Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

Bazy Danych

Ćwiczenie 15: Import i eksport danych. Tworzenie kopii bezpieczeństwa oraz odzyskiwanie danych

opracował: dr hab. inż. Artur Gramacki (a.gramacki@issi.uz.zgora.pl)

1. Uwagi wstępne

Na niebiesko zaznaczono polecenia (tu oraz w dalszej części instrukcji) wpisywane przez studenta w konsoli tekstowej. Symbol `shell>` zawsze będzie oznaczać znak zachęty w konsoli tekstowej systemu Windows a symbol `mysql>` znak zachęty w konsoli tekstowej MySQL-a.

Serwer MySQL, jak każde oprogramowanie, nie jest całkowicie bezbłędne i odporne na awarie. Gdy przechowywane dane są ważne (a po cóż byłoby przechowywać dane nieważne!) należy zadbać o odpowiednią ich ochronę. Chodzi o stworzenie takich warunków, aby nawet w przypadku wystąpienia bardzo poważnej awarii serwera MySQL (nieważne przez kogo zawinionej – przez człowieka, przez wadliwie działające oprogramowanie lub sprzęt komputerowy) zgromadzone dane były bezpieczne lub łatwe do odzyskania. W niniejszym ćwiczeniu poznasz jakimi metodami i jakimi narzędziami można zapewnić bezpieczeństwo naszym danym.

Często zachodzi również potrzeba wyeksportowania zgromadzonych w bazie MySQL danych „na zewnątrz", celem przykładowo wykorzystania ich w innej aplikacji. Mówimy wówczas o *eksporcie danych*. Z kolei odwrotna sytuacja ma miejsce wówczas, gdy chcemy do bazy MySQL załadować dane gromadzone w innych aplikacjach. Mamy wówczas do czynienia z *importem danych*. MySQL bardzo dobrze radzi sobie w obu przypadkach.

Pliki z wyeksportowanymi danymi mogą być uważane za naturalne źródło kopii bezpieczeństwa. W razie wystąpienia awarii serwera MySQL można łatwo z nich skorzystać, aby odzyskać utracone dane. Nawet jeżeli nie uda nam się odzyskać wszystkich utraconych danych, to niemal zawsze, posiadając kopię bezpieczeństwa, jesteśmy w stanie odtworzyć ich znakomitą większość.

```
SELECT
  id, first_name, last_name, commission_pct, salary
INTO OUTFILE
  'c:/temp/plik1.txt'
  FIELDS
    TERMINATED BY '\t'
    ESCAPED BY ''
    OPTIONALLY ENCLOSED BY '"'
  LINES
```

```
    TERMINATED BY '\r\n'
FROM
  emp
WHERE
  salary > 1500;
```

Plik wynikowy ma następującą postać:

```
1   "Carmen"    "Velasquez" NULL    2500.00
5   "Audry" "Ropeburn"  NULL    1550.00
13  "Yasmin"    "Sedeghi"   10.00   1515.00
14  "Mai"   "Nguyen"    15.00   1525.00
```

Uwagi:

- aby móc eksportować zawartość tabel w opisany wyżej sposób wymagane jest posiadanie przywileju `FILE`. Każdy użytkownik, który posiada ten przywilej może zapisywać pliki w dowolnym miejscu na dysku, do którego ma dostęp demon *mysqld*!. Z powodów bezpieczeństwa nie należy przyznawać przywileju `FILE` użytkownikom innym niż administratorzy! Aby nieco zwiększyć bezpieczeństwo, pliki generowane za pomocą instrukcji `SELECT ... INTO OUTFILE` nigdy nie nadpisują istniejących plików. Gdy nie posiadamy wymaganych uprawnień otrzymamy komunikat: `ERROR 1045 (28000): Access denied for user 'summit2'@'localhost' (using password: YES)`,
- wpisując ścieżkę dostępu do pliku stosujemy ukośniki w stylu systemów UNIX-owych (*slash* a nie *backslash*),
- plik, który wymieniamy w klauzuli `INTO OUTFILE` nie może już istnieć na dysku. Gdy plik o podanej nazwie już istnieje, MySQL nie nadpisze jego zawartości, tylko wypisze komunikat ostrzegawczy: `ERROR 1086 (HY000): File 'c:/temp/plik1.txt' already exists`. Jest to spowodowane względami bezpieczeństwa, aby nawet omyłkowo nie nadpisać jakiegoś ważnego pliku,
- opcje występujące w klauzulach `FIELDS` oraz `LINES` pozwalają wpływać na postać pliku wynikowego. Korzystając z dokumentacji postaraj się samodzielnie „rozszyfrować” poszczególne komendy.

