HP OpenSource ブループリント

MySQL Server 5.0 管理監視ガイド

ver 1.0

invent

はじめに

本書は、オープンソースのデータベースである MySQL 5.0 データベースの運用監視時に必要とされる情報およびその情報の収集方法について説明します。

本書の範囲

本書では、MySQL 5.0 データベースを運用監視するための情報収集方法について説明します。MySQL 5.0 データベースのインストール方法等はマニュアルまたはその他資料を参照ください。
本書で説明する MySQL データベースのバージョンは 5.0.18 以降の MySQL 5.0 Community Edition および MySQL 5.0 Enterprise Server を対象としています。

本書の構成

目次

第 1 章　ログ収集
MySQL 5.0 データベースの各種ログファイルの説明および取得方法について説明します。

第 2 章　ステータス監視
動作中の MySQL 5.0 データベースおよびクライアントのステータスを監視方法について説明します。

第 3 章　ヘルスチェック
稼働中の MySQL 5.0　データベースに対するヘルスチェック方法について説明します。

付録

目次

1. ログ収集

MySQL 5.0 データベースでは、その動作およびデータベース更新情報等を記録するために「図 1 MySQL データベースのログファイル」に示すログファイルを出力します。

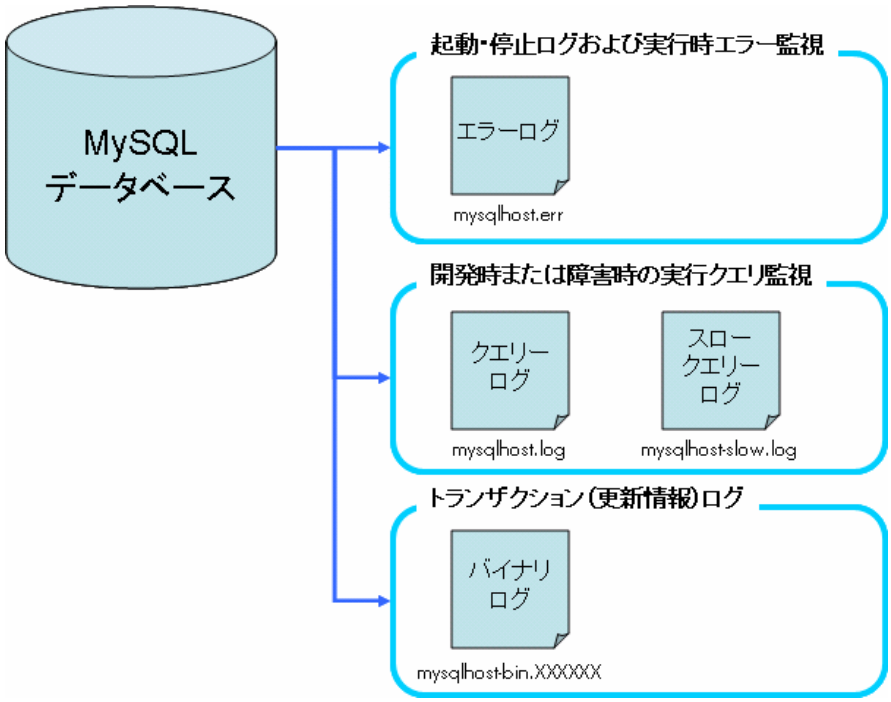

図 1 MySQL データベースのログファイル

各ログファイルの概要説明を「表 1MySQLデータベースのログファイル一覧」に示します。

表 1MySQL データベースのログファイル一覧

ログファイル	概要説明	
エラーログ	MySQL データベース・デーモン(mysqld) の起動・停止・実行の各時点で発生した問題を記録しています。	
	デフォルトでの出力	○
	ファイル名	$datadir/ホスト名.err
クエリーログ	クライアントとの接続確立およびクライアントから受信したすべての SQL ステートメントを記録しています。	
	デフォルトでの出力	×
	ファイル名	$datadir/ホスト名.log
スロークエリーログ	クエリ実行に long_query_time 設定値で指定した秒数以上の処理時間を必要としたすべてのクエリ、または INDEX を利用しなかったクエリを記録しています。	
	デフォルトでの出力	×
	ファイル名	$datadir/ホスト名-slow.log
バイナリログ	MySQL データベースに対するすべての更新情報を記録しています。	
	デフォルトでの出力	×
	ファイル名	$datadir/ホスト名-bin.XXXXXX (XXXXX は 6 桁のシーケンス番号)

注) $datadir は MySQL データベースのデータディレクトリです。MySQL 設定ファイルの datadir 設定で指定されます。

以下の節では、各ログファイルの以下の項目について説明します。

1. 概要
   ログファイルが出力する情報の概要を説明します。
2. 設定方法
   2.1. 設定オプション
   対象ログファイルのファイル名や振る舞いを設定するための設定オプションを説明します。

2.2. コマンド･オプション設定例
MySQL データベースの起動オプションとして設定する場合の設定例を示します。
MySQL 設定ファイルとして「Appendix 1: MySQL 設定ファイル･サンプル」に示す設定を利用しています。
2.3. MySQL 設定ファイル例
MySQL 設定ファイル(my.cnf / my.ini)ファイルで設定する場合の設定例を示します。
3. 詳細説明
ログファイルの出力サンプルや参照方法などを説明します。

1.1. エラーログ

1.1.1. 概要

エラーログファイルは、MySQL データベース･デーモンである mysqld の起動･停止時の情報、および実行時に発生した重大なエラーを記録するログファイルです。
MySQL データベース起動時のトラブルおよび実行中での障害発生時にこのログを確認することで、障害原因特定のための情報を得ることができます。

1.1.2. 設定方法

1.1.2.1. 設定オプション

- **--log-error[=filename]**
  エラーログの出力先ファイル名を指定します。ファイル名指定に相対パスを利用した場合、MySQL データベースのデータディレクトリからの相対パスで示されるディレクトリに、指定ファイル名で出力されます。ファイル名を指定しない場合、**ホスト名.err** で出力されます。
  オプションを指定しない場合でも MySQL データベースのデータディレクトリに出力されます。ただし Microsoft Windows で **--console** オプションを指定した場合は出力されません。
- **--log-warnings[=level], -W [level]**
  MySQL レプリケーション構成時の予期しないコネクション断などの警告メッセージをエラーログに出力するか否かを設定します。level に **0** を指定した場合、警告メッセージは記録されません。level に **1** 以上を指定した場合、警告メッセージが記録されます。デフォルトは **1** で、level を指定しなかった場合も **1** とみなされます。
- **--console**
  Microsoft Windows のみ有効です。
  このオプションを指定した場合、**mysqld** を実行したコンソールの標準エラー出力にログが出力されます。

