Uniwersytet w Białymstoku

Wydział Matematyczno-Fizyczny

Instytut Matematyki

Elżbieta Mądra

Tworzenie kopii zapasowych i replikacja bazy danych MySQL

Praca dyplomowa napisana
pod kierunkiem
dr Mariusza Żynela

Białystok 2005

Spis treści

Wstęp

Tu powinien być tekst wstępu

Rozdział 1

Tabele w MySQL

Ponieważ sposób tworzenia kopii zapasowych jest ściśle związany ze sposobem przechowywania danych, należy najpierw przyjrzeć się jak te dane przechowywane są w bazie MySQL.

Upraszczając można powiedzieć, że baza danych to zestaw wypełnionych tabel. Wszystkie wiersze takiej tabeli wyglądają podobnie. Każda kolumna tabeli przechowuje wartości dla wszystkich wierszy – nie ma przerw ani kolumn skróconych. Tak jest oczywiście również w przypadku MySQL-a. Zwykle z tabelą związany jest jeden lub więcej plików. Ich ilość zależy od typu tabeli i konfiguracji MySQL-a. MySQL tworzy dla każdej bazy danych oddzielny katalog a w nim przechowuje pliki odpowiadające poszczególnym tabelom.

Poniżej przedstawiono zawartość katalogu systemowego MySQL, w którym zgromadzone są wszystkie bazy danych.

```
drwx------   2 mysql    other       2048 kwi 15 11:25 dbMatFiz
-rw-rw----   1 mysql    other      25088 paz  5  2004 ib_arch_log_0000000000
-rw-rw----   1 mysql    other    5242880 maj  6 10:04 ib_logfile0
-rw-rw----   1 mysql    other    5242880 kwi 15 15:53 ib_logfile1
-rw-rw----   1 mysql    other    4040192 maj  5 14:15 ibdata1
-rw-rw----   1 mysql    other    2147145 maj  6 10:04 lab-b01-bin.001
-rw-rw----   1 mysql    other        522 maj  6 10:04 lab-b01-bin.index
-rw-rw----   1 mysql    root       42094 maj  6 10:04 lab-b01.err
-rw-rw----   1 mysql    other          3 maj  6 10:04 lab-b01.pid
drwx------   2 mysql    other        512 paz  5  2004 mysql
drwx------   2 mysql    other        512 kwi 15 15:08 patchdb
drwx------   2 mysql    other        512 sty  5 11:57 student
drwx------   2 mysql    other        512 paz  5  2004 test
```

Katalogi `dbMatFiz`, `mysql`, `patchdb`, `student` i `test` odpowiadają bazom danych w MySQL. Pliki z przedrostkiem `ib` związane są z tabelami typu InnoDB, natomiast pliki zaczynające się na `lab-b01` związane są z konkretną instalacją

MySQL, w tym wypadku na komputerze o nazwie `lab-b01`.

Tabele w MySQL-u tworzymy poleceniem CREATE TABLE. Składnia tego polecenia jest następująca:

```
CREATE TABLE nazwa_tabeli (deklaracja kolumn) TYPE=rodzaj tabeli;
```

Z punktu widzenia tej pracy nie jest istotne, jak deklaruje się kolumny, istotna jest natomiast opcja TYPE, na której się teraz skupimy. Za pomocą tej opcji określamy rodzaj tabeli, to znaczy jak ma ona być przechowywana w MySQL-u. Z typem związane są ściśle konkretne możliwości lub ograniczenia co do funkcjonalności tabel. Kluczową funkcjonalnością ze względu na którą dzielimy tabele na dwie grupy są transakcje.

Transakcja to zbiór operacji, które mogą być wykonane jedynie wszystkie lub żadna. Nazwa pochodzi od operacji bankowych - przelew musi jednocześnie zabrać z jednego konta i dodać na drugie. W przypadku niepowodzenia żadna z tych operacji nie powinna mieć miejsca. Jeśli zajdzie tylko jedna skutki były by potencjalnie katastrofalne. Transakcje opisuje zasada ACID - Atomicity, Consistency, Isolation and Durability[1]:

- **Atomicity** - transakcja może być wykonana tylko w całości albo wcale.
- **Consistency** - stan bazy danych zawsze przedstawia stan przed lub po transakcji. Zapytania składane systemowi w czasie wykonywania transakcji muszą pokazywać sytuację przed transakcją, nie sytuację przejściową.
- **Isolation** - transakcja dzieje się niezależnie od innych wykonywanych operacji, w tym od innych transakcji.
- **Durability** - w przypadku krachu całego systemu bazodanowego, np. w wyniku odcięcia elektryczności, transakcja będzie albo wykonana w całości albo wcale.

Bazy danych implementują transakcje, prowadząc dziennik wszystkich zmian dokonanych przez każdą z transakcji. Z chwilą cofnięcia transakcji system DBMS[2] korzysta z tego dziennika, by przywrócić spójny stan sprzed rozpoczęcia transakcji.

Żeby wycofać transakcję należy użyć instrukcji ROLLBACK. Potwierdzenia transakcji dokonamy zapytaniem COMMIT.

[1] Niepodzielność, Spójność, Izolacja i Trwałość
[2] System Zarządzania Bazą Danych (Database Management System)

Transakcje mają sens jedynie podczas aktualizacji tabeli przy użyciu kwerend INSERT i UPDATE. Nie stosuje się ich do zmian w schemacie bazy. Instrukcje ALTER TABLE, BEGIN, CREATE INDEX, DROP DATABASE, DROP TABLE, RENAME i TRUNCATE kończą transakcję.

Tak, jak zostało powiedziane wcześniej, typy tabel dzielimy na dwie grupy:

Tabele z transakcjami	Tabele bez transakcji
HEAP	BDB(BerkeleyDB)
ISAM	InnoDB
MyISAM	Gemini
Merge	

Jeśli nie użyjemy polecenia TYPE, MySQL domyślnie utworzy nam tabelę typu MyISAM.

1.1 Tabele HEAP

Tabele HEAP (z ang. Stos) począwszy od wersji 4.1 możemy nazywać też MEMORY. Są one przechowywane nie w systemie plików, ale w pamięci operacyjnej komputera. W rezultacie działają bardzo szybko, ale podczas awarii systemu zawarte w nich dane giną bezpowrotnie. Dlatego też typ ten jest bardzo użyteczny do tworzenia tymczasowych tabel pomocniczych. W przypadku rozłączenia z serwerem struktura tabeli zostanie, ponieważ jest ona zapisana w pliku `*.frm`, znajdującym się na dysku, ale będzie to tylko struktura. Dlatego też typ ten jest bardzo użyteczny do tworzenia tymczasowych tabel pomocniczych.

Tabele te nie mają wielu właściwości istniejących w innych typach tabel. Nie mogą zawierać kolumn typu BLOB ani TEXT. Opcja AUTO_INCREMENT jest dla nich dostępna dopiero od wersji 4.1.0. Można tworzyć w nich indeksy, ale nie można indeksować kolumn, które mogą zawierać wartości puste (null). Typ HEAP pozwala użyć 32 indeksów w jednej tabeli, maksymalna długość klucza wynosi 500 bajtów. Indeksy są używane jedynie do wyrażeń "=" i "⟨=⟩".

Rekordy tabel HEAP mają stałą długość. MySQL dynamicznie alokuje całą przewidzianą długość kolumn typu `varchar`.

Użytkownicy korzystają z tabel HEAP tak jak ze wszystkich innych, ale tabele te znikają w chwili rozłączenia klienta z serwerem.

1.2 Tabele ISAM

W wersjach starszych niż 3.23 ISAM był domyślnym typem tabel. Choć jest on w dużej mierze zgodny ze swym następcą - MyISAM. Ma on jednak mniejsze możliwości, dlatego też obecnie jest niemal w ogóle nie używany. Autorzy MySQL-a usunęli ten typ w wersji 5.0.3.

Tabele ISAM można przekonwertować na typ MyISAM (powinniśmy to zrobić, jeśli zamierzamy używać najnowszych wersji MySQL-a) poleceniem:

```
ALTER TABLE nazwa_tabeli TYPE=myisam;
```

Dane w ISAM są przechowywane w postaci plików uporządkowanych. Rozszerzenia tych plików to:

`*.ISM` - indeksy,

`*.ISD` - dane.

Indeksy przechowywane są w postaci drzew binarnych[3]. Maksymalna długość klucza wynosi 256. Dane są zapisywane w zależności od środowiska (platformy), co daje szybkość, ale kosztem przenośności. Tabele w ISAM nie mogą mieć więcej niż 4GB.

Narzędzie do naprawy i optymalizacji tabel ISAM nazywa się `isamchk`, a narzędzie kompresujące to `pack_isam`.

1.3 Tabele MyISAM

MyISAM jest domyślnym typem tabel począwszy od wersji 3.23, choć można to zmienić w pliku konfiguracyjnym. Tabele tego typu posiadają pewne specjalne narzędzia, pozwalające operować na odpowiadającym im plikach. Narzędzia te to `myisamchk`, służący do sprawdzania poprawności i naprawiania tabel oraz `myisampack`, który kompresuje tabele MyISAM.

Tabele MyISAM bazują na kodzie ISAM, jednakże posiadają istotne rozszerzenia takie jak:

- Podczas startu `mysqld`[4] tabele MyISAM są automatycznie naprawiane i sprawdzane.

[3]Drzewo binarne to taka struktura, w której każdy element (z wyjątkiem jednego elementu, który jest główny) ma jeden element nadrzędny i co najwyżej 2 elementy podrzędne. Nie występują w nim żadne cykle.

[4]`mysqld` to nazwa programu wykonywalnego bazy MySQL. Tak też zatem nazywa się proces w systemie odpowiadający uruchamianej bazie MySQL.