2. Program *mysqldump*

Program narzędziowy o nazwie *mysqldump* służy do tworzenia zrzutu bazy danych lub kolekcji baz danych jako kopii bezpieczeństwa lub też w celu przeniesienia danych na inny serwer (niekoniecznie MySQL) Oczywiście ten inny serwer musi „umieć” odczytać pliki przygotowane przez *mysqldump*). Zrzut zawiera instrukcje SQL tworzące tabele i wypełniające je danymi. Istnieją trzy podstawowe sposoby uruchomienia programu. Szczegóły powinieneś samodzielnie doczytać w dokumentacji:

```
mysqldump [options] db_name [tbl_name ...]
mysqldump [options] --databases db_name ...
mysqldump [options] --all-databases
```

Załóżmy, że naszym celem jest wykonanie kopii tabeli *emp* oraz *dept* znajdujących się w bazie danych o nazwie *lab* (chodzi o demonstracyjną strukturę relacyjną. Patrz wcześniejsze instrukcje). Chcemy, aby w pliku wynikowym pojawiły się zarówno polecenia tworzące struktury tabel jak i dane zapisane w tych tabelach. Poniżej pokazano w jaki sposób to wykonać. W wywołaniu programu użyto jawnie wiele opcji. Część z nich występuje „nadmiarowo", gdyż są one przyjmowane domyślnie (sprawdź w dokumentacji które). Samodzielnie wykonaj różne eksperymenty, dodając lub usuwając różne opcje, i obserwuj plik wynikowy.

Uwaga: niektóre polecenia (jak to poniżej) są bardzo długie. Wpisujemy je w jednej linii, ew. możemy zapisać je do pliku i wykonywać jako polecenia wsadowe systemu operacyjnego (często takie pliki mają rozszerzenie *bat*).

```
shell> mysqldump --databases lab --tables emp dept --skip-quote-names --
comments=1 --complete-insert --create-options --add-drop-table --skip-opt --
user=lab --password=lab --result-file=c:\temp\emp-dept.sql
```

Wykorzystując odpowiednie opcje (jakie?) możemy za pomocą programu *mysqldump* skopiować do pliku tekstowego tylko strukturę bazy danych bez zawartości tabel. Opcja ta przydaje się wówczas, gdy chcemy np. przenieść bazę na inny komputer.

Gdy zamiast opcji `--databases` podamy `--all-databases`, spowoduje to utworzenie kopii zapasowych wszystkich przechowywanych baz. Musimy oczywiście podać nazwę i hasło użytkownika, który ma wystarczające uprawnienia do wyeksportowania wszystkich baz danych (poniżej logujemy się na konto administratora).

```
shell> mysqldump --all-databases --opt --no-data --user=root --password=root >
c:\temp\alldbdump.sql
```

Korzystając z opcji `--tab` można wygenerować plik z danymi rozdzielonymi znakami tabulacji. Dla każdej zrzuconej tabeli, program *mysqldump* tworzy plik `nazwa_tabeli.sql`, który zawiera instrukcję `CREATE TABLE` tworzącą tabelę, oraz plik `nazwa_tabeli.txt`, zawierający dane. Domyślnie pliki danych *.txt* są formatowane ze znakami tabulacji między wartościami kolumn oraz znakiem nowej linii na zakończenie rekordu. Można to jednak zmienić używając opcji `--field-xxx` oraz `lines-xxx`. Jak łatwo zauważyć działanie opcji `--tab` jest zbliżone do funkcjonalności oferowanej przez polecenie `SELECT ... INTO OUTFILE` omówione wcześniej.