1.1.2.2. コマンド・オプション設定例

エラーログの出力設定例を以下に示します。以下の例ではエラーログファイル名を mysql-error.log とし、警告メッセージの出力を指定しています。その他の設定はすべて MySQL 設定ファイル(my.cnf または my.ini)に設定されています。

UNIX および Linux の場合

```
$ mysqld_safe --log-error=mysql-error.log --log-warnings=1
```

Windows の場合

```
C¥> mysqld-nt.exe --log-error=mysql-error.log --log-warnings=1
```

上記により、MySQL 設定ファイルで指定された MySQL データディレクトリに mysql-error.log というファイル名のエラーログが出力されることになります。

1.1.2.3. MySQL 設定ファイル例

MySQL 設定ファイルでは、 mysqld セクションに設定を記述します。以下に設定例を示します。

```
[mysqld]
datadir=/MySQL_DATA
log-error=mysql-error.log
log-warnings=1
…(省略)…
```

上記により、/ MySQL_DATA / mysql-error.log にエラーログが出力されることになります。

コマンド･オプションと MySQL 設定ファイルの両方でエラーログ出力の設定を行った場合、コマンド･オプションで指定した値が優先されます。ただし、UNIX および Linux 環境にて、MySQL データベースの起動に **mysqld_safe** コマンドを利用した場合、コマンド･オプション設定で指定した**--log-error** オプションが **mysqld** に通知されません。このため、MySQL データベースの起動と停止情報のみがコマンド･オプションで指定したファイルに出力され、その他のメッセージは MySQL 設定ファイルに指定したファイルに出力されます。

1.1.3. 詳細説明

エラーログは以下のような情報を出力します。

- **起動時に出力される情報**
  以下のようなエラーや初期化情報が出力されます。
  - 起動オプション指定不正時エラー
  - データディレクトリの読み書き許可エラーや必要なファイルが存在しない場合のエラー
  - ストレージ･エンジン固有の情報
    ストレージ･エンジン毎の初期化処理や、リカバリ処理時のメッセージが出力されます。
- **動作中に出力される情報**
  以下のようなエラーや警告が出力されます。
  - メモリ不足などのクリティカルなエラー情報
  - MySQL レプリケーション構成時の予期しないコネクション断などの警告メッセージ
- **停止時に出力される情報**
  以下のような情報が出力されます。
  - 各ストレージ･エンジン固有の停止処理
  - MySQL データベースの停止完了

下記にエラーログのサンプルを示します。このサンプルは MySQL データベースの初回起動時のものです。このサンプルでは、InnoDB ストレージエンジン用のデータファイルが新規作成されていることがわかります。

```
070215 15:48:39  mysqld started
InnoDB: The first specified data file ./ibdata1 did not exist:
InnoDB: a new database to be created!
070215 15:48:39  InnoDB: Setting file ./ibdata1 size to 10 MB
InnoDB: Database physically writes the file full: wait...
070215 15:48:39  InnoDB: Log file ./ib_logfile0 did not exist: new to be created
InnoDB: Setting log file ./ib_logfile0 size to 5 MB
InnoDB: Database physically writes the file full: wait...
070215 15:48:39  InnoDB: Log file ./ib_logfile1 did not exist: new to be created
InnoDB: Setting log file ./ib_logfile1 size to 5 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
070215 15:48:40  InnoDB: Started; log sequence number 0 0
070215 15:48:40 [Note] /usr/local/mysql/bin/mysqld: ready for connections.
Version: '5.0.27-standard'  socket: '/home/mysql/tmp/testdb/mysql.sock'  port: 3306  MySQL
Community Edition - Standard (GPL)
…(省略)…
070215 16:04:55 [Note] /usr/local/mysql/bin/mysqld: Normal shutdown

070215 16:04:55  InnoDB: Starting shutdown...
070215 16:04:56  InnoDB: Shutdown completed; log sequence number 0 43655
070215 16:04:56 [Note] /usr/local/mysql/bin/mysqld: Shutdown complete

070215 16:04:56  mysqld ended
```

1.2. クエリーログ

1.2.1. 概要

クエリーログはクライアントとのコネクション確立･切断およびクライアントから送信されたすべての SQL ステートメントを記録します。SQL ステートメントのログ出力順序は、必ずしも実行順序とは一致しません。これらのログはクエリ実行完了後からロック解放までの間に出力されます。
アプリケーション開発時およびデバッグ時にクエリーログを有効にすることで、クライアントーサーバ間の通信を確認することができます。実運用中にクエリーログの出力を有効にした場合、多数のクエリが記録されることになり、パフォーマンスの低下およびクエリーログの肥大化を招く可能性があります。
クエリーログはコマンド･オプションまたは MySQL 設定ファイルで設定しない場合、出力されません。

1.2.2. 設定方法

1.2.2.1. 設定オプション

- **--log[=filename], -l [filename]**
  クエリーログの出力先ファイル名を指定します。ファイル名指定に相対パスを利用した場合、MySQL データベースのデータディレクトリからの相対パスで示されるディレクトリに、指定ファイル名で出力されます。ファイル名を指定しない場合、**ホスト名.log** で出力されます。
  オプションを指定しない場合でも MySQL データベースのデータディレクトリに出力されます。

1.2.2.2. コマンド・オプション設定例

クエリーログの出力設定例を以下に示します。以下の例ではクエリーログファイル名を mysql-query.log と指定しています。その他の設定はすべて MySQL 設定ファイル(my.cnf または my.ini)に設定されています。

UNIX および Linux の場合

```
$ mysqld_safe --log=mysql-query.log
```

Windows の場合

```
C:¥> mysqld-nt.exe --log=mysql-query.log
```

上記により、MySQL 設定ファイルで指定された MySQL データディレクトリに mysql-query.log というファイル名のクエリーログが出力されることになります。

1.2.2.3. MySQL 設定ファイル例

MySQL 設定ファイルでは、 mysqld セクションに設定を記述します。以下に設定例を示します。

```
[mysqld]
datadir=/MySQL_DATA
log=mysql-query.log
…(省略)…
```

上記により、/MySQL_DATA/mysql-query.log にクエリーログが出力されることになります。

コマンド･オプションと MySQL 設定ファイルの両方でクエリーログ出力の設定を行った場合、コマンド･オプションで指定した値が優先されます。

1.2.3. 詳細説明

1.2.3.1. 利用時の注意点

クエリーログはすべてのクライアントからの接続･切断および SQL ステートメントを記録するため、アプリケーション開発時やクエリ実行を監視する際に有用です。クエリーログの性質上、ファイルサイズが大きくなりやすいため、実運用中のシステムでの常用は避けた方が良いでしょう。