- Możliwe jest odczytanie tabeli, podczas gdy inny użytkownik wykonuje operację INSERT, o ile tabela nie zawiera "dziur" powstałych w wyniku użycia instrukcji DELETE.
- Obsługa dużych plików (powyżej 4 GB) dla systemów ich używających.
- W inny sposób obsługują AUTO_INCREMENT. W przypadku użycia tej kwerendy MySQL zapamiętuje licznik i zwiększa jego wartość, co jest nieznacznie szybsze, niż gdyby miał dodawać 1 do największej wartości z kolumny, jak robi to ISAM. W tabelach ISAM, jeśli zostanie usunięty wiersz z największym numerem, ten największy numer będzie użyty ponownie. W MyISAM wartości pominięte już nigdy nie zostaną użyte ponownie.
- Kolumny BLOB i TEXT można indeksować.
- NULL jest dozwoloną wartością indeksowanej kolumny.
- Maksymalna długość klucza wynosi 1000B (500 bajtów dla wersji starszych niż 4.1.2).
- Maksymalna ilość indeksów w tabeli wynosi 32 (64 po przekompilowaniu `myisamchk`) dla wersji MySQL poniżej 4.1.2. W wersjach nowszych maksymalna ilość indeksów wynosi 64 bez użycia `myisamchk`.
- Format plików związanych z tabelą MyISAM nie zależy od platformy, możemy zatem przenosić je między różnymi systemami o różnej budowie, bez konieczności dokonywania w nich zmian.

MySQL rejestruje liczniki połączeń z MyISAM. Jeśli tabela nie jest używana, licznik zwraca 0. Gdy serwer nagle ulegnie awarii, licznik zostaje z wartością większą od 0. Przy uruchomieniu serwera stan licznika różny od zera sygnalizuje problem. Może (ale wcale nie musi) oznaczać to, że tabela jest uszkodzona. W takiej sytuacji należy użyć polecenia CHECK TABLE lub programu `myisamchk`. Można też uruchomić `mysqld` z opcją `--myisam recover` wymuszającą naprawę wszystkich tabel MyISAM, dla których licznik jest różny od zera.

Każda tabela MyISAM jest przechowywana w trzech plikach o rozszerzeniach:

`*.frm` - dane dotyczące definicji tabeli i jej kolumn,

`*.MYI` - dane dotyczące indeksów (myindex),

`*.MYD` - fizyczne dane (mydata).

Oznacza to, że tabela o nazwie `dane` będzie przechowywana w plikach o nazwach `dane.frm`, `dane.MYI` i `dane.MYD`. Poniżej przedstawiona jest lista plików z katalogu `dbMatFiz`, który widzieliśmy wcześniej:

```
-rw-rw----   1 mysql    other      12876 mar 11 12:09 tCzasopisma.MYD
-rw-rw----   1 mysql    other       6144 mar 11 12:09 tCzasopisma.MYI
-rw-rw----   1 mysql    other       8888 mar 11 12:09 tCzasopisma.frm
-rw-rw----   1 mysql    other      33412 mar 11 12:09 tPracownicy.MYD
-rw-rw----   1 mysql    other      10240 mar 11 12:09 tPracownicy.MYI
-rw-rw----   1 mysql    other      10056 mar 11 12:09 tPracownicy.frm
-rw-rw----   1 mysql    other      71944 kwi 15 11:28 tPublikacje.MYD
-rw-rw----   1 mysql    other       5120 kwi 15 16:28 tPublikacje.MYI
-rw-rw----   1 mysql    other       9458 mar 11 12:09 tPublikacje.frm
-rw-rw----   1 mysql    other        268 mar 18 14:55 tSeminaria.MYD
-rw-rw----   1 mysql    other       4096 mar 18 15:07 tSeminaria.MYI
-rw-rw----   1 mysql    other       8854 mar 11 12:09 tSeminaria.frm
-rw-rw----   1 mysql    other       8600 mar 18 14:46 tZaklady.MYD
-rw-rw----   1 mysql    other       5120 mar 18 14:55 tZaklady.MYI
-rw-rw----   1 mysql    other       9162 mar 11 12:09 tZaklady.frm
```

W bazie dbMatFiz wszystkie tabele są typu MyISAM. W przypadku utraty pliku `*.MYI` MySQL odbudowuje indeks korzystając z informacji z pliku `*.frm`. Indeksy MySQL są drzewami binarnymi.

MyISAM wspiera trzy różne formaty tabel zwane FIXED, DYNAMIC i COMPRESSED. Oznacza to, iż w tabelach MyISAM używane są wiersze odpowiednio: stałe, dynamiczne lub skompresowane. Tabele skompresowane tworzy się jedynie za pomocą narzędzia `myisampack`. Tabele stałe i dynamiczne są automatycznie zależne od typu kolumn w nich użytych.

1.4 Tabele Merge

Tabele Merge (Złączenie) są dostępne w MySQL-u poczynając od wersji 3.23.25. Na tabelę Merge składa się kilka tabel MyISAM o identycznej strukturze, które możemy traktować jak jedną tabelę. "Identyczna struktura" oznacza takie same kolumny i takie same indeksy. Jeżeli w składowych tabelach nie zgadza się kolejność kolumn, bądź indeksów, nie możemy ich "skleić" w tabelę Merge. Przy tworzeniu tabeli Merge MySQL tworzy dwa nowe pliki o rozszerzeniach:

`*.MRG` - który zawiera informacje o tabelach wchodzących w skład tabeli Merge oraz o ich indeksach

`*.frm` - który zapamiętuje definicję tabeli

Początkowo wszystkie złączane tabele musiały znajdować się w tej samej bazie danych, co tabela MERGE. Tę niedogodność usunięto w wersji MySQL 4.1.1. W przypadku złączania dwóch lub więcej tabel ich wiersze traktuje się tak, jakby należały do pojedynczej tabeli. Jedyne operacje, które możemy wykonywać na tabeli Merge to SELECT, UPDATE, DELETE i (od wersji 4.0 począwszy) operację INSERT. Wcześniej nie można było dodawać wierszy bezpośrednio

do złączanych tabeli[5], ponieważ MySQL nie potrafił rozstrzygnąć w takiej sytuacji, do której tabeli składowej wiersz ów miałby być dodany. Obecnie wśród opcji kwerendy CREATE TABLE znajduje się instrukcja INSERT_METHOD, która określa do której z tabel składowych (licząc w kolejności podania): pierwszej (FIRST), czy ostatniej (LAST) wstawiane są wiersze. Jeżeli pominiemy to zapytanie, lub wstawimy wartość NO, wtedy każda próba wstawiania wartości bezpośrednio do tabeli Merge spowoduje błąd.

Przy tworzeniu tabeli Merge musimy pamiętać, aby w składni polecenia podać nazwy tabel składowych. Służy do tego instrukcja UNION=(tabela1, tabela2,...) w poleceniu CREATE TABLE.

Poniżej znajduje się przykład tabeli Merge. Pierwsze sześć poleceń to utworzenie i wypełnienie danymi tabel składowych. Następna instrukcja, to tworzenie tabeli Merge, a ostatnia pokazuje jak wygląda nowo utworzona tabela Merge:

```
mysql> create table T1 (
    -> ID int not null auto_increment primary key,
    -> wiadomosc char(30));

mysql> insert into T1 values (
    -> '','To jest pierwszy wiersz T1');

mysql> insert into T1 values (
    -> '','A to jest drugi wiersz T1');

mysql> create table T2 (
    -> ID int not null auto_increment primary key,
    -> plotka char(30));

mysql> insert into T2 values (
    -> '','To juz tabela T2');

mysql> insert into T2 values (
    -> '','a to jej drugi wiersz');

mysql> create table Total (
    -> ID int not null auto_increment,
    -> tekst char(30), index(ID))
    -> type=MERGE union=(T1,T2);

mysql> select*from Total;
+----+----------------------------+
| ID | tekst                      |
+----+----------------------------+
```

[5]Można było je wstawić tylko do tabel składowych.

```
|  1 | To jest pierwszy wiersz T1 |
|  2 | A to jest drugi wiersz T1  |
|  1 | To juz tabela T2           |
|  2 | a to jej drugi wiersz      |
+----+----------------------------+
```

W poleceniu tworzącym tabelę znajduje się nota o tym, że kolumną indeksowaną w tabeli Total jest ID, ale nie jest ona zadeklarowana jako PRIMARY KEY, jak w przypadku tabel składowych T1 i T2. Takie rozwiązanie jest konieczne, gdyż Merge nie mogą utworzyć unikalnego klucza.

Usunięcie tabel Merge (poleceniem DROP TABLE) powoduje jedynie usunięcie tabeli logicznej (tabele składowe pozostają).

Tabele typu Merge mają pewne niedogodności. Tabele złączane korzystają z większej liczby deskryptorów plików, gdyż MySQL musi otworzyć kolejno wszystkie tabele składowe. Inną wadą jest fakt, iż tabele Merge nie utrzymują unikalności kluczy.

Główną zaletą takich złączanych tabel jest możliwość traktowania wielu tabel jak jednej, a przy tym normalnego odwoływania się do tabel składowych. Jest to idealny sposób dzielenia tabel. Praca z tabelą Merge będzie szybsza niż z jedną dużą tabelą.

1.5 Tabele InnoDB

Tabele InnoDB zostały stworzone przez fińskiego programistę nazwiskiem Heikki Tuuri z Innobase Oy - fińskiej firmy informatycznej, która specjalizuje się w technologii relacyjnych baz danych. InnoDB jest efektem prac badawczych pana Tuuri na Uniwersytecie Helsińskim. Obsługa tabel InnoDB została wprowadzona do MySQL w wersji 3.23.34a. Jest to aktualnie najbardziej rozwijający się typ tabel MySQL-a. W wersjach 5 wiele zmian dotyczy właśnie tego typu.

Ten tryb przechowywania danych powstał z myślą o uzyskaniu maksymalnych osiągów przy przetwarzaniu danych o dużych rozmiarach. Doskonale się nadaje do tabel rzędu TB[6].

Jest to pierwszy typ tabel obsługujący klucze obce[7] (ang. foreign keys). Nie nakładają one jednak automatycznie indeksów na klucze obce.

[6]Przykładem może być firma Mytrix Inc., która używa tabel InnoDB większych niż 1 TB i wykonuje na nich średnio 800 operacji INSERT/UPDATE na minutę.

[7]Klucze obce są sposobem łączenia danych przechowywanych w różnych tabelach.

InnoDB przechowuje dane w dużych współdzielonych plikach. Można utworzyć tyle plików, ile potrzeba, jednak potem nie można ich usunąć. Wielkość tych plików określana jest w pliku konfiguracyjnym, jednak nie może przekroczyć 4GB.