Uwaga: opcja `--tab` powinna być używana tylko wtedy, gdy *mysqldump* jest uruchamiany na tej samej maszynie co serwer *mysqld*. Aby można używać tej opcji musimy ponadto posiadać `FILE` a serwer MySQL musi mieć prawa do zapisu plików w podanym katalogu.

```
shell> mysqldump lab --tab=c:\temp\lab --tables emp --user=root --password=root
--skip-quote-names --fields-terminated-by=\t --fields-escaped-by= --lines-
terminated-by=\r\n
```

Zwróćmy uwagę, że tym razem musieliśmy zalogować się na konto administratora.

3. Import. Program *mysqlimport*

W poprzednim podrozdziale poznaliśmy metody eksportowania danych do plików tekstowych. Pora więc zrobić z nich użytek. Pokażemy w jaki sposób zawartość tych plików załadować z powrotem do bazy MySQL. Będziemy to robili za pomocą programu narzędziowego *mysqlimport*. Funkcjonalność tego programu odpowiada w zasadzie instrukcji SQL `LOAD DATA INFILE`.

Dla każdego pliku tekstowego wymienionego w wierszu poleceń *mysqlimport* obcina rozszerzenie z nazwy pliku i wykorzystuje rezultat w celu określenia nazwy tabeli, do której zaimportować zawartość pliku. Załóżmy, że mamy następującą tabelę, do której chcemy wstawić dane:

```
CREATE TABLE pracownicy
(
   prac_id     INT             PRIMARY KEY,
   imie        VARCHAR(20)     NOT NULL,
   nazwisko    VARCHAR(30)     NOT NULL,
   zarobki     NUMERIC(11,2)       NULL,
   data_ur     DATE                NULL
);
```

Załóżmy następnie, że w pliku tekstowym o nazwie *pracownicy.txt* mamy następujące dane zapisane w pliku zewnętrznym:

```
1,'Artur','Nowakowski',NULL,'1980-01-26'
2,'Jan','Kowalski',2821.30,NULL
3,'Ewa','Konieczna',NULL,NULL
4,'Anna','Wojtasik',NULL,NULL
```

Po wykonaniu następującego polecenia (zwróćmy uwagę, że musimy zalogować się na konto z uprawnieniami administratora. Czy potrafisz wytłumaczyć dlaczego?):

```
shell>mysqlimport --user=root --password=root --fields-terminated-by=, --fields-
enclosed-by=' --lines-terminated-by=\r\n --delete lab c:\temp\lab\pracownicy.txt

lab.pracownicy: Records: 4  Deleted: 0  Skipped: 0  Warnings: 0
```

dane zostają prawidłowo załadowane do tabeli *pracownicy*, co potwierdzamy wykonując polecenie `SELECT`:

```
mysql> SELECT * FROM pracownicy;
+---------+-------+------------+---------+------------+
| prac_id | imie  | nazwisko   | zarobki | data_ur    |
+---------+-------+------------+---------+------------+
|       1 | Artur | Nowakowski |    NULL | 1980-01-26 |
|       2 | Jan   | Kowalski   | 2821.30 | NULL       |
|       3 | Ewa   | Konieczna  |    NULL | NULL       |
|       4 | Anna  | Wojtasik   |    NULL | NULL       |
+---------+-------+------------+---------+------------+
4 rows in set (0.00 sec)
```

4. Import. Wykorzystanie plików programu *mysqldump*

Mając plik eksportu możemy w bardzo łatwy sposób zapisane tam dane załadować do MySQL-a. Plik możemy wczytać albo w tzw. trybie wsadowym albo będąc już zalogowanym do MySQL-a wykonać polecenie `source`. Obie wersje pokazano poniżej:

```
shell> mysql -u lab -plab lab < c:\temp\lab\lab.sql
```

oraz

```
mysql> source c:\temp\blab.sql
```

Uwaga: Niezależnie od tego, którą z metod tworzenia kopii bezpieczeństwa wybierzemy, **bardzo ważne jest, aby sprawdzić poprawność utworzonej kopię**. Zawsze bowiem istnieje prawdopodobieństwo, że popełniliśmy jakiś błąd i możemy być bardzo niemile zaskoczeni, gdy odtworzenie bazy (np. po nagłej awarii) po prostu się nie uda. Wówczas będzie oczywiście za późno na wszelkie działania naprawcze. Inna sprawa, to właściwe przechowywanie plików z kopiami bazy. Oczywiście przechowywanie ich na tym samym dysku, co pliki serwera MySQL nie ma absolutnie sensu (dlaczego?).

