

MySQLのバックアップ・リカバリ (オープンソースカンファレンス2005)

2005年3月26日

ソニーグローバルソリューションズ株式会社
プラットフォーム技術部
松信 嘉範 (MATSUNOBU Yoshinori)

■ 基本事項の整理

- ◆ MySQLの機能概要
- ◆ メディア障害とリカバリの流れ

■ バイナリログ

■ コールドバックアップ・リカバリ

■ オンラインバックアップ・リカバリ

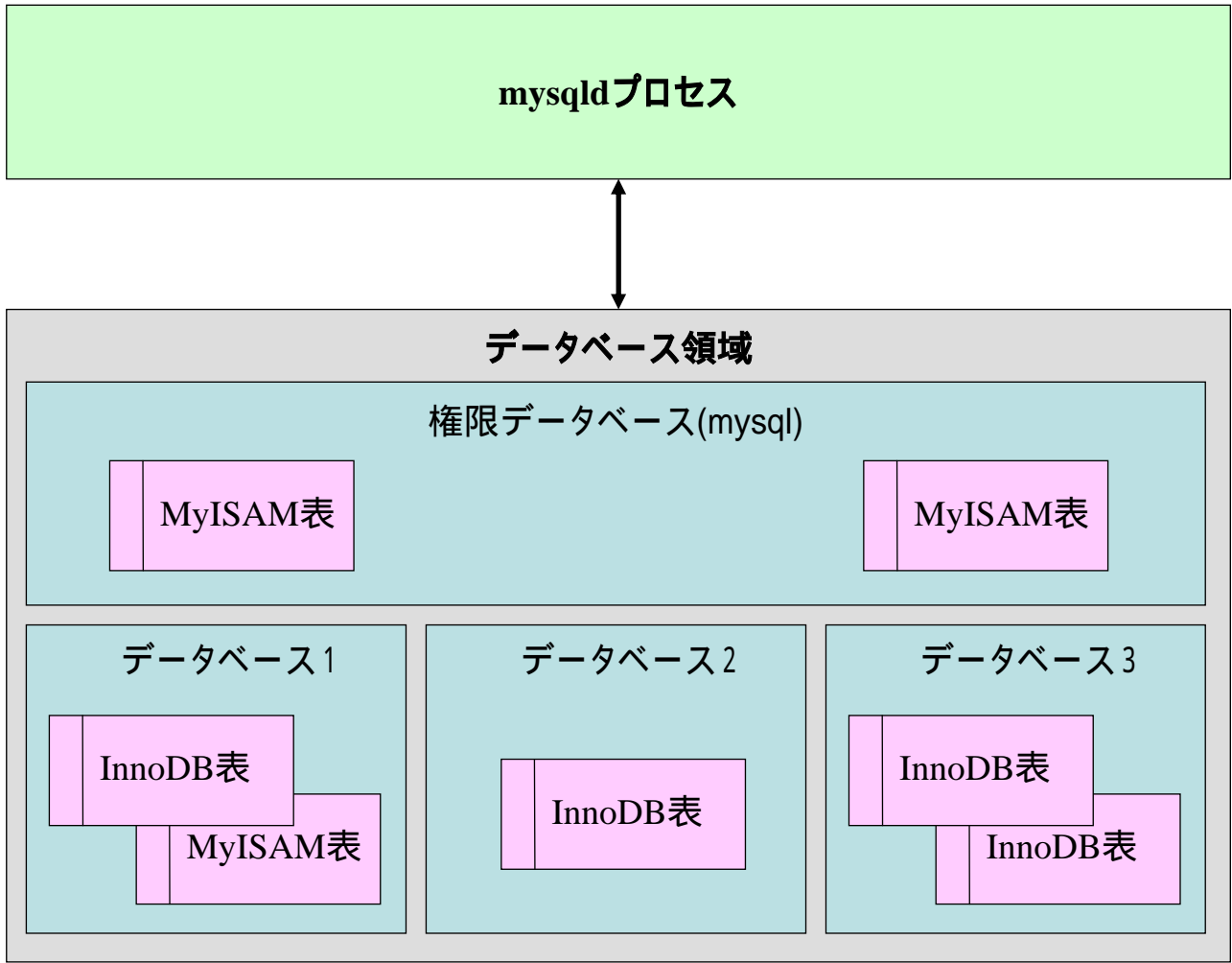
- ◆ mysqldump
- ◆ FLUSH TABLES WITH READ LOCK