1.2.3.2. クエリーログの記録項目

クエリーログは以下のような情報を出力します。

- Time
  クエリーログへの記録時間を示します。この時間はクエリー実行完了時間からロック解除までの間の時間です。
- Id
  クライアントの識別番号を示し、コネクションが切断されるまで一意に割り振られます。
- Command
  クライアントからの接続やクエリー実施、切断などのコマンドが出力されています。
- Argument
  Command に対する引数です。Query の場合、SQL ステートメントが記述されます。

以下にクエリーログのサンプルを示します。

```
mysqld, Version: 5.0.27-community-log. started with:
Tcp port: 3306  Unix socket: (null)
Time                 Id Command    Argument
070215 17:14:01       1 Connect     root@localhost on
070215 17:14:06       1 Query       SELECT DATABASE()
                      1 Init DB     test
070215 17:14:26       1 Query       create table test_table (id int primary key, data text)
070215 17:14:44       1 Query       show create table test_table
070215 17:14:46       1 Quit
```

1.3. スロークエリーログ

1.3.1. 概要

MySQL 設定の long_query_time で設定されている秒数以上のクエリ実行時間を必要とした SQL ステートメントを記録するログです。記録するタイミングは、クエリが完了し、すべてのロックが解放された後になります。記録順序は実行順序とは異なる可能性があります。
アプリケーション開発時やデバッグ時に、処理時間のかかるクエリおよびインデックスを利用していないクエリを特定し、チューニングすべきクエリを識別するためにスロークエリーログを利用できます。
スロークエリーログは、コマンド・オプションまたは MySQL 設定ファイルに設定しない場合、出力されません。

1.3.2. 設定方法

1.3.2.1. 設定オプション

- **--log-slow-queries[=filename]**
  スロークエリーログの出力先ファイル名を指定します。ファイル名指定に相対パスを利用した場合、MySQL データベースのデータディレクトリからの相対パスで示されるディレクトリに、指定ファイル名で出力されます。ファイル名を指定しない場合、**ホスト名-slow.log** で出力されます。
- **--long-query-time=second**
  クエリ実施時間が指定秒数以上となる場合に、スロークエリログにクエリを記録します。1 以上の値を設定することができます。デフォルトは 10 (秒) です。
- **--log-queries-not-using-indexes**
  クエリ実施時間に関係なく、実施クエリがインデックスを利用しなかった場合に、スロークエリーログにクエリを記録します。デフォルトは記録しません。
- **--log-slow-admin-statements**
  OPTIMIZE TABLE, ANLYZE TABLE, ALTER TABLE のような管理用ステートメントの実行が遅い場合にも、スロークエリーログに記録するようにします。デフォルトは記録しません。

1.3.2.2. コマンド・オプション設定例

スロークエリーログの出力設定例を以下に示します。以下の例ではスロークエリーログファイル名を mysql-slow.log とし、スロークエリとして記録するクエリ実施時間を 30 秒、インデックスを利用しないクエリの記録、処理の遅い管理用ステートメントの記録を指定しています。その他の設定はすべて MySQL 設定ファイル(my.cnf または my.ini)に設定されています。

UNIX および Linux の場合

```
$ mysqld_safe --log-slow-queries=mysql-slow.log --long-query-time=30 ¥
              --log-queries-not-using-indexes --log-slow-admin-statements
```

Windows の場合

```
C:¥> mysqld-nt.exe --log-slow-queries=mysql-slow.log --long-query-time=30 ¥
                --log-queries-not-using-indexes --log-slow-admin-statements
```

上記により、MySQL 設定ファイルで指定された MySQL データディレクトリに mysql-slow.log というファイル名のスロークエリーログが出力されることになります。

1.3.2.3. MySQL 設定ファイル例

MySQL 設定ファイルでは、 mysqld セクションに設定を記述します。以下に設定例を示します。

```
[mysqld]
datadir=/MySQL_DATA
log-slow-queries=mysql-slow.log
long-query-time=30
log-queries-not-using-indexes
log-slow-admin-statements
…(省略)…
```

上記により、/MySQL_DATA/mysql-slow.log にスロークエリーログが出力されることになります。

コマンド･オプションと MySQL 設定ファイルの両方でスロークエリーログ出力の設定を行った場合、コマンド･オプションで指定した値が優先されます。

1.3.3. 詳細説明

1.3.3.1. スロークエリーログの記録項目

スロークエリーログは記録したクエリ毎に以下の情報を出力します。

- **クエリ実施日時**
- **クエリ実施ユーザおよび実施ホスト**
- **クエリ実施時間、ロック回数、行数**
- **実行クエリ**

スロークエリーログのサンプルを下記に示します。

```
# Time: 070220 14:34:00
# User@Host: root[root] @ localhost []
# Query_time: 20  Lock_time: 0  Rows_sent: 2000000  Rows_examined: 4000000
SELECT SID,CDATA,VIPD,CLTPT,VPDA,FROM_UNIXTIME(STIME,'%Y%m%d%H%i%s'),
FROM_UNIXTIME(ETIME,'%Y%m%d%H%i%s'),BYTESENT,BYTERECV,RID,URL INTO OUTFILE './1_test.txt'
FROM test ORDER BY CDATA,SID,ETIME,RID;
# Time: 070220 14:35:36
# User@Host: root[root] @ localhost []
# Query_time: 24  Lock_time: 0  Rows_sent: 0  Rows_examined: 0
LOAD DATA INFILE '/home/mysql/test-data.txt' INTO TABLE test;
# Time: 070220 14:35:58
# User@Host: root[root] @ localhost []
# Query_time: 22  Lock_time: 0  Rows_sent: 2000000  Rows_examined: 4000000
SELECT SID,CDATA,VIPD,CLTPT,VPDA,FROM_UNIXTIME(STIME,'%Y%m%d%H%i%s'),
FROM_UNIXTIME(ETIME,'%Y%m%d%H%i%s'),BYTESENT,BYTERECV,RID,URL INTO OUTFILE './2_test.txt'
FROM test ORDER BY CDATA,SID,ETIME,RID;
```

1.3.3.2. mysqldumpslow コマンドによるサマリの確認

チューニング対象のクエリ特定を容易にするため、スロークエリーログのサマリを作成する mysqldumpslow コマンドが用意されています。このコマンドをスロークエリーログに対して実行することで、問題となるクエリの実行回数、平均処理時間、ロック回数、参照した行数、クエリのサマリを確認することができます。

```
$ mysqldumpslow mysql-host-slow.log

Reading mysql slow query log from mysql-host-slow.log
Count: 1  Time=24.00s (24s)  Lock=0.00s (0s)  Rows=1999790.0 (1999790), root[root]@localht
  SELECT SID,CLTIP,SRVIP,CLTPT,SRVPT,FROM_UNIXTIME(STIME,'S'),FROM_UNIXTIME(ETIME,'S'),BYD

Count: 5  Time=23.80s (119s)  Lock=0.00s (0s)  Rows=0.0 (0), root[root]@localhost
  LOAD DATA INFILE 'S' INTO TABLE test

Count: 3  Time=21.00s (63s)  Lock=0.00s (0s)  Rows=2000000.0 (6000000), root[root]@localht
  SELECT SID,CLTIP,SRVIP,CLTPT,SRVPT,FROM_UNIXTIME(STIME,'S'),FROM_UNIXTIME(ETIME,'S'),BYD

Count: 1  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=4.0 (4), root[root]@localhost
  show databases

Count: 1  Time=0.00s (0s)  Lock=0.00s (0s)  Rows=2.0 (2), root[root]@localhost
  show tables
```