5. Dziennik binarny. Uwagi wstępne

Opisane w poprzednich rozdziałach metody tworzenia kopii bezpieczeństwa mają charakter statyczny. Obraz bazy danych tworzony jest bowiem w określonym punkcie czasowym i plik eksportu zarejestruje stan, w jakim była baza w tym właśnie momencie. Wszystkie zmiany, które później zajdą w bazie nie zostaną już zapisane. Oczywiście, gdy wystąpi nagła awaria bazy, możemy odtworzyć jej stan tylko na chwilę, gdy była wykonywana kopia bezpieczeństwa.

Przykładowo wykonując kopię bazy co 24 godziny musimy liczyć się z tym, że w wariancie najbardziej pesymistycznym (gdy awaria pojawi się na kilka chwil przed wykonaniem kolejnej kopii) utracimy wszystkie dane z ostatnich 24 godzin. Teoretycznie kopie bezpieczeństwa można wykonywać częściej niż raz na dobę, ale zwykle jest to bardzo niewygodne. Często zdarza się, że ilość danych jest tak duża, że czas sporządzania kopii bezpieczeństwa liczony bywa w godzinach.

Opisane powyżej problemy są na tyle istotne, że każdy porządny serwer baz danych (a takim niewątpliwie jest MySQL) powinien sobie umieć z tym poradzić. W MySQL-u rozwiązano to w bardzo prosty sposób. Serwer można mianowicie uruchomić w trybie z rejestracją tzw. *dziennika binarnego.* W dzienniku tym zapisywane są *wszystkie* polecenia, które zmodyfikowały dane lub też próbowały je zmodyfikować (np. polecenie `DELETE`, w którym żaden rekord nie spełniał warunku `WHERE`). Polecenia, które nie modyfikują danych (np. `SELECT`) nie są rejestrowane w dzienniku binarnym.

Podstawowym zadaniem dziennika binarnego jest zapewnienie możliwości jak najpełniejszego uaktualnienia danych podczas operacji odtwarzania bazy (np. po awarii). Zapisane są tam bowiem wszystkie działania wykonywane na bazie od momentu wykonania ostatniej kopii bezpieczeństwa. Aby więc odtworzyć stan bazy w jakim była na chwilę przed awarią potrzebujemy zarówno ostatnią pełną kopię bazy jak również plik (lub pliki) dziennika binarnego.

Pliki te możemy więc traktować jako tzw. *kopie przyrostowe* i są one zwykle wielokrotnie mniejsze niż wielkość pliku z pełną kopią bazy.

Aby serwer MySQL utworzył dziennik binarny należy uruchomić go z opcją `--log-bin` lub też wpisać parametr `log-bin` do pliku z opcjami (domyślnie, w systemie Windows, jest to plik o nazwie *my.ini*, który znajduje się w katalogu, gdzie zainstalowano serwer MySQL).

6. Dziennik binarny. Przykład działania

W tym miejscu pokażemy w jaki sposób wykonać odtworzenie stanu bazy sprzed awarii wykorzystując pliki eksportu oraz pliki dziennika binarnego. Zakładamy więc następujący scenariusz:

- Upewniamy się, że serwer MySQL pracuje w trybie z rejestracją dziennika binarnego.

```
mysql> SHOW VARIABLES LIKE 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
```

- Na początku w bazie *lab* istnieje tylko jedna testowa tabela z danymi (w zasadzie obojętnie jaka i obojętnie z jakimi danymi).

```
mysql> DROP TABLE IF EXISTS test;
mysql> CREATE TABLE test (id INT) engine=InnoDB;
mysql> INSERT INTO test VALUES (1);

mysql> SHOW TABLES;
+----------------+
| Tables_in_blab |
+----------------+
| test           |
+----------------+

mysql> SELECT * FROM test;
+------+
| id   |
+------+
|    1 |
+------+
```

- Wykonujemy kopię bezpieczeństwa bazy za pomocą programu *mysqldump*.

```
shell> mysqldump lab --opt -u root -proot --single-transaction --flush-logs --
result-file=c:\temp\lab\lab-dump.sql
```