■ その他のトピック

- ◆ メディア障害を考慮したファイル配置
- ◆ 過去の一時点へのリカバリ
- ◆ 内部的な動作

MySQLの特徴/機能概要

観点	特徴
動作環境	Linux, Solaris, HP-UX, AIX, Windows他
実装言語	C/C++(マルチスレッド)
SQL文	<ul style="list-style-type: none"> ・SQL92の大半とSQL99 Coreの一部、また便利な独自構文を提供 ・サブクエリは4.1から、ストアドプロシージャは5.0から
テーブル定義	<ul style="list-style-type: none"> ・InnoDB、MyISAM等の「型」がある(ストレージエンジン) ・B-Tree索引、整合性制約、シーケンスをサポート(ビューは5.0から)
トランザクション	<ul style="list-style-type: none"> ・ロック無しでの読み取り一貫性、行ロック、4段階全ての分離レベルなどをサポート(InnoDBのみ)
バックアップ	<ul style="list-style-type: none"> ・ロックをかけずに一貫性のあるオンラインバックアップが可能(InnoDBのみ) ・増分バックアップ相当の機能を提供
リカバリ	<ul style="list-style-type: none"> ・ロールフォワードリカバリ(過去の一時点も含む)が可能 ・クラッシュリカバリが可能(InnoDBのみ)
日本語処理	<ul style="list-style-type: none"> ・列単位でキャラクタセットの指定が出来る(4.1から) ・ベンダー定義文字のコード変換はサポートしていない(文字化けする)
セキュリティ	<ul style="list-style-type: none"> ・認証手段は、パスワード認証とSSLクライアント証明書認証 ・暗号化の手段はSSLとVPN(SSHポートフォワードリングなど)



MySQLのディレクトリ/ファイルの種類

ベースディレクトリ

データディレクトリ
データベースディレクトリ

表定義ファイル

InnoDBデータファイル
InnoDBログファイル
MyISAM表ファイル
MyISAM索引ファイル

バイナリログファイル

エラーログファイル
一般クエリログファイル
スロークエリログファイル

ソケットファイル
PIDファイル

ディレクトリ・ファイルの種類

```
[mysqld]
basedir=/usr/mysql4110
datadir=/data/mysql4110
innodb_data_home_dir=/data/mysql4110/idata
innodb_data_file_path= ibdata1:1000M:autoextend
innodb_log_group_home_dir=/data/mysql4110/ilog
socket=/data/mysql4110/mysql.sock
log-bin=/data/mysql4110/blog/mysql-host
```

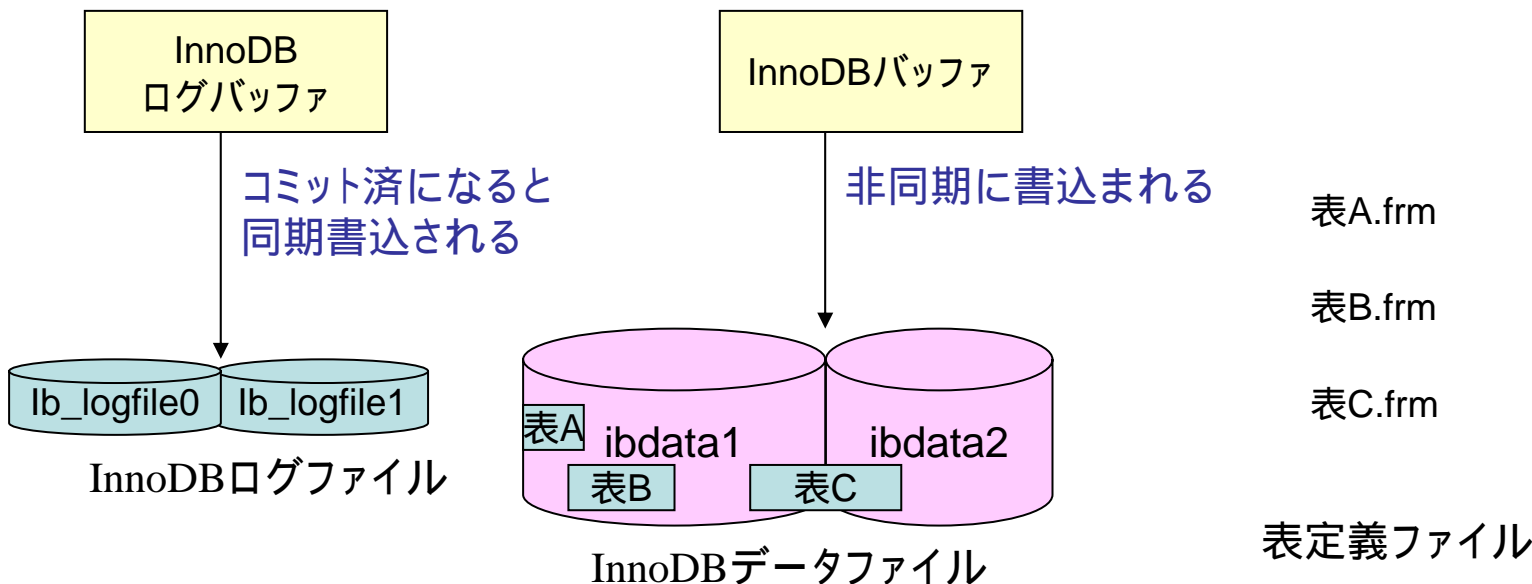
初期化パラメータファイル(my.cnf)

