASTER_LOG_FILE='mysqld-bin.000012', MASTER_LOG_POS=106,
MASTER_CONNECT_RETRY=10;
```

#master1

```
auto_increment_increment= 2
auto_increment_offset   = 1

#master2
auto_increment_increment= 2
auto_increment_offset   = 2
```

3. Bonus – kopia zapasowa

3.1 Używanie

Duplikacja jest kopią bazy z jednoczesną zmianą unikalnego identyfikatora bazy. Możena odtworzyć bazę danych na podstawie pliku backupu oraz archivelogów. Możliwa jest również duplikacja do punktu w czasie, dzięki czemu istnieje możliwość porównania aktualnego stanu obiektów (z bazy produkcyjnej) z ich stanem nawet w odległej przeszłości. Taka duplikacja do punktu w czasie działa podobnie jak odtwarzanie do punktu w czasie, z tym że na innej bazie docelowej. Duplikację można zastosować jako prostą metodę kopiowania bazy w celach testowych. Na takiej kopii możesz np. sprawdzić upgrady, patche, modyfikacje przed wdrożeniem zmian na bazie produkcyjnej.

Bazę możesz zduplikować na tym samym hoście, lub po skopiowaniu niezbędnych plików na innym.

MySQL posiada wbudowane narzędzie nazywające się **mysqldump** które służy do tworzenia niezależnych od platformy plików tekstowych które zawierają pełną kopię tego co znajdowało się w bazie, która przez przypadek została utracona.

3.2 Jak wykonać

Najłatwiejszym sposobem aby stworzyć duplikacje bazy MySQL jest użycei komendy **mysqldump**:

```
mysqldump -u <uzytkownik> -p <haslo> -h <nazwahosta> db_przykladowa >
/usr/backups/mysql/db_przykladowa
```

MySQL użyje **username**, **password**, oraz **host** aby sprawdzić czy posiadamy uprawnienia dostępu

do bazy. Po udanej autentykacji, przekierowujemy cały potok wyjścia wytworzony przez polecenie mysqldump do wskazanego przez nas pliku. W niektórych przypadkach gdy kopie robione są kilka razy na dzień przydatne jest również wstawianie informacji z czasem ich utworzenia.

Przydatne parametry

Parametr	Opis
–databases	Kilka baz danych. W pliku sql dodawne są klauzule 'USE'. Użycie: –database db1 db2 d3
–all-databases	Zrzut wszystkich baz danych do jednego pliku
–default-character-set	Domyślne kodowanie znaków. Użycie: –default-character-set=utf8
–add-drop-database	Dodaje klauzlę DROP DATABASE (IF EXIST). Przy przyrwacaniu kopii zapasowej baza danych jest kasowana, a następnie ładowany wykonany zrzut bazy
–add-drop-table	Podobnie jak wyżej, dodaje DROP TABLE. Domyślnie ta wartość jest włączona, nie trzeba podawać tego parametru
–no-data	Zrzut bez danych z bazy, sama struktura
–no-create-info	Odwrotnie niż wyżej. Same dane, bez struktury bazy
–ignore-table	Ignoruje tabelę o podanej nazwie. Użycie: –ignore-table=table1
–xml	Zrzut bazy w formacie -xml

Przykłady użycia

Zrzut lokalnej bazy db1 do pliku dump.sql:

```
mysqldump -uuser -ppassword db1 > dump.sql
```

Zrzut tabel t1 t2 t3 z bazy db1:

```
mysqldump -uuser -ppassword db1 t1 t2 t3 > dump.sql
```

Zrzut bazy db1 z pominięciem tabeli t1:

```
mysqldump -uuser -ppassword db1 --ignore-table t1 > dump.sql
```

Zrzut samej struktury bazy db1:

```
mysqldump -uuser -ppassword db1 --no-data > dump.sql
```

Zrzut baz db1, db2, db3:

```
mysqldump -uuser -ppassword --databases db1 db2 db3 > dump.sql
```

Zrzut wszystkich baz:

```
mysqldump -uuser -ppassword --all-databases > dump.sql
```

Zrzut bazy db1 do pliku xml:

```
mysqldump -uuser -ppassword --xml db1 > dump.xml
```

Zrzut bazy db1 do skompresowanego pliku gz (zalecane w przypadku dużych baz):