Uwagi:

- opcja `--single-transaction` gwarantuje, że dane odczytane w trakcie działania programu będą spójne (tzn., że jeżeli w trakcie wykonywania kopii ktoś inny będzie

próbował zmieniać kopiowane dane, nie będzie mógł tego zrobić zanim program *mysqldump* nie ukończy swojej pracy),

- opcja `--flush-logs` powoduje, że w momencie tworzenia kopii następuje wymuszenie zamknięcia oraz ponownego otwarcia plików dziennika binarnego. Bieżący plik dziennika binarnego zostaje zamknięty (staje się on plikiem archiwalnym. MySQL nigdy już do niego nic więcej nie zapisze), natomiast serwer MySQL inicjuje nowy plik dziennika binarnego. Nowy plik otrzymuje tą samą nazwę co poprzedni, z tym że zmienia się jego rozszerzenie (generowany jest kolejny numer. Gdy np. było: `xxx.000007`, to teraz będzie: `xxx.000008` itd.),
- dzięki użyciu opcji `--flush-log` mamy gwarancję, że w bieżącym pliku dziennika binarnego zapisywane są wszystkie zmiany od chwili utworzenia kopii bezpieczeństwa za pomocą programu *mysqldump*. Plik ten możemy więc traktować jako tzw. *kopię przyrostową*,
- aby móc użyć opcji `--flush-log` musimy posiadać uprawnienia `RELOAD`, a takowe posiadać powinien tylko administrator. Logujemy się więc na konto *root*.

- Po wykonaniu kopii, modyfikujemy w dowolny sposób dane w tabeli.

```
mysql> UPDATE test SET id = 2;
mysql> SELECT * FROM test;
+------+
| id   |
+------+
|    2 |
+------+

mysql> UPDATE test SET id = 3;
mysql> SELECT * FROM test;
+------+
| id   |
+------+
|    3 |
+------+
```

- Załóżmy, że w tym momencie nastąpiła awaria serwera MySQL. Gdy uporaliśmy się z awarią i serwer MySQL „podniósł się”, stwierdziliśmy, iż dane z tabeli *test* zniknęły. Pozostaje nam więc tylko próba odzyskania tych utraconych danych wykorzystując posiadane kopie bezpieczeństwa.

  Wymuszamy ,,przeładowanie'' dziennika binarnego. Skutkuje to tym, że bieżący plik dziennika binarnego zostaje zamknięty (można powiedzieć: zarchiwizowany) i system otwiera nowy plik. Możemy mieć pewność, że w zarchiwizowanym właśnie pliku na pewno znajduje się informacja o wykonanych przed chwilą poleceniach `UPDATE`. Czy wiesz jak poznać, który plik został zarchiwizowany? Aby móc wykonać poniższe polecenie musimy posiadać uprawnienia `RELOAD`, a takowe posiadać powinien tylko administrator.

```
Shell> FLUSH LOGS;
```

- Odtwarzamy stan bazy korzystając z kopii utworzonej programem *mysqldump*. Stwierdzamy, że zmiany dokonane w poprzednich punktach (polecenia `UPDATE`) nie zostały odtworzone (bo oczywiście dane zostały zmienione już *po* wykonaniu kopii bezpieczeństwa).

```
mysql> SOURCE c:\temp\lab\lab-dump.sql
mysql> SELECT * FROM test;
+------+
| id   |
+------+
|    1 |
+------+
```

- Odtwarzamy „brakujące” dane korzystając z informacji zapisanych w dzienniku binarnym. Uwaga: jako parametr dla programy *mysqlbinlog* musimy podać właściwy plik dziennika binarnego.

  Gdy serwer MySQL pracuje dłuższy czas z włączoną opcją rejestrowania dziennika binarnego na dysku powstaje sporo takich plików. Każdy z nich przechowuje zmiany z pewnego przedziału czasu w przeszłości. Bieżącym plikiem jest zwykle ten z najwyższym numerem w rozszerzeniu.