テーブル(ストレージエンジン)とファイルの関係

表A ~ 表C: InnoDB

表D ~ 表F: MyISAM とすると...

InnoDB



MyISAM

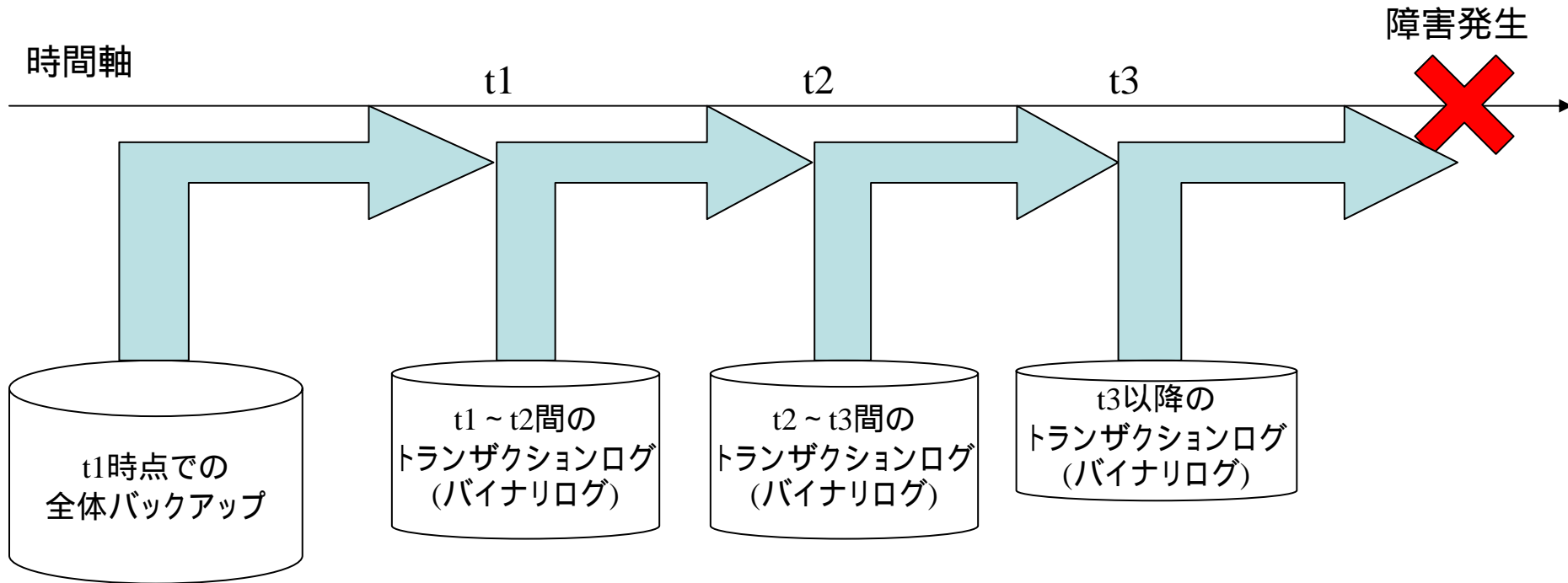
表D.MYD	表D.MYI	表D.frm
表E.MYD	表E.MYI	表E.frm
表F.MYD	表F.MYI	表F.frm
表データファイル	索引データファイル	表定義ファイル