Windows **の場合も同様に実施できます。**

1.4. バイナリログ

1.4.1. 概要

バイナリログは、MySQL **データベースに対するテーブル作成･削除、データ挿入･更新･削除などの更新情報をイベントとして記録しているログファイルです。その内容はトランザクション･セーフであり、障害発生時にデータベースの状態を障害発生直前の状態まで戻す、ロールフォーワード･リカバリに利用することができます。**
また MySQL **レプリケーション構成で、マスターからスレーブに送信する情報を記録するためにもバイナリログは利用されます。**
バイナリログはコマンド･オプションまたは MySQL **設定ファイルで設定しない場合、出力されません。**

1.4.2. 設定方法

1.4.2.1. 設定オプション

- **--log-bin[=basename]**
  バイナリログの出力先ベース･ファイル名を指定します。ベース･ファイル名指定に相対パスを利用した場合、MySQL **データベースのデータディレクトリからの相対パスで示されるディレクトリに、指定ベース･ファイル名で特定されるファイル名で出力されます。ファイル名を指定しない場合、ホスト名-bin をベース･ファイル名として出力します。**
  ファイル名は、ベース･ファイル名.000001 の形式で、通し番号が増加していきます。また現在利用しているバイナリログ･ファイルを特定するためのインデックスファイルが ベースファイル名.index として作成されます。
- **--log-bin-index[=filename]**
  バイナリログのインデックスファイル名を指定します。相対パスでファイル名を指定した場合、MySQL **データベースのデータディレクトリからの相対パスでしめされるディレクトリに、指定ファイル名で出力します。設定を行わないか、ファイル名を指定しなかった場合、ホスト名-bin.index をファイル名とします。**
- **--max-binlog-size=size**
  バイナリログの最大ファイルサイズを指定します。最大値を超えると、バイナリログはログ･ローテートされ、新しいバイナリログ･ファイルに出力されます。指定可能な範囲は 4096 **から** 1G (4096 **バイトから**1G **バイト) で、デフォルト値は** 1G **です。**
- **--sync-binlog={0|1}**
  バイナリログへの書き込み時に、ファイル書き込みの同期を行うか設定します。同期を有効(1 **に設定)した場合、**MySQL **データベースはトランザクション毎にファイル書き込みの同期を行います。トランザクション情報を確実に記録する必要がある場合は、同期を有効にすべきです。**
  同期を無効 (0 **に設定) した場合、**MySQL **データベースは同期を行わず、オペレーティングシステムによってファイルの同期が行われることが期待されます。**
  デフォルトは無効(0 **に設定)されています。**
- **--expire-logs-days=day**
  古いバイナリログを自動削除するまでの日にちを指定します。自動削除は MySQL **データベースの起動時またはログ･ローテート発生時に実行されます。日にちに** 0 **を指定した場合、自動削除は行われません。**
  デフォルトは 0 **(自動削除を行わない) に設定されています。**

■ **--binlog-do-db=dbname**
指定データベースに対してのみバイナリログを出力します。他のデータベースに対する更新情報は無視されます。2 つ以上のデータベースに対してバイナリログの出力を指定する場合は、その数だけオプションを設定します。
注意すべき点があります。指定されていないデータベースを利用(USE コマンドでデータベースを指定)している際に、バイナリログ出力を指定したデータベース内のテーブルに更新を行った場合、その更新情報はバイナリログに記録されません (下記参照)。

```
/* バイナリログに出力されない例  --binlog-do-db=sales を設定 */
USE prices;                              /* バイナリログ出力をしないデータベースを利用 */
UPDATE sales.fy07  SET value=value+10;   /* バイナリログを出力するデータベースを更新 */
```

■ **--binlog-ignore-db=dbname**
指定データベースに対してバイナリログを出力しません。2 つ以上のデータベースに対して設定を行う場合は、その数だけオプションを設定します。
注意点すべき点があります。バイナリログ出力を指定しているデータベース利用時に、バイナリログ出力を禁止したデータベースのテーブルを更新した場合、バイナリログにその更新情報が記録されてしまいます (下記参照)。

```
/* バイナリログに出力されてしまう例 --binlog-ignore-db=sales を設定 */
USE prices;                              /* バイナリログ出力をするデータベースを利用 */
UPDATE sales.fy07  SET value=value+10;   /* バイナリログを出力しないデータベースを更新 */
```

1.4.2.2. コマンド・オプション設定例

バイナリログの出力設定例を以下に示します。以下の例ではバイナリログのベースファイル名を mysql-bin **とし、インデックスファイル名を** mysql-bin.index**、バイナリログの最大サイズを** 256MB**、自動削除日時を** 7 **日、記録対象データベースを** sales, prices **の** 2 **つのデータベースに設定しています。その他の設定はすべて** MySQL **設定ファイル**(my.cnf **または** my.ini)**に設定されています。**

UNIX **および** Linux **の場合**

```
$ mysqld_safe --log-bin =mysql-bin ¥
              --log-bin-index=mysql-bin.index --max-binlog-size=256M --expire-log-days=7¥
              --binlog-do-db=sales --binlog-do-db=prices
```

Windows **の場合**

```
c:¥> mysqld-nt.exe --log-bin =mysql-bin ¥
                   --log-bin-index=mysql-bin.index --max-binlog-size=256M --expire-log-days=7 ¥
                   --binlog-do-db=sales --binlog-do-db=prices
```

上記により、MySQL **設定ファイルで指定された** MySQL **データディレクトリに** mysql-bin **をベースファイル名にしたバイナリログが出力されることになります。バイナリログは** 256MB **前後で新しいバイナリログにログ･ローテーションされ、データベースの再起動またはログローテーション時に更新終了後** 7 **日を経過したバイナリログを削除します。またバイナリログに記録する対象データベースは、**MySQL **データベース内の** sales **および** prceis **データベースのみとなります。**

1.4.2.3. MySQL 設定ファイル例

MySQL **設定ファイルでは、** mysqld **セクションに設定を記述します。以下に設定例を示します。**

```
[mysqld]
datadir=/MySQL_DATA
log-bin=mysql-bin
log-bin-index=mysql-bin.index
max-binlog-size=256M            # 1GB を指定する場合は 1G と記述
expire-log-days=7
binlog-do-db=slaes
binlog-do-db=prices
…(省略)…
```