  Uwaga: zanim wykonasz poniższe polecenie upewnij się, który plik (lub pliki) dziennika należy podać jako parametr. Na każdym komputerze w laboratorium będzie to najprawdopodobniej inny plik!

```
shell> mysqlbinlog mysql-bin.000004 | mysql -u root -proot lab
mysql> SELECT * FROM test;
+------+
| id   |
+------+
|    3 |
+------+
```

Stwierdzamy, że zawartość tabeli wróciła do stanu z przed awarii.

- Na koniec oglądnijmy zawartość jednego przykładowego pliku dziennika. W zasadzie jedyną sensowną metodą ich oglądania jest użycie programu *mysqlbinlog*, który wyświetla zawartość liku w czytelnej dla człowieka postaci (czytelnej to nie znaczy, że łatwej!). Używamy przełącznika `--short-form`, aby wyświetlane były tylko najistotniejsze fragmenty. Analizując nawet pobieżnie otrzymany wynik, łatwo dostrzeżemy wydawane wcześniej polecenia `UPDATE`.

```
shell> mysqlbinlog mysql-bin.000004  --short-form
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
SET TIMESTAMP=1464084884/*!*/;
SET @@session.pseudo_thread_id=999999999/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0,
@@session.unique_checks=1, @@sessi
on.autocommit=1/*!*/;
```

```
SET @@session.sql_mode=1073741824/*!*/;
SET @@session.auto_increment_increment=1,
@@session.auto_increment_offset=1/*!*/;
/*!\C cp852 *//*!*/;
SET
@@session.character_set_client=40,@@session.collation_connection=40,@@session.co
llation_server=8
/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
BEGIN
/*!*/;
use `lab`/*!*/;
SET TIMESTAMP=1464084884/*!*/;
UPDATE test SET id = 2
/*!*/;
COMMIT/*!*/;
SET TIMESTAMP=1464084886/*!*/;
BEGIN
/*!*/;
SET TIMESTAMP=1464084886/*!*/;
UPDATE test SET id = 3
/*!*/;
COMMIT/*!*/;
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```

7. Zadania do samodzielnego wykonania

- załaduj do MySQL-a demonstracyjną strukturę relacyjną wraz z danymi (patrz poprzednie instrukcje,
- wykonaj eksport struktury bazy do pliku tekstowego (tylko struktury. Dane zapisane w tabelach nie powinny się w tym pliku znaleźć),
- wykonaj eksport zawartości poszczególnych tabel tworzących demonstracyjną strukturę relacyjną. Dane z wszystkich tabel powinny znaleźć się w jednym pliku. Napisz dwie wersje ćwiczenia: jedna powinna zawierać polecenia `INSERT` w tradycyjnej postaci (jedno polecenie `INSERT` wstawia tylko jeden rekord), druga natomiast w postaci bardziej skondensowanej (w jednym poleceniu `INSERT` wstawiamy wiele rekordów),
- utwórz nową bazę danych oraz nowego użytkownika z odpowiednimi uprawnieniami (jeżeli nie pamiętasz jak to zrobić, sięgnij do wcześniejszych instrukcji). Następnie korzystając z plików utworzonych w dwóch poprzednich punktach, odtwórz zawartość demonstracyjnej struktury relacyjnej w nowo utworzonej bazie danych,
- wykonaj ćwiczenie z podpunktu „*Import. Program mysqlimport*”, ale tym razem zamiast programu *mysqlimport* używaj polecenie SQL `LOAD DATA INFILE`,

- w pisanych poleceniach celowo wprowadzaj różne błędy i obserwuj jakimi komunikatami odpowiada serwer MySQL. Przykładowo w parametrach wywołania polecenia *mysqlimport* wpisz nazwę użytkownika, który nie posiada uprawnień administratora (a dokładniej chodzi o przywilej `FILE`),
- zaproponuje eksperyment podobny do tego, który opisano w rozdziale „*Dziennik binarny. Przykład działania*”, jednak przeprowadź go tak, aby zademonstrować możliwość odtwarzania stanu bazy do określonego punktu w przeszłości (ang. *Point-In-Time Recovery*). Brakujące informacje samodzielnie doczytaj w dokumentacji,
- korzystając z plików eksportu spróbuj zaimportować dane z bazy MySQL do arkusza kalkulacyjnego Excel.