メディア障害とは

- アプリケーションの動作に必要な表やファイルが消失し、アプリケーションとして機能しなくなる障害のこと

- メディア障害の主な要因
 - ◆ ハードウェア障害
 - ◆ ソフトウェア障害
 - ◆ 人為的ミス

- メディア障害からの回復手段
 - ◆ 事前に全体バックアップを取っておく
 - ◆ リストアして、リカバリする
 - リストア: バックアップの時点まで復旧すること
 - リカバリ: 障害直前の時点まで復旧すること(ロールフォワードリカバリ)



・過去のバックアップをリストアした後、バックアップ以降のトランザクションログ(バイナリログ)を順次適用し、障害直前の状態に復旧する。これをロールフォワードリカバリと言う。

・MySQLでは、全体バックアップに「一貫性」がなければならない。

一貫性: t1時点でコミットされているデータのみをバックアップしていること

■ 基本事項の整理

- ◆ MySQLの機能概要
- ◆ メディア障害とリカバリの流れ

■ バイナリログ

■ コールドバックアップ・リカバリ

■ オンラインバックアップ・リカバリ

- ◆ mysqldump
- ◆ FLUSH TABLES WITH READ LOCK

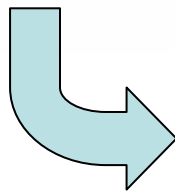
■ その他のトピック

- ◆ メディア障害を考慮したファイル配置
- ◆ 過去の一時点へのリカバリ
- ◆ 内部的な動作

バイナリログ(1)

- バイナリログとは
 - ◆ MySQLのトランザクションログ(バイナリファイル)
 - ロールフォワードリカバリには必須
 - ◆ SQL文を時系列に蓄積したもの
 - ◆ コミット後に書き込まれる
 - ◆ mysqlbinlogというコマンドでテキスト形式のSQL文に変換できる

```
[mysql@mysql-host]$ mysqlbinlog /data/mysql4110/blog/mysql-host.000001
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
# at 4
#050220 0:00:34 server id 1 log_pos 4          Start: binlog v 3, server v 4.1.10-standard-log created 050220 0:00:34 at startup
# at 79
#050220 0:00:48 server id 1 log_pos 79        Query  thread_id=1      exec_time=0      error_code=0
use db1;
SET TIMESTAMP=1108825248;
CREATE TABLE tbl1 (col1 INTEGER, col2 DATETIME) ENGINE=InnoDB;
# at 174
#050220 0:00:48 server id 1 log_pos 174       Query  thread_id=1      exec_time=0      error_code=0
SET TIMESTAMP=1108825248;
INSERT INTO tbl1 VALUES(100,SYSDATE());
# at 246
#050220 0:00:50 server id 1 log_pos 246       Query  thread_id=1      exec_time=0      error_code=0
SET TIMESTAMP=1108825250;
DROP TABLE tbl1;
[mysql@mysql-host]$
```



```
use db1;
SET TIMESTAMP=1108825248;
CREATE TABLE tbl1 (col1 INTEGER, col2 DATETIME) ENGINE=InnoDB;
SET TIMESTAMP=1108825248;
INSERT INTO tbl1 VALUES(100,SYSDATE());
SET TIMESTAMP=1108825250;
DROP TABLE tbl1;
```

バイナリログ(2)