コマンド･オプションと MySQL **設定ファイルの両方でバイナリログ出力の設定を行った場合、コマンド･オプションで指定した値が優先されます。**

1.4.3. 詳細説明

1.4.3.1. バイナリログ利用時のパフォーマンス低下について

バイナリログを利用した場合、MySQL データベースのパフォーマンスが約 1%低下する可能性があります。しかし障害発生時やリストア時のロールフォーワード・リカバリが利用でき、またレプリケーション機能の利用を可能となる利点もあります。システム要件にあわせ利用の可否を決定する必要があります。

1.4.3.2. バイナリログの内容確認

バイナリログは名前の通り、テキストファイルではなく特殊フォーマットで出力されています。内容確認を行うためには、mysqlbinlog コマンドを利用する必要があります。以下に利用例を示します。下記例では newdb という新規データベースの作成を確認できます。

```
$ mysqlbinlog mysql-host-bin.000009
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
# at 4
#070216 16:33:39 server id 1  end_log_pos 98    Start: binlog v 4, server v 5.0.27-standard-log
created 070216 16:33:39 at startup
# Warning: this binlog was not closed properly. Most probably mysqld crashed writing it.
ROLLBACK;
# at 98
#070216 16:36:16 server id 1  end_log_pos 183   Query   thread_id=1     exec_time=0
error_code=0
SET TIMESTAMP=1171611376;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=1, @@session.unique_checks=1;
SET @@session.sql_mode=0;
/*!¥C latin1 */;
SET
@@session.character_set_client=8,@@session.collation_connection=8,@@session.collation_server=8;
create database newdb;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
```

Windows の場合も同様に実施できます。

バイナリログは「# at 4」の行から「# at 98」の行までのような単位で 1 つのイベント（更新情報）を記録しています。上記では「# at 98」の行から CREATE DATABASE の実行が記録されていることが分かります。このようにバイナリログにはすべての更新情報が記録されているため、バックアップデータと、バックアップ時から障害発生時までのバイナリログが存在すれば、データベースの復旧を行うことが可能になります。

1.4.3.3. バイナリログによる増分バックアップ

バイナリログには更新情報が保持されているため、バイナリログを定期的にバックアップすることで、MySQL データベースの増分バックアップとすることができます。
例えば、1 週間に 1 回フルバックアップを行い、1 日に 1 度バイナリログをバックアップするとします。「図 2 バイナリログを利用したデータベース復旧」に示すように、障害発生時の未バックアップのバイナリログを合わせて、フルバックアップデータをリストアし、その後バックアップしておいたバイナリログを適用していくことで、障害発生直前の状態まで MySQL データベースを復旧することが可能になります。

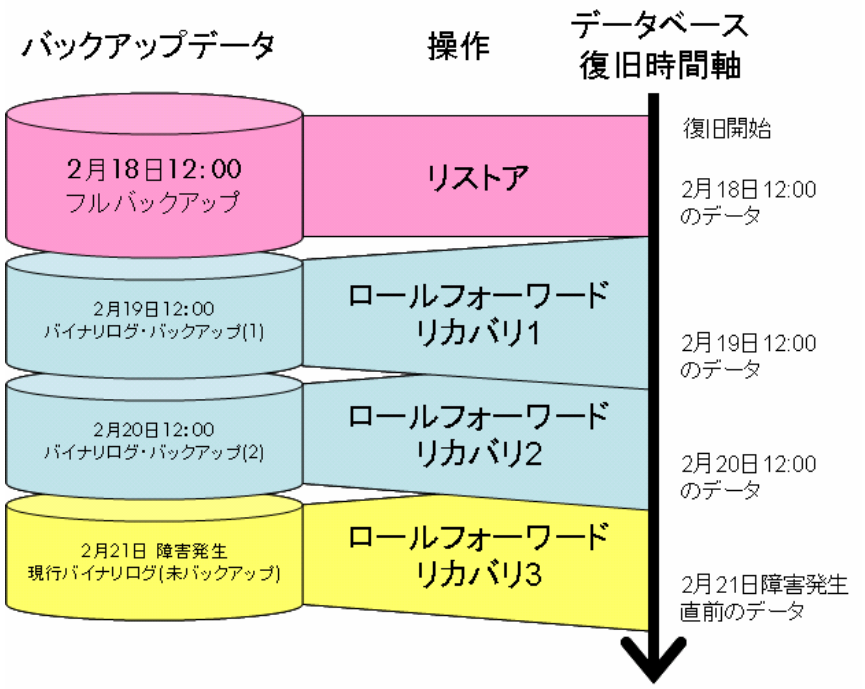

図 2 バイナリログを利用したデータベース復旧

1.5. ログの管理

1.5.1. UNIX および Linux でのログ・ローテーション

MySQL データベースは様々なログを出力しつづけます。このためログファイルがディスク・スペースを圧迫しないようにログのバックアップ・削除を行う必要があります。

MySQL データベースではコマンド **mysqladmin flush-logs** または SQL ステートメント **FLUSH LOGS** を利用してログをフラッシュすることができます。これらは以下の処理を行います。

- 既存のエラーログを、ファイル名サフィックスに “-old”を付加したファイル名に変更してクローズします。
- 新規にエラーログを作成し、オープンします。
- クエリーログおよびスロークエリーログをクローズし、再度ファイルを追記型でオープンします。同一ファイル名のログファイルが存在しない場合は新規に作成します。
- 現在利用中のバイナリログをクローズし、新しいシーケンス番号のバイナリログをオープンします。

クエリーログ、スロークエリーログおよびバイナリログを利用していない場合は、それらのログファイルに対する上記操作は行われません。

この処理を利用して、ログファイルのバックアップを行うことができます。以下は、エラーログ、クエリーログ、スロークエリーログをバックアップディレクトリにログを移動する手順を示したものです。

(1) **ファイル名変更**

```
$ su - mysql                                          # MySQL サーバ管理ユーザに切り替え
$ mv mysql-host.log mysql-host.log.back               # クエリーログ名を変更
$ mv mysql-host-slow.log mysql-host-slow.log.back     # スロークエリーログ名を変更
```

ファイル名変更後も MySQL データベースのログは、それぞれのログファイルに記録されます。

(2) **ログのフラッシュ**

```
$ mysqladmin -u root flush-logs # ログをフラッシュする
```

RELOAD 権限を持つデータベース・ユーザを指定する必要があることに注意してください。
この操作により、既存ログファイルがクローズされ、以下のログファイルを新規に生成してログの出力先を切り替えます。
エラーログ: mysql-host.err (既存エラーログは mysql-host.err-old にファイル名変更)

クエリーログ: mysql-host.log
スロークエリーログ: mysql-host-slow.log
バイナリログ: mysql-bin.XXXXXX (以前のバイナリログよりシーケンス番号を+1 したファイル名)