Tabele InnoDB blokowane są na poziomie rekordów. Blokowanie odbywa się niejawnie podczas wykonywania instrukcji w ramach transakcji. Po użyciu instrukcji LOCK TABLES istnieje ryzyko konfliktu z blokadami InnoDB. Może to spowodować zakleszczenie przy próbach odczytu lub modyfikacji tabeli.

Tabele InnoDB korzystają z dzienników do odwoływania transakcji. InnoDB tworzy duży plik i używa tej samej pamięci cyklicznie. Kiedy zajęty jest cały dziennik, usuwane są najstarsze wpisy.

Z tabelami InnoDB związane są pewne ograniczenia, które zapewne szybko zostaną usunięte. Najważniejszym z nich jest interakcja między sposobem śledzenia tabel w MySQL a katalogiem tabel InnoDB. Usunięcie bazy danych nie powoduje usunięcia tabel z katalogu InnoDB; aby uniknąć tego problemu powinniśmy usuwać tabele pojedynczo. Poza tym InnoDB nie pozwala indeksować przedrostka kolumny, nie można również indeksować kolumn BLOB ani TEXT. Liczba kolumn jest ograniczona do 1000. W instrukcji INSERT nie można używać flagi DELAYED.

Poniżej znajduje się zawartość katalogu `patchdb`, który widzieliśmy na początku:

```
-rw-rw----   1 mysql    other       8724 kwi 15 15:08 hosts.frm
-rw-rw----   1 mysql    other       8636 kwi 15 15:08 showrev.frm
-rw-rw----   1 mysql    other       8822 kwi 15 15:08 users.frm
-rw-rw----   1 mysql    other       8834 kwi 15 15:08 xref.frm
```

W katalogu tym wszystkie tabele są typu InnoDB.

1.6 Tabele BerkeleyDB

Projekt BerkeleyDB powstał na uniwersytecie Kalifornijskim w Berkeley. W MySQL-u obsługa tabel Berkeley DB (w skrócie BDB) została dołączona w wersji 3.23.34 MAX. Jego autorzy otworzyli później firmę Sleepycat Software oferującą wersję komercyjną.

BerkeleyDB to baza danych w plikach płaskich obsługująca transakcje. Nie zawiera żadnego języka zapytań: MySQL łączy się z BDB, aby umożliwić użytkownikom dostęp do tabel z transakcjami. Zespół MySQL ściśle współpracuje z pracownikami Sleepycat, aby w jak największym stopniu wykorzystać bibliotekę BDB.

Każda tabela BDB jest przechowywana na dysku w dwóch plikach. Nazwy tych plików zaczynają się nazwą tabeli, a rozszerzenia mają następujące:

`*.frm` – który zawiera definicję tabeli,

`*.db` – zawierający dane oraz indeksy.

BDB jest kolejnym typem tabel w MySQL, który umożliwia korzystanie z transakcji. W tabelach BDB do realizacji transakcji używa się tabel danych. Jeśli transakcja jest wycofywana, biblioteka BDB wykorzystuje dziennik do cofania zmian. Nazwy dzienników są typu `log.0000000001`; pliki te są umieszczane w katalogu z danymi. Zaleca się jednak, aby umieścić je na innym dysku, ponieważ wpływa to korzystnie na wydajność bazy danych. Można to zrobić używając jednej z funkcji `mysqld`: `--bdb-logdir`.

Funkcjonalnie tabele BerkeleyDB są podobne do tabel MyISAM. Nie ma tu ograniczeń dla kolumn i indeksów, ale tabele BDB wymagają istnienia klucza głównego. Jeśli użytkownik nie poda klucza głównego, MySQL sam utworzy wewnętrzny, pięciobajtowy klucz główny, który będzie się zwiększał przy każdej operacji wstawiania wierszy, podobnie jak przy AUTO_INCREMENT. Pracując z tabelami BDB można używać instrukcji LOCK TABLES, chociaż zaleca się raczej korzystać z transakcji.

BDB blokuje tabele na poziomie stron. Strona jest zbiorem sąsiadujących ze sobą wierszy tabeli.

Dane tabel są przechowywane w drzewach binarnych. Biblioteka zapamiętuje "dziury" w drzewie, aby przyspieszyć potem wstawianie danych, co sprawia, że drzewa są większe niż jest to naprawdę konieczne. Indeksy tabel BDB nie są kompresowane, jak ma to miejsce w tabelach MyISAM, więc zajmuje więcej miejsca. Możliwe jest w przypadku tych tabel łączne użycie klucza głównego i jakiegoś innego indeksu.

1.7 Tabele Gemini

Kod obsługujący tabele Gemini stworzyli programiści z NuSphere - firmy oferującej usługi związane z obsługą i szkoleniem w zakresie MySQL. Testy fazy beta zaczęły się w kwietniu 2001.

Tabele Gemini nie pozwalają używać typu BLOB ani TEXT. Domyślnie ograniczają liczbę użytkowników do 100, ale można to zmienić.

W przypadku Gemini blokowanie odbywa się na poziomie wiersza, przez co większa liczba użytkowników może korzystać z bazy. Blokady tabeli powodują, że inni użytkownicy nie mają dostępu do tabeli i muszą czekać. Blokady

na poziomie wierszy uniemożliwiają sięgnięcie do pojedynczych wierszy, dzięki czemu różne wątki, o ile tylko sięgają do różnych wierszy, mogą korzystać z tabeli jednocześnie.

Rozdział 2

Sposoby tworzenia kopii zapasowych

2.1 mysqldump

`Mysqldump` jest programem dostępnym w standardowej wersji MySQL. Program ten pobiera dane z bazy i tworzy kod SQL za pomocą którego można odtworzyć tabele, wiersze i zawarte w nich dane w innej bazie danych na innym serwerze MySQL. Kod ten zawiera instrukcje CREATE TABLE i INSERT. Zostaje on przesłany do `stdout`[1] i może być przekierowany do pliku, tak jak w poniższym przykładzie:

```
mysqldump opcje nazwa_bazy_danych > nazwa_pliku.sql
```

W pliku `nazwa_pliku.sql` tworzona jest dokładna kopia wszystkich tabel z bazy `nazwa_bazy_danych`. Poniżej umieszczone są dwie takie kopie, obie wykonane poleceniem:

```
mysqldump --opt -u root dbMatFiz.tPublikacje

-- MySQL dump 9.08
--
-- Host: localhost    Database: dbMatFiz
---------------------------------------------------------
-- Server version       4.0.14

--
-- Table structure for table 'tPublikacje'
--

DROP TABLE IF EXISTS tPublikacje;
CREATE TABLE tPublikacje (
```

[1] Standardowe wyjście, konsola, ekran.

```
  cID int(10) unsigned NOT NULL auto_increment,
  cTS timestamp(14) NOT NULL,
  cCT timestamp(14) NOT NULL,
  cOWNER varchar(16) default NULL,
  cGROUP varchar(16) default NULL,
  cKind varchar(16) NOT NULL default '',
  cInst varchar(64) default NULL,
  cReported enum('T','N') default 'N',
  cAuthor varchar(128) default NULL,
  cTitle varchar(128) default NULL,
  cJournal varchar(64) default NULL,
  cVolume varchar(8) default NULL,
 [...]
  PRIMARY KEY  (cID)
) TYPE=ISAM PACK_KEYS=1;

--
-- Dumping data for table 'tPublikacje'
--

/*!40000 ALTER TABLE tPublikacje DISABLE KEYS */;
LOCK TABLES tPublikacje WRITE;
INSERT INTO tPublikacje VALUES [...];
UNLOCK TABLES;
/*!40000 ALTER TABLE tPublikacje ENABLE KEYS */;
```

W powyższym przykładzie dla skrócenia wyrzucono fragmenty i zastąpiono je `[...]`. Jest to wynik `mysqldump` z MySQL-a w wersji 4.0.14.

W nowszych wersjach MySQL-a (w tym wypadku 4.1.9) `mysqldump` wykonuje kilka dodatkowych, przydatnych rzeczy. W poniższym przykładzie również usunięto niektóre fragmenty i zastąpiono je `[...]`:

```
-- MySQL dump 10.9
--
-- Host: localhost    Database: dbMatFiz
-- ------------------------------------------------------
-- Server version       4.1.9-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES latin2 */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE="NO_AUTO_VALUE_ON_ZERO" */;
```

```
--
-- Table structure for table 'tPublikacje'
--

DROP TABLE IF EXISTS tPublikacje;
CREATE TABLE tPublikacje (
  cID int(10) unsigned NOT NULL auto_increment,
  cTS timestamp(14) NOT NULL,
  cCT timestamp(14) NOT NULL,
  cOWNER varchar(16) default NULL,
  cGROUP varchar(16) default NULL,
  cKind varchar(16) NOT NULL default '',
  cInst varchar(64) default NULL,
  cReported enum('T','N') default 'N',
  cAuthor varchar(128) default NULL,
  cTitle varchar(128) default NULL,
  cJournal varchar(64) default NULL,
  cVolume varchar(8) default NULL,
  [...]
  PRIMARY KEY  (cID)
) TYPE=ISAM PACK_KEYS=1;

--
-- Dumping data for table 'tPublikacje'
--

/*!40000 ALTER TABLE `tPublikacje` DISABLE KEYS */;
LOCK TABLES `tPublikacje` WRITE;
INSERT INTO `tPublikacje` VALUES [...];
UNLOCK TABLES;
/*!40000 ALTER TABLE `tPublikacje` ENABLE KEYS */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Jak widać `mysqldump` w nowszej wersji dba dodatkowo o prawidłowe odtworzenie kodowania znaków (SET NAMES) i kluczy obcych (FOREIGN_KEY_CHECKS).

Używając `mysqldump` do zrzutu tabel InnoDB należy pamiętać o tym, że mogą wystąpić problemy, jeśli tabele te zawierają klucze obce. Problem ten pojawia się podczas prób odtworzenia tabel w niewłaściwej kolejności. W wersji 4.1.x problem ten został rozwiązany poprzez dodanie do "zdumpowanego" pliku instrukcji powodującej ignorowanie kluczy obcych przy odtwarzaniu danych,

a po ich odtworzeniu dump dodaje instrukcję, która przywraca klucze obce. W poprzedniej wersji wykonanie kopii tabel zawierających klucze obce bez dodatkowych zabiegów nie było wręcz możliwe.

Z pliku uzyskanego przez `mysqldump` możemy później odtworzyć dane w następujący sposób:

```
mysql < nazwa_pliku.sql
```

Niektóre opcje z jakimi występuje `mysqldump` to[2]:

`--add-drop table`
: Powoduje umieszczanie przed każdą instrukcją CREATE TABLE instrukcji DROP TABLE IF EXISTS.

`--add-locks`
: Blokuje tabele dla zapisu przed wstawianiem wierszy. Przed grupą instrukcji wstawiających pojawia się wówczas instrukcja LOCK TABLES, a po niej następuje UNLOCK TABLES.

`--all-databases (-A)`
: Powoduje "zrzucenie" wszystkich baz danych. Korzystając z tej opcji nie należy wprowadzać listy baz danych.

`--allow-keywords`
: Uzupełnia kolumny nazwane słowami kluczowymi o prefiks w postaci nazwy tabeli.

`--character-sets-dir=/ścieżka/do/katalogu`
: Określa ścieżkę do zestawów znaków.

`--complete-insert (-c)`
: Wstawia listy kolumn do instrukcji INSERT.

`--compress (-C)`
: Powoduje pakowanie informacji przesyłanych między klientem a serwerem.

`--create-options`
: Włącza wszystkie specyficzne opcje tabel w MySQL do instrukcji CREATE TABLE. Opcja ta pojawiła się w wersji 4.1.2.

`--databases (-B)`
: Opcja ta umożliwia wskazanie kilku baz danych do archiwizacji. Wyklucza ona stosowanie listy tabel. Wszystkie podawane nazwy powinny być nazwami baz danych.

[2]Pełna lista opcji znajduje się na stronie `http://dev.mysql.com/doc/mysql/en/mysqldump.html`

`--debug`
Funkcja nakazuje `mysqldump` zapis informacji diagnostycznych do pliku. Klient musi być odpowiednio skompilowany, aby skorzystać z tej opcji. Generowana treść będzie zapisywana domyślnie w `/tmp/client.trace`

`--default-character-set=nazwa_pliku`
Zmienia domyślny zestaw znaków, ustawiony z chwila kompilacji MySQL. Wybrany zestaw znaków musi znajdować się w katalogu wskazywanym przez opisaną wcześniej opcje `--character-sets-dir`.

`--delayed-insert`
Używa słowa kluczowego DELAYED podczas operacji wstawiania.

`--disable-keys (-K)`
Dla każdej tabeli, otacza instrukcje INSERT poleceniami:

```
/*!40000 ALTER TABLE nazwa_tabeli DISABLE KEYS */;
    i
/*!40000 ALTER TABLE nazwa_tabeli ENABLE KEYS */;
```

To powoduje "ładowanie" pliku na serwer MySQL 4.0 szybciej, ponieważ indeksy są tworzone po wstawieniu wszystkich wierszy. Ta opcja jest efektywna tylko w wypadku tabel MyISAM.

`--extended-insert (-e)`
Powoduje, że dane ze wszystkich wierszy umieszczane są w jednej instrukcji INSERT. To powoduje, że "zdumpowany" plik jest mniejszy, a wczytywanie danych jest szybsze.

`--flush-logs (-F)`
Opróżnia logi przed rozpoczęciem archiwizacji.

`--force (-f)`
Wymusza kontynuację archiwizacji po wystąpieniu błędu.

`--help (-?)`
Zwraca listę dostępnych opcji i kończy działanie.

`--host=nazwa (-h nazwa)`
Łączy się z podanym hostem określonym przez adres IP lub nazwę domeny.

`--lock tables (-l)`
Blokuje wszystkie tabele dla odczytu przed rozpoczęciem procesu archiwizacji.

`--no-create-db (-n)`
Instrukcja ta sprawia, że przy zastosowaniu opcji `--all-databases` lub `--databases` nie jest generowane zapytanie CREATE DATABASE (dump "zrzuca" same tabele bez tworzenia baz danych, w których się te tabele znajdują).

`--master-data[=wartość]` Opcja ta powoduje , że dodatkowo wypisywana jest pozycja w logu binarnym i nazwa pliku. Ta opcja wymaga przywileju RELOAD.
Jeśli wartość podana w poleceniu jest równa 1, pozycja ta wraz z nazwą pliku są wpisywane do dumpowanego pliku w formie instrukcji CHANGE MASTER[3], co powoduje, że serwer SLAVE "startuje" z prawidłową pozycją dziennika binarnego serwera MASTER[4]. Jeśli wartość ta wynosi 2, instrukcja CHANGE MASTER pisana jest jako komentarz SQL. Jest to działanie domyślne, zatem jeśli pominiemy tę część polecenia, MySQL potraktuje to tak, jakbyśmy podali wartość 2. Wartość możemy podawać począwszy od wersji 4.1.8.

`--no-create-info (-f)`
Sprawia, że w tworzonym pliku nie zostanie zawarta informacja o utworzeniu tabeli (CREATE TABLE). W ten sposób powstanie kopia samych danych (instrukcje INSERT), a nie struktura tabel.

`--no-data (-d)`
W pliku zostanie wygenerowana informacja o strukturze tabel, a dane zostaną pominięte (utworzony za pomocą tej opcji plik nie zawiera instrukcji INSERT).

`--opt`
Najczęściej używana opcja. Reprezentuje ona ustawienia optymalne. Uaktywnia ona następujące funkcje: `--add-drop table`, `--add-locks`, `--extended-insert`, `--lock-tables` i `--quick`; w nowszych wersjach także `create-options`, `--disable-keys` oraz `--set-charset`.
Od wersji 4.1 `--opt` jest opcją domyślną. Możemy ją wyłączyć poleceniem `--skip-opt`. Można w ten sposób wyłączyć tylko jedną z opcji wchodzącą w skład `--opt` np. `--skip-add-locks` lub `--skip-quick`.

`--password[=hasło] (-p[hasło])`
Określa hasło stosowane przy połączeniu z serwerem. Jeśli używamy skróconej wersji tej opcji, powinniśmy pamiętać, że pomiędzy `-p` a hasłem nie może być spacji.

[3]Patrz rozdział 3
[4]Jeśli używamy tej opcji `mysqldump`, aby odpowiednio przygotować replikację na serwerze SLAVE

`--port=port (-P port)`
Opcja zmienia używany przez klienta numer portu (domyślnie jest nim 3306).

`--quick (-q)`
Wyłącza buforowanie treści, zapisując ją bezpośrednio do `stdout`.

`-- socket=ścieżka_dostępu (-S ścieżka_dostępu)`
Zmienia domyślne gniazdo używane do łączenia się z serwerem. Typowy plik gniazda to `/tmp/mysql.sock`.

`--tables`
Niweluje działanie opcji `--databases`.

`--user=nazwa (-u nazwa)`
Określa nazwę użytkownika używaną podczas logowania się do bazy danych.

`--verbose (-v)`
Sprawia, że do wygenerowanej treści dodawanych jest kilka komentarzy opisujących kolejno wykonywane kroki.

`--set-charset`
Opcja dodana w wersji 4.1.2. Dodaje SET NAMES `default_character_set` na wyjście.

`--single-transaction`
Opcja `--single-transaction` została dodana do MySQL-a w wersji 4.0.2. Wydaje ona polecenie BEGIN SQL przed zrzutem danych z serwera. Jest to użyteczne tylko w przypadku tabel InnoDB, ponieważ wtedy `mysqldump` tworzy kopię spójnego stanu bazy danych w czasie, gdy BEGIN było zawołane bez blokowania żadnych aplikacji.
Podczas używania tej opcji, należy pamiętać, że tylko tabele InnoDB są zrzucane w spójnym stanie. Na przykład, każda z tabel MyISAM lub HEAP zrzucanych przy użyciu tej opcji może zmienić swój stan.
Opcja ta z opcją `--lock-tables` wzajemnie się wykluczają, gdyż LOCK TABLES powoduje, że rozpoczęte transakcje są kończone poprzez COMMIT.
Aby zrzucać większe tabele, zaleca się stosować tę opcję razem z opcją `--quick`.

`--where=warunki (-w warunki)`
Opcja ta powoduje, że na wierszach wstawionych za pomocą INSERT stosowana jest klauzula WHERE. Powinniśmy pamiętać o umieszczeniu całej treści opcji w apostrofach, jeśli chcemy uniknąć problemów z interpretacją argumentów podawanych w wierszu poleceń. Przykład użycia tej instrukcji:

```
mysqldump "--where imie='Karolina'" dane > dane.sql;
```

`-xml (-X)`
: Tworzy plik w języku XML.

Pamiętajmy, że metoda ta generuje bardzo duży ruch w sieci, szczególnie jeśli posiadamy dużą bazę danych. Tak naprawdę jest to największa przeszkoda do wykonywania kopii zapasowych całych baz danych poprzez sieć. Jeżeli jednak wykonujemy kopię zapasową lokalnie, program jest wprost idealny do tego typu zadania.

2.2 mysqlhotcopy

`Mysqlhotcopy` jest skryptem napisanym w Perlu przez Tima Bunce'a, który umożliwia tworzenie kopii zapasowych uruchomionej bazy danych. Narzędzie to blokuje wszystkie tabele w bazie (poleceniem LOCK TABLES) i kasuje zawartość wewnętrznych buforów (FLUSH TABLES), po czym kopiuje ich zawartość do wskazanego katalogu.
`Mysqlhotcopy` jest najszybszym sposobem robienie kopii zapasowych całych baz danych lub tylko wybranych tabel. Z pomocą tego skryptu nie możemy jednak skopiować danych na inny komputer. Narzędzie `mysqlhotcopy` działa sprawnie w systemie UNIX, a od wersji 4.0.18 również na NetWare[5]. Skrypt ten wymaga modułu DBI Perla.

Składnia użycia:

```
mysqlhotcopy [opcje] nazwa_tabeli  /ścieżka/do/katalogu
```

Opcje z jakimi występuje mysqlhotcopy

`--allowold`
: Opcja ta zachowuje istniejące archiwa w tymczasowym katalogu archiwum. Jeżeli archiwizacja się powiedzie, stare archiwa zostaną usunięte. Jeżeli archiwizacja się nie powiedzie, wówczas stare archiwa są przywracane.

`--checkpoint=tabela`
: Opcja ta wstawia wiersz do tabeli kontrolnej z chwilą utworzenia kopii zapasowej każdej z tabel. Konieczne jest wskazanie tabeli wraz z nazwą bazy danych. Tabela kontrolna musi mieć następujące kolumny, ale może zawierać także inne:

  - time_stamp TIMESTAMP NOT NULL

[5] Marka oprogramowania sieci lokalnych.

- src VARCHAR(32)
- dest VARCHAR(60)
- msg VARCHAR(255)

`--debug`
Opcja generuje informacje diagnostyczne.

`--dryrun`
Opcja interpretuje wszystkie, polecenia nie wykonując ich.

`--flushlog`
Wykonuje instrukcję FLUSH LOGS po zablokowaniu tabel.

`--help (-?)`
Zwraca informacje o sposobie użycia skryptu.

`--keepold`
Zachowuje kopie starszych archiwów (o ile takie istnieją)

`--method=nazwa`
Określa metodę kopiowania plików. Może to być `cp` lub `scp`.

`--noindices`
Wstrzymuje kopiowanie plików indeksów. Można ją cofnąć za pomocą myisamchk.

`--password=hasło (-phasło)`
Określa hasło używane przy łączeniu się z serwerem.

`--port=nazwa_portu (-P nazwa_portu)`
Określa port używany przy łączeniu się z serwerem.

`--quiet (-q)`
Wstrzymuje wszystkie komunikaty poza komunikatami błędów.

`--regexp=wzorzec`
Powoduje dopasowywanie nazw baz danych do podanego wzorca. Zarchiwizowane zostają wszystkie pasujące do wzorca bazy.

`--socket=/ścieżka/dostępu (-S /ścieżka/dostępu)` Określa ścieżkę do gniazda używanego podczas połączenia.

`--suffix=sufiks`
Dodaje sufiks do nazwy katalogu archiwum. Domyślnie sufiksem jest `_copy`.

`--tmpdir=ścieżka`
Określa katalog tymczasowy (inny niż `/tmp`).

`--user=nazwa_użytkownika (-u nazwa_użytkownika)`
Określa nazwę użytkownika

Możliwe jest porównywanie nazw tabel z wyrażeniem regularnym. Znak tyldy odwraca dopasowanie w ten sposób, że kopiowane są tylko tabele, które nie pasują do podanego wzorca. Ścieżkę określa katalog przeznaczony do archiwum.

`Mysqlhotcopy` ma jednak poważne ograniczenia – pracuje tylko na tabelach typu ISAM i MyISAM. Poza tym musi być uruchomiony na serwerze, a nie zdalnie. Kopie plików z danymi można wykorzystywać jedynie w MySQL.

Aby wywołać `mysqlhotcopy` musimy mieć

- dostęp do plików, które zawierają nasze kopiowane tabele
- przywileje SELECT dla tych tabel i przywilej RELOAD (aby było możliwe wykonanie instrukcji FLUSH TABLES).

2.3 Programy firmy Innobase Oy Inc.

Jak już wspomniałam w rozdziale pierwszym, tabele InnoDB zostały wymyślone przez firmę Innobase Oy Inc. Dostęp do tabel typu InnoDB zyskujemy razem z dostępem do bazy danych MySQL, a firma Innobase stworzyła programy "wspierające" ten rodzaj tabel. Poniżej omówię dwa z nich dotyczące kopiowania danych.

2.3.1 InnoDB Hot Backup

InnoDB Hot Backup (`ibbackup`) jest skryptem perlowym, który pozwala na zrobienie kopii zapasowej (migawki) działającej bazy danych MySQL zawierającej tabele InnoDB bez blokowania i przerywania zwykłej pracy bazy danych. W wyniku użycia tego narzędzia otrzymamy spójną kopię bazy danych. InnoDB Hot Backup jest też dobrą metodą na przygotowanie nowych serwerów podrzędnych, jeśli zamierzamy replikować tabele InnoDB.

Podstawowa komenda, aby wywołać ten program brzmi

```
ibbackup my.cnf my2.cnf
```

`Ibbackup` działa w ten sposób, że czyta z pliku `my.cnf` wszystkie informacje o tym, gdzie są `ibdata` i `ib_logfiles`[6], a potem tworzy ich kopię zgodnie z konfiguracją w pliku określonym wyżej jako `my2.cnf`.

Pliki `my.cnf` oraz `my2.cnf` muszą zawierać następujące parametry:

[6] nawiązać jakoś do listy plików z rozdziału pierwszego

```
datadir=...
innodb_data_home_dir=...
innodb_data_file_path=...
innodb_log_group_home_dir=...
innodb_log_files_in_group=...
innodb_log_file_size=...
```

Ścieżka do katalogu musi być koniecznie podana, gdyż `ibbackup` nie obsługuje opcji domyślnych. Warunek o liczbie plików z danymi (data files), a także o ich rozmiarach musi być zgodny pomiędzy `my.cnf` oraz `my2.cnf`. Jeśli ostatni plik z danymi jest określony jako `auto-extending` w `my.cnf`, wtedy musi on być także tak zadeklarowany również w `my2.cnf`. Liczba plików dziennikowych (log files) oraz ich rozmiar muszą być wyraźnie określone, ale nie jest wymagane, aby wartości określone w `my.cnf` oraz `my2.cnf` pasowały do siebie.

Autor podaje taki przykład na zmodyfikowanie pliku `my.cnf`:

```
[mysqld]
datadir = /home/heikki/data
innodb_data_home_dir = /home/heikki/data
innodb_data_file_path = ibdata1:10M:autoextend
innodb_log_group_home_dir = /home/heikki/data
set-variable = innodb_log_files_in_group=2
set-variable = innodb_log_file_size=20M
```

Załóżmy, ze chcemy zrobić kopię w katalogu `/home/heikki/backup`. Wtedy `my2.cnf` powinien wyglądać następująco:

```
datadir = /home/heikki/backup
innodb_data_home_dir = /home/heikki/backup
innodb_data_file_path = ibdata1:10M:autoextend
innodb_log_group_home_dir = /home/heikki/backup
set-variable = innodb_log_files_in_group=2
set-variable = innodb_log_file_size=20M
```

Opcje z jakimi występuje `ibbackup`:

`--apply-log`

`--include wzorzec`Jeżeli użyjemy tej opcji, zostaną skopiowane tylko te pliki, które pasują do podanego wzorca, który jest wyrażeniem regularnym. Dla każdej tabeli jej pełna nazwa, czyli nazwa_bazy_danych.nazwa_tabeli jest porównywana z wzorcem.

`--restore`
Opcja ta jest poprzedniczką instrukcji `--apply-log`. Autorzy uznali, że

nazwa `--restore` nie oddaje w pełni tego, co robi ta instrukcja, dlatego w wersji 2.0 zostało dodane `--apply-log`. Używanie `--restore` nie jest zalecane, gdyż ta opcja może być w przyszłości usunięta.

`--sleep ms`
Powoduje, że program "zasypia" (wstrzymuje kopiowanie) na `ms` milisekund po każdym megabajcie skopiowanych danych. `ms` musi być mniejsze niż 1000000. Domyślna wartość `ms` wynosi 0.

`--suspend-at-end`
Ta opcja pojawiła się w wersji 1.05. Możemy jej używać, jeśli chcemy zablokować i wykonać kopie rezerwowe dla tabel MyISAM. Dzięki tej instrukcji otrzymamy spójny obraz tabel InnoDB oraz MyISAM.
Instrukcja ta zmusza `ibbackup` do zachowywania się w następujący sposób: kiedy procedura kopiowania danych jest bliska końca, `ibbackup` tworzy plik o nazwie `ibbackup_suspended` w katalogu, który podaliśmy w `my.cnf` przy instrukcji `log_group_home_dir` i czeka, aż użytkownik usunie plik `ibbackup_suspended`.
Jeśli skrypt widzi plik `ibbackup_suspended`, wtedy wykonuje następujące instrukcje:

1. wykonuje instrukcję FLUSH TABLES WITH READ LOCK;
2. kopiuje wszystkie pliki o rozszerzeniu `.frm` i tabele MyISAM z katalogów objętych "backupowaniem";
3. wznawia pracę `ibbackup` przez usunięcie pliku `ibbackup_suspended`;
4. wykonuje polecenie UNLOCK TABLES;

`--use-memory mb`
Opcja ta jest istotna tylko wtedy, gdy instrukcja `--apply-log` jest zadeklarowana. Informuje ona program `ibbackup`, że może on użyć `mb` megabajtow pamięci. Domyślnie jest to 100 MB.

`--compress poziom`
Powoduje kompresję kopii plików z danymi. Nazwy skompresowanych plików są tworzone poprzez dodanie sufixu `.ibz` do nazwy pliku. Poziom kompresji może być przez nas podany jako argument za poleceniem. Poziom kompresji jest liczbą całkowitą pomiędzy 1 a 9: 1 daje najszybszą kompresję, 9 daje najlepszą kompresję, a 0 oznacza brak kompresji. Wartością domyślną jest 1.

`--uncompress` Opcja `uncompress` jest istotna tylko w przypadku, zawołania również `--apply-log`. Powoduje ona, że `ibbackup` odzyskuje dane ze skompresowanych kopii.

`Ibbackup` robi kopię tabel InnoDB oraz ich indeksów. Nie kopiuje jednak plików `*.frm`, tabel MyISAM i ich indeksów.

InnoDB Hot Backup w wersji 2.0 działa we wszystkich najpopularniejszych systemach operacyjnych (w tym w systemie Windows oraz Solaris). Niestety, nie jest on programem darmowym. Roczna licencja kosztuje 510$, a licencja wieczysta – 1300$

2.3.2 `innobackup`

`Innobackup` również jest skryptem perlowym. To proste w użyciu narzędzie służy do robienia kompletnych kopii tabel MyISAM oraz InnoDB. Jest używane w kombinacji z programem `ibbackup`, który zaczyna pracę jako proces potomny (child process) wywołany przez `innobackup`.

`Innobackup` będzie pracował prawidłowo tylko wtedy, gdy:

1. Podczas robienia kopii nie używamy długich instrukcji SELECT, czy też innych zapytań SQL.
2. Kopiowane tabele MyISAM nie są zbyt duże.

`Innobackup` nie "backupuje" tabel HEAP ani BDB.

Opcje:

`--help`
: Wyświetla informacje na temat programu

`--version`
: Wypisuje na ekran informacje o wersji programu.

`--apply-log`

`--copy-back`
: Kopiuje dane z katalogu, w którym są trzymane kopie z powrotem do początkowego położenia.

Opcje przekazywane procesowi potomnemu `ibbackup`. Zostały one omówione dokładniej w podrozdziale poprzednim.

`--use-memory=mb`

`--sleep=ms`

`--compress[=poziom]`

`--include=wzorzec`

`--uncompress`

Opcje przekazywane procesowi potomnemu `mysql`:

`--user=nazwa_użytkownika`
: Definiuje użytkownika używaną podczas logowania się do bazy danych, jeśli nie jest to ta sama nazwa użytkownika, która posłużyła do zalogowania się na danym komputerze.

`--password=hasło`
: Definiuje hasło, którego należy użyć, aby połączyć się z bazą danych.

`--port=port`
: Zmienia numer portu używany przy łączeniu się z lokalnym serwerem bazodanowym przez TCP/IP. Domyślnie jest nim 3306.

`--socket=ścieżka`
: Zmienia domyślną ścieżkę używaną przy łączeniu się z serwerem poprzez gniazdo sieciowe UNIX-a.

W przeciwieństwie do `ibbackup`, za użytkowanie `innobackup` nie trzeba płacić. Jego kod źródłowy możemy znaleźć na stronie `http://www.innodb.com/innobackup.txt`.

2.4 `mysqlsnapshot`

Skrypt napisany przez programistę Yahoo imieniem Jeremy Zawodny. Został on napisany, aby ułatwić przygotowywanie replikacji dla MySQL-a w wersji 3.23.xx. W tych wersjach przygotowanie replikacji było trudne, ponieważ potrzebujemy najpierw spójnego obrazu ("migawki") danych z serwera nadrzędnego na serwerze podrzędnym. Należy zadbać, aby uruchomienie dziennika binarnego i stworzenie tego obrazu pochodziły z tej samej chwili. `Mysqlsnapshot` automatyzuje ten proces. To narzędzie jest używane, aby stworzyć "migawkę" i zapoczątkować dziennik binarny dla replikacji na serwerze nadrzędnym bez wyłączania go. Dziennik binarny przed zawołaniem powinien być uaktywniony.

`Mysqlsnapshot` jest także alternatywą dla `mysqlhotcopy`, jeśli chcemy wykonać kopię pracującego serwera MySQL.

Program `mysqlsnapshot` był znany jako `myrepl`, lecz autor uznał, że nazwa `mysqlsnapshot` lepiej opisuje ten skrypt.

Aby użyć `mysqlsnapshot` należy najpierw go zainstalować, program nie jest bowiem udostępniany razem z MySQL. Można go pobrać ze strony `http://jeremy.zawodny.com/mysql/mysqlsnapshot`. Po instalacji możemy uruchomić to narzędzie z linii poleceń.

`Mysqlsnapshot` pracuje należycie wyłącznie na tabelach MyISAM.

2.5 Polecenie BACKUP TABLE

BACKUP TABLE jest dostępne w wersji MySQL 3.23.25 oraz późniejszych. W dokumentacji MySQL można przeczytać, że wykonywanie kopii zapasowej tym poleceniem jest rozwiązaniem przestarzałym. Programiści pracują właśnie nad nowymi rozwiązaniami i raczej nie zalecają korzystania z tej kwerendy. Na razie autorzy proponuja korzystanie z `mysqlhotcopy`. Niemniej jednak polecenie BACKUP TABLE działa prawidłowo i nie ma tutaj znaczenia, czy jest to metoda przestarzała, czy nie.

Instrukcji tej używamy w następujący sposób:

```
BACKUP TABLE nazwa_tabeli TO '/pełna/ścieżka/dostępu/';
```

Możemy też skopiować kilka tabeli jednym poleceniem:

```
BACKUP TABLE [Tabela1, Tabela2...] TO '/pełna/ścieżka/dostępu/';
```

BACKUP TABLE kopiuje do podanego katalogu najmniejszą ilość plików, potrzebną, aby odtworzyć tabele. Kwerenda ta pracuje tylko dla tabel MyISAM. Kopiuje ona definicję tabeli z pliku `*.frm` oraz dane z pliku `*.MYD`. Plik z indeksami `*.MYI` zostanie odbudowany na podstawie zachowanych dwóch plikow. Aby wykorzystać BACKUP TABLE, będziemy potrzebowali dostępu do MySQL z poziomu powłoki. Oczywiście użytkownik, ktory chce wykorzystać tę instrukcję musi miec prawo zapisu w podanym katalogu.

Przed procesem kopiowania danej tabeli zakładana jest na nią blokada odczytu. Po skopiowaniu plików, tabela jest odblokowywana. Blokady nie są zakładane jednocześnie na wszystkie tabele, ale tylko na tę, która aktualnie jest "backupowana". Jeśli chcemy zapobiec zmianom tabel podczas kopiowania, powinniśmy przed zawołaniem BACKUP TABLE zablokować tabele poleceniem LOCK TABLES.

BACKUP TABLE zwraca tabelę z następującymi kolumnami:

Kolumna	**Zawartość**
Table	Nazwa tabeli
Op	Zawsze "backup"
Msg_type	Jedno ze "status", "error", "info" albo "warning"
Msg_text	Wiadomość

Kopiowanie plików jest realizowane na poziomie systemu operacyjnego, wobec czego taka metoda robienia kopii zapasowych jest dość szybka.

W przypadku, gdy dochodzi do ewentualnej awarii bazy danych, MySQL umożliwia przywrócenie zawartości tabel w oparciu o ich kopię zapasową, wykonaną przy pomocy polecenia BACKUP. Służy do tego instrukcja RESTORE, która wgrywa dane ze stworzonych uprzednio plików i na podstawie definicji tabel, zawartych w plikach .frm, tworzy poszczególne indeksy. Polecenie RESTORE jest wykorzystywane w momencie, gdy automatyczna naprawa tabel nie przynosi skutku – przywraca bowiem ostatnio zapisaną wersję bazy danych.

2.6 Polecenie SELECT INTO OUTFILE

W MySQL-u możliwe jest również zapisywanie poszczególnych wierszy w ramach tabeli do osobnego pliku `*.txt`. Służy do tego instrukcja SELECT, połączona w odpowiedniej relacji z adresem pliku tekstowego. Jeżeli baza danych składa się tylko z kilku tabel i mamy dostęp do serwera MySQL z poziomu powłoki, polecenie to jest najprostszym sposobem wykonania kopii zapasowej danych. Dzięki tej metodzie możemy szybko zapisać tabelę lub jej część do pliku tymczasowego. Wykorzystując pełnię możliwości instrukcji SELECT możemy ograniczyć wybór wierszy, które mają znaleźć się w kopii zapasowej, aby zapisać całą tabelę, wpisujemy po prostu polecenie SELECT *.

Składnia polecenia SELECT INTO OUTFILE:

```
SELECT * FROM nazwa_tabeli
     INTO OUTFILE ‘/ścieżka/do/pliku/kopia.txt’;
```

Pamiętajmy, że plik wynikowy nie może już znajdować się na dysku. MySQL sprawdza, czy taki plik istnieje – jest to forma prostego zabezpieczenia się przed przypadkowym nadpisaniem istniejących kopii zapasowych. Ponadto konto użytkownika, z którego uruchomione zostanie polecenie SELECT INTO OUTFILE, musi posiadać uprawnienia do zmiany plików na serwerze MySQL. Największą zaletą tej metody jest niewiarygodna szybkość wykonywania kopii zapasowej pojedynczej tabeli.

System automatycznie rozdziela przy użyciu tabulatorów poszczególne komórki zapisane w ramach jednego wiersza, natomiast po zakończeniu każdego kolejnego wiersza przechodzi do następnej linii w dokumencie. Tak uporządkowane dane mogą później być łatwo wczytane przez odrębne programy informatyczne, jak również skopiowane do odpowiedniej tabeli w ramach bazy danych MySQL za pomocą polecenia LOAD.