```
mysqldump -uuser -ppassword db1 | gzip > dump.gz
```

przykład:

MySQL haslo: q1w2e3!@#
nazwa bazy: grand
duplikat bazy: grand_dev
nowy uzytkownik bazy: dev_user
haslo: D3eVVbf7

Na poczatku aby skopiowac i zduplikowac baze na serwerze, nalezy stworzyc baze:

mysql -u root -p'q1w2e3!@#' -e "stworzenie bazy grand_dev"

I uzyc nastepujacej komendy aby zduplikowac zawartossc bazy z **grand** to **grand_dev**:

mysqldump -u root -p'q1w2e3!@#' grand | mysql -u root -p'q1w2e3!@#' grand_dev
Format: mysqldump [user] [password] [source database] | mysql [user] [password] [destination database]

Teraz tworzyny uzytkownika MySQL ktory bedzie skojarzony z nowa baza.

$ mysql -u root -p'q1w2e3!@#'

I uruchamiany nastepujace komendy:

mysql> **GRANT USAGE ON** grand_dev.* **TO** dev_user@localhost IDENTIFIED BY 'D3eVVbf7';
mysql> **GRANT ALL PRIVILEGES ON** grand_dev.* **TO** dev_user@localhost ;

Proba dostepu do bazy:

mysql -h localhost -u dev_user -p'D3eVVbf7'

3.3 Przywracanie bazy MySQL

Poniżej przedstawiam procedurę przywracania bazy danych. Najpierw logujemy się do mysql. Kolejna komenda powoduje przywrócenie bazy danych `mysql -u root - p'q1w2e3!@#'`

```
mysql -u root - p'q1w2e3!@#'  grand < /katalog/grand_dev.sq
```

3.4 INNE PRZYDATNE PRZYKLADY:

W jaki sposób przenieść bazę danych z jednego serwera na drugi.
Są na to dwie metody:

1) Przeniesienie bazy danych zgrywając ją do pliku i przenosząc ręcznie przez usługę SFTP/SSH na drugi serwer

Zgrywamy (dump) bazę, którą chcemy przenieść. W tym celu wydajemy komendę.

```
mysqldump -u user -p db-name > db-name.out
```

Kopiujemy używając SFTP/SSH

```
scp db-name.out user@remote.box.com:/backup
```

Wgrywamy bazę na nowym serwerze, wydajemy komendę

```
mysql -u user -p db-name < db-name.out
```

2) Bezpośrednie przenoszenie bazy danych
Możemy bezpośrednio przenieść bazę pomijając defakto proces tworzenia pliku. Możemy to zrobić na kilka sposobów

Zgrywamy mysqldump bazę i zarazem wykorzystując mysql łączymy się ze zdalną instancją i wgrywamy naszą bazę.
Uwaga jest taka, że MySQL serwer docelowy musi nasłuchiwać na publicznym IP, jeżeli nie będzie nie uda się nam podłączyć w ten sposób.

```
mysqldump db-name | mysql -h remote.box.com db-name
```

Ta sama metoda co wyżej, ale bezpieczniejsza (szyfrowanie połączenia). Korzystać będziemy z SSH

```
mysqldump db-name | ssh user@remote.box.com mysql db-name
```

Jeżeli musimy podać użytkownika (i przypisane hasło) do bazy danych (co zapewne w większości przypadków tak będzie) wydajemy komendę

```
mysqldump -u username -p'password' db-name | ssh
user@remote.box.com mysql -u username -p'password db-name
```

Możemy też skopiować tylko wybraną tabelę z bazy danych, tutaj będzie nią tabela o nazwie *foo*

```
mysqldump -u user -p'password' db-name foo | ssh
user@remote.box.com mysql -u user -p'password' db-name foo
```

Można również napisać prosty skrypt robiący kopię w bashu. Potem możemy taki skrypt uruchamiać o określonych porach z crona.

```
1. #!/bin/bash
2. datautw=`date +"%Y-%m-%d"`
3. user='user'
4. password='haslo'
5. host=''
6. mysqldump -u $user -p$password -h $host nazwa_bazy >
   kopia_bazy$datautw.sql
```

INNE SPOSOBY TWORZENIA DUPLIKACJI

```
#!/bin/bash
```

```
DBUSER=user
DBPASSWORD=pwd
DBSNAME=sourceDb
DBNAME=destinationDb
DBSERVER=db.example.com

fCreateTable=""
fInsertData=""
echo "Copying database ... (may take a while ...)"
DBCONN="-h ${DBSERVER} -u ${DBUSER} --password=${DBPASSWORD}"
echo "DROP DATABASE IF EXISTS ${DBNAME}" | mysql ${DBCONN}
echo "CREATE DATABASE ${DBNAME}" | mysql ${DBCONN}
for TABLE in `echo "SHOW TABLES" | mysql $DBCONN $DBSNAME | tail
-n +2`; do
        createTable=`echo "SHOW CREATE TABLE ${TABLE}"|mysql -B
-r $DBCONN $DBSNAME|tail -n +2|cut -f 2-`
        fCreateTable="${fCreateTable} ; ${createTable}"
        insertData="INSERT INTO ${DBNAME}.${TABLE} SELECT * FROM
${DBSNAME}.${TABLE}"
        fInsertData="${fInsertData} ; ${insertData}"
done;
echo "$fCreateTable ; $fInsertData" | mysql $DBCONN $DBNAME
```

Nie jest to metoda szybsza jednak nie dostaniemy błędu związanego z LOCK TABLES w odróżnieniu od `mysqldump`.