(3) ログのバックアップ

```
$ mv mysql-host.err-old *.back backup_dir/                    # バックアップディレクトリにログを移動
```

mysqladmin flush-logs によって新しいログファイルが生成されているため、ファイル名を変更した各ログファイルをバックアップまたは削除が可能となります。ただし操作対象の MySQL データベースが MySQL レプリケーション構成のマスターである場合、古いバイナリログの内容がすべてのスレーブのリレイログに反映されたことを確認してから削除する必要があります。反映前に削除した場合、データの整合性が損なわれることになります。

1.5.2. Windows でのログ・ローテーション

エラーログおよびバイナリログ

エラーログおよびバイナリログについては、Unix および Linux の場合と同じく **mysqladmin flush-logs** で、ログ・ローテーションの実施が可能です。

```
C:¥MYSQL_DATA> mysqladmin.exe -u root flush-logs
```

上記コマンドにより、以下のログファイルが新規に作成されます。

エラーログ: mysql-host.err (既存エラーログは mysql-host.err-old にファイル名変更)
バイナリログ: mysql-bin.XXXXXX (以前のバイナリログよりシーケンス番号を+1 したファイル名)

クエリーログおよびスロークエリーログ

クエリーログおよびスロークエリーログについては、Microsoft Windows ではプロセスがファイルをロックするため、MySQL データベース起動中にログファイル名の変更を行えません。このため UNIX および Linux と同様の方法ではログのバックアップ・削除を行うことができません。MySQL データベースを一度停止し、ログファイルの削除またはバックアップ作業を行う必要があります。

1.6. その他のログ

MySQL データベースは前述したログファイル以外にもいくつかの特殊なログファイルを利用する場合があります。これらは特定条件下でのみ利用されます。「表 2 MySQL データベースのその他のログ」に主なログファイルを示します。

表 2 MySQL データベースのその他のログ

ログファイル	概要説明
リレイログ	MySQL レプリケーション環境下のスレイブで利用されます。マスターの更新情報 (バイナリログ) がこのログにコピーされており、スレイブはリレイログの内容を順次読み込んでデータを更新し、マスターのデータとの同期を取ります。
InnoDB ログ	InnoDB ストレージエンジン利用時に作成・利用されるログです。障害発生時のクラッシュ・リカバリに利用されます。

その他のログファイルについては MySQL 5.0 Reference Manual を参照してください。

2. ステータス監視

MySQL データベース実行時の設定状況および実行状況を確認するための SQL ステートメントが用意されています。障害の原因切り分けや、MySQL データベースのチューニングを行うために、ステータスを収集することが可能です。

2.1. 起動中のMySQLデータベースの設定情報の収集

障害発生時の切り分けのための基本的な情報として MySQL データベースの設定情報が必要とされることがあります。MySQL 設定ファイルおよびコマンド･オプションからも MySQL データベースの設定情報を得ることができますが、MySQL データベースのデフォルト設定値および、取得時点での MySQL データベースの設定値を得ることができるため、本節で説明する方法で MySQL データベースの設定情報を収集することをお勧めします。

MySQL データベースの設定情報は、MySQL データベース全体で共通な設定と、各クライアントとのセッション毎の設定の 2 種類が存在します。セッション毎の設定の初期値は MySQL データベース全体の共通設定となっており、セッション中の MySQL データベースの設定情報変更が反映されます。MySQL データベースの設定情報は以下の SQL ステートメントを実行することで取得することができます。

- MySQL データベース全体の共通設定を取得
  SHOW GLOBAL VARIABLES;

- セッション毎の設定情報を取得
  SHOW SESSION VARIABLES;

SHOW VARIABLES として実行した場合、セッション毎の設定情報を取得します。

同様に **mysqladmin variables** をコマンドラインから実行することでも、設定情報を取得することもできます。この場合、MySQL データベース全体の共通設定を取得した場合と同じ結果を得ることができます。

```
$ mysqladmin variables
+---------------------------------+-----------------------------------------------------+
| Variable_name                   | Value                                               |
+---------------------------------+-----------------------------------------------------+
| auto_increment_increment        | 1                                                   |
| auto_increment_offset           | 1                                                   |
| automatic_sp_privileges         | ON                                                  |
| back_log                        | 50                                                  |
| basedir                         | /usr/local/mysql-standard-5.0.27-linux-x86_64-glibc23|
...
```

Windows でも同様に実行可能です。

各設定情報の詳細は、MySQL 5.0 Reference Manual 5.2.3. System Variables を参照してください。

2.2. MySQLのステータス情報の収集

MySQL データベース全体の実行状況を把握するためにステータス情報を取得することができます。MySQL のステータス情報を定期的にまたは着目する操作の前後で取得し、2 つの異なる時点での結果の差分を確認することで、MySQL データベースのチューニングや、障害解析に利用することが可能になります。

MySQL データベースの設定情報は以下の SQL ステートメントを実行することで取得することができます。

- MySQL データベース全体のステータス情報を取得
  SHOW GLOBAL STATUS;

- セッション毎のステータス情報を取得
  SHOW SESSION STATUS;

SHOW STATUS として実行した場合、セッション毎のステータス情報を取得します。

同様に mysqladmin extended-status をコマンドラインから実行することでも、設定情報を取得することができます。コマンドを利用した場合、MYSQL データベース全体のステータス情報を取得することになります。

```
$ mysqladmin extended-status
+----------------------------------+----------+
| Variable_name                    | Value    |
+----------------------------------+----------+
| Aborted_clients                  | 0        |
| Aborted_connects                 | 0        |
| Binlog_cache_disk_use            | 0        |
| Binlog_cache_use                 | 0        |
| Bytes_received                   | 735      |
...
```

Windows でも同様に実行可能です。

各ステータス情報の詳細については、MySQL 5.0 Reference Manual 5.2.5. Status Variables を参照してください。

2.3. InnoDBストレージエンジンのステータス情報の収集

InnoDB ストレージエンジンを利用している場合、MySQL データベース全体のステータス情報のほかに InnoDB ストレージエンジン固有のステータス情報を取得することが可能です。InnoDB ストレージエンジンのステータス情報を収集することで、InnoDB データベースのチューニングのための情報を収集することが可能です。

InooDB ストレージエンジンのステータス情報の取得には、SQL ステートメント SHOW INNODB STATUS\G を利用します。具体的には mysql コマンド等を利用して、以下のように取得します。

```
$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or ¥g.
Your MySQL connection id is 6 to server version: 5.0.27-standard-log

Type 'help;' or '¥h' for help. Type '¥c' to clear the buffer.

mysql> SHOW INNODB STATUS¥G
*************************** 1. row ***************************
Status:
=====================================
070222 16:59:05 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 43 seconds
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 2, signal count 2
Mutex spin waits 1, rounds 20, OS waits 0
RW-shared spins 4, OS waits 2; RW-excl spins 1, OS waits 0
------------
…(省略)…
```