```
LOAD DATA INFILE ‘/ścieżka/do/pliku/kopia.txt’
    INTO TABLE nazwa_tabeli;
```

W tej sytuacji następuje załadowanie zawartości pliku do tabeli o określonej nazwie. Jeżeli do wykonywania kopii zapasowych tabel chcemy wykorzystać tandem SELECT INTO OUTFILE i LOAD DATA INFILE, zanim wykonamy LOAD DATA powinniśmy upewnić się, czy tabela docelowa jest pusta. Możemy także określić parametr IGNORE – w ten sposób polecenie pominie linie zawierające podwójne klucze główne, zapobiegając w ten sposób powstawaniu błędów w przypadku, gdy tabela zawierałaby już te same dane. Wadą rozwiązania SELECT INTO OUTFILE i LOAD DATA INFILE jest to, że opcje te posiadają zbyt wiele ograniczeń, by znalazły zastosowanie przy robieniu profesjonalnego backupu. Możemy zapisywać i ładować wyłącznie całe tabele. Oznacza to konieczność modyfikowania skryptów wykonujących kopie zapasowe za każdym razem, gdy stworzymy nową tabelę. Jednakże, jeżeli chcemy wykonać-kopię pojedynczej tabeli, opisany sposób jest całkiem dobrym rozwiązaniem.

Rozdział 3

Replikacja

3.1 Czym jest replikacja

Replikacja bazy danych to stałe kopiowanie zmian owych danych na inny serwer (na ten sam, bądź inny komputer). Dzięki replikacji możemy mieć możliwie najświeższą kopię lub rezerwową bazę danych.

Serwer, z którego kopiujemy dane nazywamy serwerem nadrzędnym, bądź Master. Serwer na który kopiujemy dane to serwer podrzędny, inaczej Slave.

Replikacja sprawia, że dane na serwerach są niemal identyczne (pod warunkiem, że serwer Slave nie jest odłączony). Dlatego w razie utraty danych na serwerze nadrzędnym pozostają dane na serwerze podrzędnym i może on podjąć zadania serwera Master.

Nie jest wymagane, aby serwer podrzędny był stale włączony. Jeżeli wyłączymy go na kilka godzin, to po ponownym połączeniu nadrobi zaległości bez żadnej szkody. Nie należy jednak dopuścić, aby przerwy w łączności pomiędzy serwerami Master i Slave były zbyt długie, gdyż starsze zapisy zmian mogą zostać usunięte, jeśli Master nie będzie miał dość miejsca, by je trzymać.

MySQL posiada pewne mechanizmy, które pozwalają realizować automatyczną replikację w układzie nadrzędny-podrzędny. Jeden serwer nadrzędny rejestruje wszystkie zmiany i udostępnia je w formie dziennika serwerom podrzędnym. Serwer Slave może znajdować się na innym komputerze, niż Master (i taka replikacja ma praktyczne zastosowanie), ale możemy także replikować dane pomiędzy dwoma serwerami MySQL znajdującymi się na tej samej maszynie. To drugie rozwiązanie jest używane do testowania nowej wersji MySQL-a bez używania innego komputera.

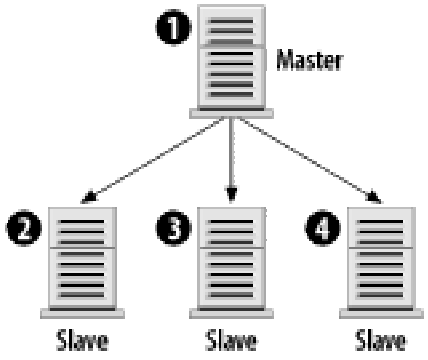
Reguły replikacji

Poniżej znajduje się kilka podstawowych reguł, których musimy się trzymać, aby replikacja działała prawidłowo:

- Każdy Slave musi mieć unikalny numer serwer ID.
- Serwer podrzędny może mieć tylko jeden serwer nadrzędny.
- Master może mieć wiele serwerów Slave.
- Serwery Slave mogą być serwerami nadrzędnymi dla innych serwerów.

3.2 Przykładowe konfiguracje

Master i wiele serwerów Slave



3.3 Jak przygotować replikację

Aby skonfigurować replikację w MySQL-u należy trzymać się kilku jasnych i prostych punktów:

1. Sprawdzamy, czy wersje MySQL-a zainstalowane na przyszłych serwerach Master i Slave nie wykluczają replikacji.
2. W pliku konfiguracyjnym serwera (w Unixie jest to `/etc/my.cnf`) dokonujemy zmian:
   - Na serwerze Master wpisujemy
     `[mysqld]`
     `server-id=`dowolna wartość od 1 do 2^{32} – numer serwera w sieci
     `log-bin` -włącza dziennik binarny

- Na serwerze SLAVE wpisujemy
  `[mysqld]`
  `server-id=`dowolna wartość od 1 do 2^{32}. Powinniśmy pamiętać, aby każdy serwer w systemie miał inny numer id

3. Przyznajemy uprawnienia (poleceniem GRANT) dla serwera Slave:

   - Uprawnienie do replikacji przyznajemy poleceniem:

     ```
     GRANT REPLICATION SLAVE ON *.*
       TO 'nazwa_serwera_slave'@'.domena.com'
       IDENTIFIED BY 'hasło';
     ```

     W wersjach MySQL starszych niż 4.0.2 nie ma uprawnienia REPLICATION SLAVE. Zamiast tego przyznajemy uprawnienie FILE. Robimy to poleceniem:

     ```
     GRANT FILE ON *.*
        TO 'nazwa_serwera_slave'@'.domena.com'
        IDENTIFIED BY 'hasło';
     ```

   - Uprawnienia potrzebne, do skorzystania z poleceń LOAD DATA FROM MASTER i LOAD TABLE FROM MASTER.[1]
     Jeśli planujemy skorzystać z któregoś z tych poleceń należy dodać przywileje SUPER i RELOAD globalnie, oraz przywilej SELECT do wszystkich tabel, które planujemy ściągnąć. Teoretycznie tabele bez przyznanego prawa SELECT dla serwera Slave nie powinny się ściągnąć w przypadku użycia LOAD DATA FROM MASTER.

4. Określamy Master na serwerze Slave. Możemy zrobić to na dwa sposoby:

   - ```
     CHANGE MASTER TO
           MASTER_HOST='nazwa_serwera_Master', (można też wpisać adres IP)
           MASTER_USER='nazwa_serwera_Slave',
           MASTER_PASSWORD='hasło'
           MASTER_LOG_FILE='nazwa_dziennika'
           MASTER_LOG_POS=pozycja początkowa w dzienniku serwera nadrzędnego
     ```

     Następnie wpisujemy komendę:

     ```
     START SLAVE;
     ```

     która powoduje, że Slave zaczyna śledzić zmiany powstałe na serwerze nadrzędnym i zapisywać je "u siebie".

   - Możemy także dopisać do `my.cnf` pod tym co zostało wpisane wcześniej:

[1] Polecenia te zostały omówionena następnej stronie
     ```

```
master-host = nazwa lub adres IP serwera nadrzędnego
master-user = nazwa_serwera_Slave
master-password = hasło
master-port=3306
replicate-do-db = replikowana_baza_danych1
replicate-do-db = replikowana_baza_danych2
```

5. Restartujemy oba serwery

Postępując według powyższych zasad, przygotujemy serwer Slave do śledzenia zmian na serwerze Master. Należy jeszcze przenieść na serwer podrzędny dane, które już znajdują się na serwerze nadrzędnym. Mamy do wyboru kilka programów i poleceń.

Programy i polecenia, które możemy wykorzystać do przeniesienia danych:

1. `mysqldump`
2. `mysqlsnapshot`
3. `ibbackup` i `innobackup`
4. polecenia LOAD DATA FROM MASTER i LOAD TABLE FROM MASTER

Poza poleceniami z punktu ostatniego wszystkie programy zostały szerzej omówione w rozdziale drugim.

Polecenia LOAD DATA FROM MASTER i LOAD TABLE FROM MASTER

Polecenie LOAD DATA FROM MASTER powoduje zrobienie "migawki" danych zawartych na serwerze Master, po czym przekopiowuje ją na serwer Slave. Na czas robienia obrazu tabel wymusza globalną blokadę odczytu na serwerze Master. Odświeża wartości MASTER_LOG_FILE i MASTER_LOG_POS, więc serwer podrzędny zaczyna replikowanie od prawidłowych pozycji podanych wyżej parametrów.

Niestety instrukcja ta działa tylko dla tabel MyISAM. Nie respektuje nawet najbliższego mu typu – ISAM. Jeśli wśród replikowanych tabel znajdą się tabele innych typów niż MyISAM, próba użycia tej kwerendy spowoduje wyświetlenie błędu[2].

[2]ERROR 1189 (08S01): Net error reading from master

Autorzy zapowiadają, że w przyszłości z polecenia LOAD DATA FROM MASTER będziemy mogli korzystać również dla tabel InnoDB. Innym zapowiadanym kierunkiem rozwoju ma być zastąpienie potrzeby globalnej blokady odczytu przez użycie tzw. non-blocking online backup.

Należy wspomnieć, że LOAD DATA FROM MASTER nie kopiuje żadnych tabel z bazy danych `mysql`. Ułatwia to sytuację w której mamy innych użytkowników i inne przywileje na serwerach nadrzędnym i podrzędnym.

Polecenie LOAD DATA FROM MASTER potrzebuje przywilejów RELOAD i SUPER, aby połączyć się z serwerem Master oraz przywileju SELECT dla wszystkich tabel serwera, które pragniemy załadować na serwer podrzędny. Wszystkie tabele, do których Slave nie ma takiego uprawnienia, zostaną (powinny zostać[3]) zignorowane przez LOAD DATA FROM MASTER. Dzieje się tak, gdyż Master ukrywa je przed użytkownikiem: Instrukcja LOAD DATA FROM MASTER wywołuje SHOW DATABASES, by wiedzieć, które z baz danych serwera Master ma załadować, ale SHOW DATABASES zwraca jedynie te bazy danych do których Slave ma przywilej SELECT. Po stronie serwera Slave, użytkownik, który może wywołać polecenie LOAD DATA FROM MASTER powinien posiadać prawo, aby stworzyć, aktualizować i usunąć kopiowane bazy danych i tabele.

Podobnym poleceniem jest LOAD TABLE FROM MASTER. Przekazuje ono kopię tabeli z serwera Master na serwer Slave. Instrukcja ta została dołączona do MySQL-a głównie po to, by naprawić błędy w tabeli nie kopiując wszystkich danych. Podobnie jak poprzednia kwerenda wymaga przywilejów RELOAD i SUPER, a także przywileju SELECT do tabeli, którą chcemy skopiować. Tak samo jak LOAD DATA FROM MASTER działa tylko dla tabel MyISAM.