Windows でも同様に実行可能です。

2.4. プロセスリストの取得

現在実行中のスレッドリストを取得することができます。PROCESS 権限を持つユーザならば、MySQL データベースで実行中のすべてのスレッドリストを取得できます。権限を持たない場合は、そのユーザに紐づいているスレッドリストのみ取得できます。プロセスリストの取得により、クライアント接続中のスレッドの状態および、MySQL レプリケーション構成時に動作する各種スレッドの状態などを確認することができます。

プロセスリストの取得には、SQL ステートメント SHOW PROCESSLIST\G を利用します。具体的には mysql コマンド等を利用して、以下のように取得します。

```
$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or ¥g.
Your MySQL connection id is 6 to server version: 5.0.27-standard-log

Type 'help;' or '¥h' for help. Type '¥c' to clear the buffer.

mysql> SHOW PROCESSLIST¥G
*************************** 1. row ***************************
     Id: 1
   User: root
   Host: localhost
     db: NULL
Command: Query
   Time: 0
  State: NULL
   Info: SHOW PROCESSLIST
1 row in set (0.00 sec)
```

Windows でも同様に実行可能です。

なんらかの障害のため、特定のプロセスを停止したい場合、SQL ステートメント KILL を利用することができます。以下に mysql コマンドを利用した例を示します。

```
mysql> show processlist;
+----+------+-----------+------+---------+------+-------+------------------+
| Id | User | Host      | db   | Command | Time | State | Info             |
+----+------+-----------+------+---------+------+-------+------------------+
|  1 | root | localhost | NULL | Sleep   |    8 |       | NULL             |
|  2 | root | localhost | NULL | Query   |    0 | NULL  | show processlist |
+----+------+-----------+------+---------+------+-------+------------------+
2 rows in set (0.00 sec)

mysql> kill 1;
Query OK, 0 rows affected (0.00 sec)

mysql> show processlist;
+----+------+-----------+------+---------+------+-------+------------------+
| Id | User | Host      | db   | Command | Time | State | Info             |
+----+------+-----------+------+---------+------+-------+------------------+
|  2 | root | localhost | NULL | Query   |    0 | NULL  | show processlist |
+----+------+-----------+------+---------+------+-------+------------------+
1 row in set (0.00 sec)
```

3. ヘルスチェック

3.1. MySQLデータベース･プロセスの監視によるヘルスチェック

UNIX や Linux の場合は **ps** コマンドを、Windows(2000/XP/2003)の場合 **tasklist** コマンドを利用することで、OS 上での MySQL データベースのプロセス監視が可能です。OS 付属のコマンドで手軽にプロセスの監視を行うことができますが、MySQL データベースの内部の稼働状況を監視できるわけではないため、正常にサービスを提供できるかの判断まではできません。次項で説明する **mysqladmin ping** と併用した監視を行うべきです。

以下に実行例を示します。

Unix および Linux の場合

```
$ ps -e | grep mysqld | grep -v mysqld_safe | grep -v grep
13306 pts/3    00:00:00 mysqld
```

Windows 2000/XP/2003 の場合

```
C:¥> tasklist /FI "IMAGENAME eq mysqld-nt.exe"
イメージ名                     PID セッション名     セッション# メモリ使用量
========================= ====== ================ ======== ============
mysqld-nt.exe                3624 Console                  0      9,524 K
```

3.2. MySQLデータベースへの接続確認によるヘルスチェック

プロセス監視以外に MySQL データベースのヘルスチェックを簡単に行う方法として、 **mysqladmin ping** を利用する方法があります。この方法での監視では、**mysqladmin** コマンドにより MySQL データベースへのコネクションが作成できるか否かで、稼動の可否が判断されます。つまり MySQL データベースプロセスが稼動していても、なんらかの原因により接続が拒否された場合、稼動していないと判断されます。
このため、MySQL データベースのフェイル･オーバー構成を構築する場合には、前項で説明する MySQL データベース･プロセスの監視と併用して監視すべきです。

以下に実行例を示します。

接続可能な場合

```
$ mysqladmin ping
mysqld is alive
```

接続できない場合(UNIX ドメイン･ソケット利用時のメッセージ)

```
$ mysqladmin ping
mysqladmin: connect to server at 'localhost' failed
error: 'Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)'
Check that mysqld is running and that the socket: '/tmp/mysql.sock' exists!
```

Appendix 1: MySQL 設定ファイル・サンプル

本書で利用した MySQL 設定ファイルを以下に示します。この設定ファイルは必要最小限の設定のみを記載しており、実際の利用では、より詳細な設定が必要です。
またファイル名は、UNIX および Linux 環境では **my.cnf** とし、Windows 環境では **my.ini** となっています。
Windows 環境においても **my.cnf** を利用可能ですが、ファイルが短縮ダイアルに関連づけられているため、編集などのファイルの取り扱いに問題が発生します。

UNIX および Linux の MySQL 設定ファイル (my.cnf)

```
[client]
port=3306
socket=/MySQL_DATA/mysql.sock

[mysqld]
datadir=/MySQL_DATA
port=3306
socket=/MySQL_DATA/mysql.sock
```

Windows の MySQL 設定ファイル(my.ini)

```
[client]
port=3306
socket=C:/MySQL_DATA/mysql.sock
user=root

[mysqld]
datadir=C:/MySQL_DATA
port=3306
socket=C:/MySQL_DATA/mysql.sock
```

Windows環境でのディレクトリ・セパレータは、スラッシュ(/)を利用してください。

Appendix 2: バイナリログ関連のその他のオプション設定

バイナリログの設定として本書で説明したオプションの他に以下のオプションが提供されています。これらの設定は特殊な条件下または MySQL データベースのチューニング時に利用する可能性があります。通常、設定は不要です。

- **--log-bin-trust-function-creators[={0|1}]**
  MySQL データベースで FUNCTION を定義、利用する場合に設定が必要になる場合があります。

  CREATE FUNCTION ステートメント実行時の MySQL の振る舞いを設定します。乱数生成など演算結果が一定しない非決定的な関数を定義し、利用した場合、バイナリログの記録からデータベースの状態を再現できなくなる場合がありえます。このため関数定義時に MySQL は関数の値の決定性を評価します。このオプションは、その評価の実行を抑止する(1 に設定)か、実施(0 に設定)するかを指定します。

- **--binlog-cache-size=size**
  MySQL データベースのチューニング時に設定が必要とる場合があります。

  1 つのトランザクション中のバイナリログのための SQL ステートメントを保持するためのキャッシュサイズを指定します。このキャッシュはクライアント毎に確保されます。巨大なトランザクションの実行が必要な場合はキャッシュサイズを大きくする必要があります。
  デフォルトのキャッシュサイズは 32KB です。

- **--max-binlog-cache-size=size**
  MySQL データベースのチューニング時に設定が必要とる場合があります。

  1 つの複数ステートメントからなるトランザクションが、設定値以上のメモリを必要とする場合、ストレージエラーを生成します。

- **--log-slave-updates={0|1}**
  MySQL レプリケーション構成で、スレーブサーバ上でリカバリ等の要求で完全なバイナリログが必要な場合や、スレーブサーバからより下位のスレーブサーバに対してデータをレプリケートする必要がある場合に設定します。

  MySQL レプリケーション構成のスレーブで、マスターから通知された更新情報をスレーブのバイナリログにも記録するか否かを設定します。1 に設定すると記録します。デフォルトは 0 です。

Appendix 3: MySQL データベースのコネクション自動切断の問題

A3.1. MySQLデータベースの接続方法とコネクション自動切断

MySQL データベースとクライアントとのコネクションは大きくわけて 2 つの接続方法があります。

- ネットワーク経由での接続
  リモートサーバ等から TCP/IP ネットワークを利用して接続します。JDBC ドライバは MySQL データベースと同一サーバ上でクライアントが動作した場合でも、ネットワーク経由での接続になります。
- ローカルでの接続
  MySQL データベースと同一サーバ上でクライアントが動作し、UNIX ドメイン・ソケット、ネームド・パイプまたは共有メモリを利用して、クライアントと接続します。

MySQL データベースとネットワーク経由での接続を利用する場合、MySQL データベースのコネクション自動切断処理に留意する必要があります。ローカルでの接続を行っている場合には、コネクション自動切断は行われません。

MySQL データベースのコネクション自動切断は、一定時間以上コネクションが利用されなかった場合に発生します。この振る舞いはクライアントが正常にコネクションをクローズせずに終了した場合など、不要と考えられるコネクションを破棄するために実施されます。デフォルトではコネクションが 8 時間 (28800 秒) 利用されなかった場合にコネクションを切断します。クライアント側はコネクション切断を検知できないため、コネクションを長期間維持・利用するアプリケーションではなんらかの対応が必要となります。

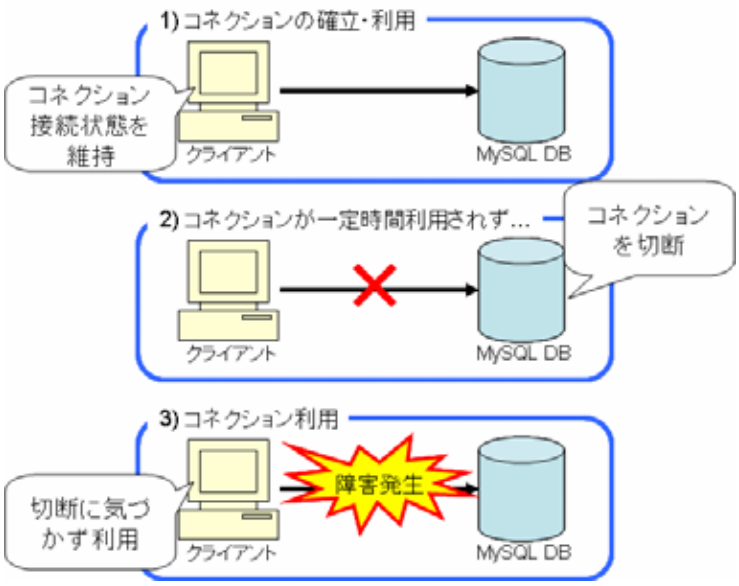

図 3 MySQL データベースのコネクション自動切断

MySQL データベースのコネクション自動切断の時間は以下の 2 つの設定を MySQL 設定ファイルに記述することで行えます。

- **interactive_timeout**
  インタラクティブな TCP/IP 接続のコネクションを切断するまでのアイドル秒数を指定します。デフォルトは 28800 (8 時間)です。**mysql** コマンド利用時など、MySQL データベースとクライアントが対話的に通信処理を行う場合に利用されます。
- **wait_timeout**
  非インタラクティブな TCP/IP 接続のコネクションを切断するまでのアイドル秒数を指定します。デフォルトは 28800 (8 時間)です。JDBC ドライバなどを利用して、MySQL データベースとクライアントが通信処理を行う場合に利用されます。

どちらのタイムアウト設定値を利用するかは、クライアント側からの接続時の接続設定に依存します。例えば **mysql** コマンドでインタラクティブと非インタラクティブを設定するには、起動オプションの **--batch** の有無で決定されます。

- インタラクティブな TCP/IP 接続で利用する場合
  --batch オプションを追加せずに mysql コマンドを実行します。このとき mysql コマンドのプロンプトが表示され、対話的に MySQL データベースを操作することが可能になります。
- 非インタラクティブな TCP/IP 接続で利用する場合
  --batch オプションを追加して mysql コマンドを実行します。このとき mysql コマンドのプロンプトが表示されず、クエリの結果出力フォーマットのカラムのセパレータがタブになります。また実行した処理のヒストリも利用できません。

A3.2. コネクション自動切断に対する対応

コネクションを長期間維持し、利用し続けるアプリケーションや、コネクションプールを利用するアプリケーションでは、長期間コネクションがアイドル状態になる可能性があります。このため MySQL データベースのコネクション自動切断処理によりコネクションが切断されてしまい、想定外の障害が発生する可能性があります。

この問題の対応方法として以下が考えられます。

- **コネクション利用の度に新規接続を行う。**
  コネクションは利用する度に新規接続を行い、利用後は切断するようにします。
- **コネクションを利用する直前に接続確認処理を行う。**
  アイドル状態後、はじめてコネクションを利用するタイミングで、コネクションの接続確認処理を行います。
  コネクションプール利用時には、コネクションプールからアプリケーションへコネクションを渡す直前に、コネクションの接続確認処理を実施するように設定します。
- **定期的にコネクションに対して接続確認処理を行う。**
  コネクションを利用しない場合でも、定期的にコネクションの接続確認処理を行うようにします。
  コネクションプール利用時では、実装によって定期的にコネクションプール中のコネクションの接続確認処理を行う設定を持つものがあります。

アプリケーションに対する修正を最小限に、かつ効率よくコネクションを利用するためには、コネクションプールを利用し、コネクションの接続確認処理機能を必ず設定することをお勧めします。

お問い合わせはカスタマー インフォメーションセンターへ
03-5304-6660 **月～金**9:00～19:00 **土**10:00～18:00（**日、祝祭日、年末年始および5/1を除く**）
Linux/**オープンソース製品に関する情報は** http://www.hp.com/jp/linux/

日本ヒューレット・パッカード株式会社
〒102-0076 東京都千代田区五番町 7