LOAD TABLE FROM MASTER używamy w następujący sposób:

```
LOAD TABLE nazwa_tabeli FROM MASTER;
```

Doświadczenia z replikacją w laboratorium komputerowym

W laboratorium komputerowym replikacja została uruchomiona według następującego schematu:

Na serwerze Master:

1. W pliku `/etc/my.cnf` zostało wpisane:

   ```
   [mysqld]
   ```

[3]Nasze doświadczenia z labortatorium komputerowego wykazują, że polecenie LOAD DATA FROM MASTER działa poprawnie bez uprawniena SELECT.

```
log-bin=mysql-bin
server-id=1
```

2. W konsoli MySQL wpisano:

```
mysql> grant replication slave on *.*
    -> to repl@'lab-b04.im.uwb.edu.pl'
    -> identified by 'slavepass';

mysql> grant super, reload on *.*
    -> to repl@'lab-b04.im.uwb.edu.pl'
    -> identified by 'slavepass';

mysql> grant select on dbMatFiz.*
    -> to repl@'lab-b04.im.uwb.edu.pl'
    -> identified by 'slavepass';
```

3. W linii poleceń zostało wpisane:

```
lab-b03\$ /etc/init.d/mysqld stop
lab-b03\$ /etc/init.d/mysqld start
```

Na serwerze Slave:

1. Do pliku `/etc/my.cnf` zostało wpisane:

```
[mysqld]
server-id=4
```

2. W konsoli MySQL-a zostało wpisane:

```
mysql> Change master to
    -> master_host='lab-b03',
    -> master_user='repl',
    -> master_password='slavepass',
    -> master_log_file='mysql-bin',
    -> master_log_pos=0;
```

3. W linii poleceń wpisano:

```
lab-b04\$ /etc/init.d/mysqld stop
lab-b04\$ /etc/init.d/mysqld start
```

4. W linii poleceń MySQL-a wpisano:

```
mysql> START SLAVE;
mysql> LOAD DATA FROM MASTER;
```

Jak widać uruchomienie replikacji nie jest skomplikowane. Poza skonfigurowaniem replikacji wykonaliśmy szereg testów. Sprawdziliśmy, co się stanie, gdy usuniemy tabelę na serwerze Slave. W takiej sytuacji, kiedy w odpowiedniej tabeli na serwerze Master wykonywane są zmiany, to replikacja przestaje działać. Aby przywrócić działanie replikacji, należy ponownie przenieść całą replikowaną bazę z serwera Master. Inne operacje (dodawanie i usuwanie wierszy) nie powodowały zakłóceń w replikacji. Usunięcie tabeli na Master powoduje usunięcie odpowiedniej tabeli na Slave. Jedną z ważniejszych cech replikacji jest możliwość wyłączenia serwera Slave na pewien okres czasu bez szkody dla replikacji. Po włączeniu serwera Slave dane zostaną uzupełnione i replikacja dalej przebiega prawidłowo.

Nasze testy pokazują, że replikacja jest mechanizmem pozwalającym na tworzenie aktualnych i wiarygodnych kopii zapasowych.

3.4 Uwarunkowania techniczne

Kompatybilność pomiędzy wersjami

Nowe wersje MySQL przynoszą zmiany. Zmiany te mogą być na tyle duże, że tabele i bazy danych będą zachowywały się zupełnie inaczej. Jak to ma się do replikacji? Oczywiście najlepiej by było, aby serwery nadrzędny i podrzędny miały zainstalowaną tę samą wersję MySQL-a. Nie zawsze jednak jest to możliwe. Ogólnie przyjmuje się zasadę, że serwer Slave powinien mieć co najmniej tak nową wersję jak Master, jednak nie jest to absolutnie konieczne. Poniższa tabelka pokazuje, czy możliwa jest replikacja dla poszczególnych wersji MySQL zainstalowanych na serwerach Master i Slave:

		Master	Master	Master
		3.23.33 wzwyż	4.0.3 wzwyż lub którakolwiek z 4.1.x	5.0.3
Slave	3.23.33 i wyższe	+	—	—
Slave	4.0.3 i wyżej	+	+	—
Slave	5.0.3	+	+	+

Inne uwarunkowania

Nie ma gwarancji, że serwer Slave będzie idealnie zgodny z serwerem Master w danym momencie. Jeśli obciążenie serwera podrzędnego jest zbyt wysokie, mo-

gą wystąpić zaległości i Slave będzie potrzebował nieco czasu, aby je nadgonić.

W przypadku replikacji bardzo ważną sprawą jest przepustowość łącza. Jeśli Slave znajduje się daleko od Master (w sensie sieci) i nie ma zagwarantowanej odpowiedniej przepustowości, Serwer Slave jest w stanie nadążyć za serwerem Master w przetwarzaniu kwerend, lecz nie dostanie danych odpowiednio szybko. Takie opóźnienie także może stać się istotną kwestią.

Rozdział 4

Strategie tworzenia kopii zapasowych

Planując tworzenie kopii zapasowych danych zgromadzonych w bazie MySQL należy rozważyć kilka czynników. Do najważniejszych, na które należy zwrócić uwagę należą:

1. Wielkość przechowywanych danych
2. Jak często dokonywane są zmiany w bazie
3. Czy częściej mamy do czynienia z zapisem czy też z odczytem bazy danych
4. Czy jest w ogóle możliwe, aby na pewien czas wyłączyć bazę danych i jak długi może być to czas
5. Jak szybko byśmy chcieli mieć odtworzone dane po awarii
6. Wartość danych przechowywanych w bazie
7. Jak dużo pieniędzy możemy przeznaczyć na tworzenie backupu

Na pierwszy rzut oka powyższe zagadnienia mogą być nieoczywiste lub nie mieć ze sobą związku. W tym rozdziale chcemy wykazać ich zasadność i odpowiedzieć na pytanie: w jaki sposób tworzyć kopie zapasowe w określonych warunkach?

Zastanówmy się nad znaczeniem wyżej wymienionych warunków.

Szacując rozmiar danych, które mamy w bazie należy wziąć pod uwagę rozwój bazy i to jak szybko rozmiar naszej bazy rośnie. Planując backup musimy zapewnić co najmniej tyle miejsca na nośniku ile danych mamy w bazie. Przy dużych bazach danych kopiowanie całości bazy za każdym razem może zupełnie nie mieć sensu ze względu na wymagany rozmiar nośnika i potrzebny na

stworzenie kopii czas. W skrajnej sytuacji może dojść do tego nawet, że czas potrzebny na wykonanie pełnej kopii będzie dłuższy niż założona częstotliwość wykonywania kopii bezpieczeństwa. Widzimy tutaj bezpośredni związek trzech z wymienionych czynników, a mianowicie 1, 2 i 4. Wynika stąd, że przy dużych bazach danych, gdzie często dokonywane są zmiany należy pomyśleć o częściowych kopiach zapasowych, w których zapisane będą tylko zmiany dokonane od momentu ostatniej archiwizacji. Może być też tak, że nie zależy nam na całości bazy, gdy na przykład część danych jest łatwa do odtworzenia bez konieczności użycia kopii zapasowych, wówczas kopiujemy np. tylko wybrane tabele z bazy danych i potrzebujemy tylko tyle miejsca ile one zajmują. Zauważmy, że dotknęliśmy tutaj zagadnienia 6.

Od częstotliwości dokonywanych zmian powinna zależeć częstotliwość tworzenia kopii zapasowych. Im częściej zmieniają się dane, tym częściej trzeba je archiwizować. Niestety, w przypadku większości programów służących do robienia kopii zapasowych zrobienie backupu wiąże się z wyłączeniem bazy danych. Dlatego częstość robienia backupów jest wypadkową czynników 2 i 4.

Jeżeli mamy do czynienia z danymi o dużej wartości jak na przykład dane bankowe i nie możemy pozwolić sobie na zbyt duży przestój spowodowany awarią, będziemy prawdopodobnie pomyśleć o czymś innym niż klasyczne kopie zapasowe. Dobrym rozwiązaniem w takiej sytuacji jest zastosowanie replikacji na inną maszynę, w połączeniu z redundantnymi macierzami dyskowymi RAID, które zmniejszają ryzyko utraty danych w przypadku awarii twardego dysku. Im cenniejsze są dane, tym dalej od siebie powinny się znajdować serwery Master i Slave, bo w przypadku np. wystąpienia przepięcia elektrycznego komputery znajdujące się w tym samym pomieszczeniu mogą jednocześnie ulec uszkodzeniu. Dobrym pomysłem jest także replikowanie danych na kilka serwerów podrzędnych. Tutaj warto zwrócić uwagę, jak mają się do siebie punkty 6 i 7.

W praktyce podstawowym pytaniem, które trzeba sobie zadać planując backup jest: jak szybko chcemy przywrócić funkcjonalność systemu, który uległ awarii, czy też może wystarczy nam posiadanie aktualnej i wiarygodnej kopii zapasowej danych. Jeśli nie zależy nam na szybkim przywróceniu systemu do działania, to replikacja danych na inny serwer jest wystarczającym zabezpieczeniem. Replikacja wystarczy również wtedy, gdy udostępniane serwisy są rozdzielone pomiędzy różne serwery i awarii uległ serwer bazy danych. Poważny problem pojawia się tam, gdzie wiele serwisów jest zgromadzonych na jednej maszynie. Wyobraźmy sobie serwer na którym mamy jednocześnie bazę danych, serwer WWW, serwer pocztowy i serwer plików. Przywrócenie funkcjonalności po awarii takiej maszyny jest bardzo trudne. Jeżeli chcemy szybko przywrócić wszystkie serwisy to tak naprawdę musimy mieć je wszystkie zdublowane na innej maszynie, najlepiej o takich samych parametrach technicz-

nych, aby możliwe było łatwe przenoszenie nie tylko danych, ale i programów. W pewnym sensie musimy zasymulować replikację nie tylko bazy danych, ale również pozostałych serwisów. Jak widać koszt takiego zabezpieczenia jest wysoki, gdyż ceny sprzętu należy przemnożyć przez 2.

Koszty zabezpieczeń przed utratą danych rosną gwałtownie wraz z nakładaniem rygorystycznych wymagań co do tych zabezpieczeń. Krótko mówiąc prosty backup można wykonać bez wyraźnych nakładów finansowych, natomiast przy replikacji musimy już się liczyć ze zwiększeniem liczby serwerów, rozbudową łącz sieciowych i wynajęciem wykwalifikowanej obsługi.

Bibliografia

core [1] Atkinson Leon, *Core MySQL*, Helion, 2003.

manual [2] Podręcznik użytkownika MySQL, *MySQL Reference Manual*
`http://dev.mysql.com/doc/mysql/en/index.html`

profdesigners [3] Leszczyński Michał, *Niekomercyjne systemy bazodanowe – MySQL*
`http://www.profdesigners.pl/articles/mysql.pdf`

netmirror [4] Jeremy Zawodny i Derek Balling, *High Performance MySQL. Chapter 7: Replication*
`http://netmirror.org/mirror/mysql.com/books/hpmysql-excerpts/ch07.html`

przystan [5] Thomas Wölfer, *Bezpieczna przystań*
`http://www.linux-magazine.pl/issue/06/Sysadmin_mysql.pdf`

innodb [6] Podręcznik użytkownika InnoDB, *InnoDB Hot Backup Manual*
`http://www.innodb.com/manual.php`

mysqlsnapshot [7] Jeremy Zawodny, *mysqlsnapshot*
`http://jeremy.zawodny.com/mysql/mysqlsnapshot`

wikipedia [8] Encyklopedia, `http://pl.wikipedia.org/`