■ バイナリログの取得方法

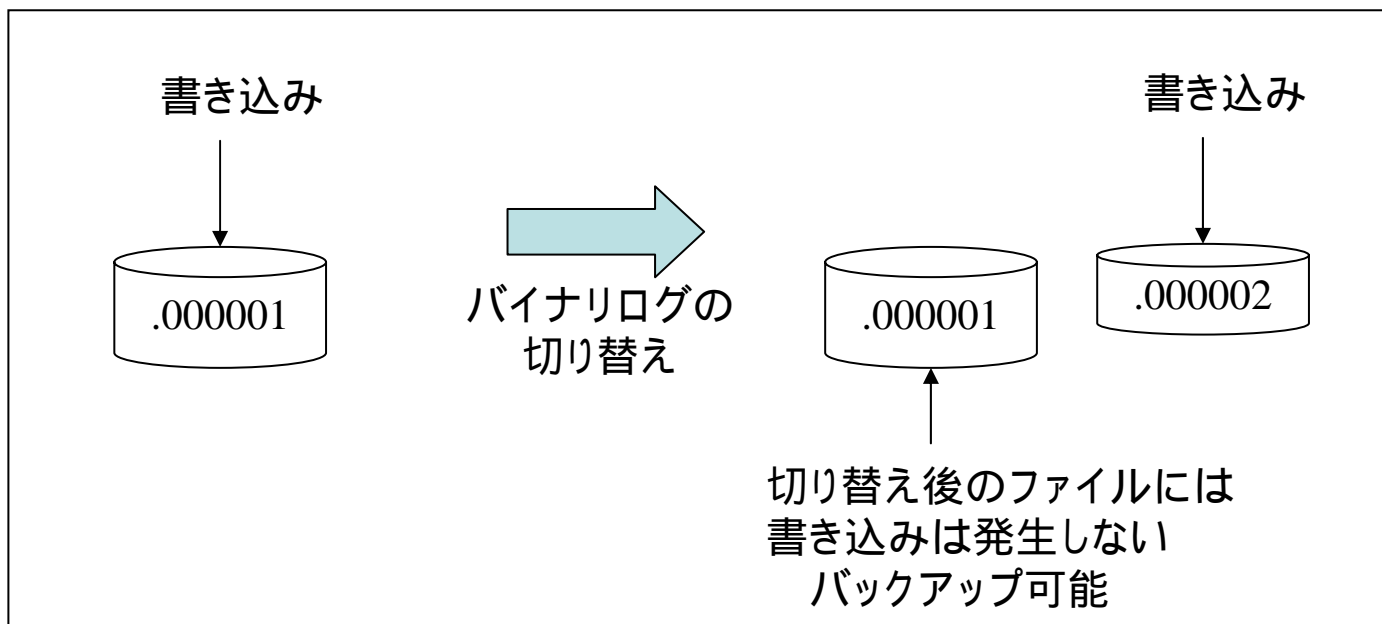
- ◆ デフォルトでは取得されない
- ◆ デフォルトではコミット時同期書き込みではない

```
[mysqld]
basedir=/usr/mysql4110
datadir=/data/mysql4110
innodb_data_home_dir=/data/mysql4110/idata
innodb_data_file_path= ibdata1:100M:autoextend
innodb_log_group_home_dir=/data/mysql4110/ilog
pid-file=/data/mysql4110/mysql-host.pid
socket=/data/mysql4110/mysql.sock
log-bin=/data/mysql4110/blog/mysql-host
innodb_safe_binlog
```

- ◆ ファイル名の拡張子は6桁の連番(4.0系では3桁) mysql-host.000001 ...
- ◆ log-binでバイナリログが有効になる
(ディレクトリ名とバイナリログの接頭辞を指定する)
- ◆ innodb_safe_binlogを指定するとコミット時同期書き込みになる

バイナリログ(3)

- バイナリログの切り替え
 - ◆ 新しくファイルを作成し、書き込み先を切り替える
 - ◆ 切り替え後のファイルには書き込みが発生しない
- 増分バックアップ相当の機能を実現可能
 - ◆ 切り替えた後のファイルをバックアップすることで可能



バイナリログ(4)

■ バイナリログが切り替わるタイミング

◆ 手動切り替え

- FLUSH LOGS; (SQL文)
- mysqladmin --flush-logs (OSコマンド)

◆ 自動切り替え

- mysqldの再起動時
- ファイルが一定サイズを超えた場合
 - (max_binlog_size: デフォルト1GB、最大1GB)

■ 基本事項の整理

- ◆ MySQLの機能概要
- ◆ メディア障害とリカバリの流れ

■ バイナリログ

■ コールドバックアップ・リカバリ

■ オンラインバックアップ・リカバリ

- ◆ mysqldump
- ◆ FLUSH TABLES WITH READ LOCK

■ その他のトピック

- ◆ メディア障害を考慮したファイル配置
- ◆ 過去の一時点へのリカバリ
- ◆ 内部的な動作

コールドバックアップ・リカバリ(1)

■ コールドバックアップとは

- ◆ mysqldプロセスを停止して取得するバックアップ
- ◆ OSコマンドによって取得する

1 . mysqldを停止

```
$mysqladmin shutdown --user=root --password=(パスワード) --socket=(ソケットファイル名)
```

2 . バックアップを取得

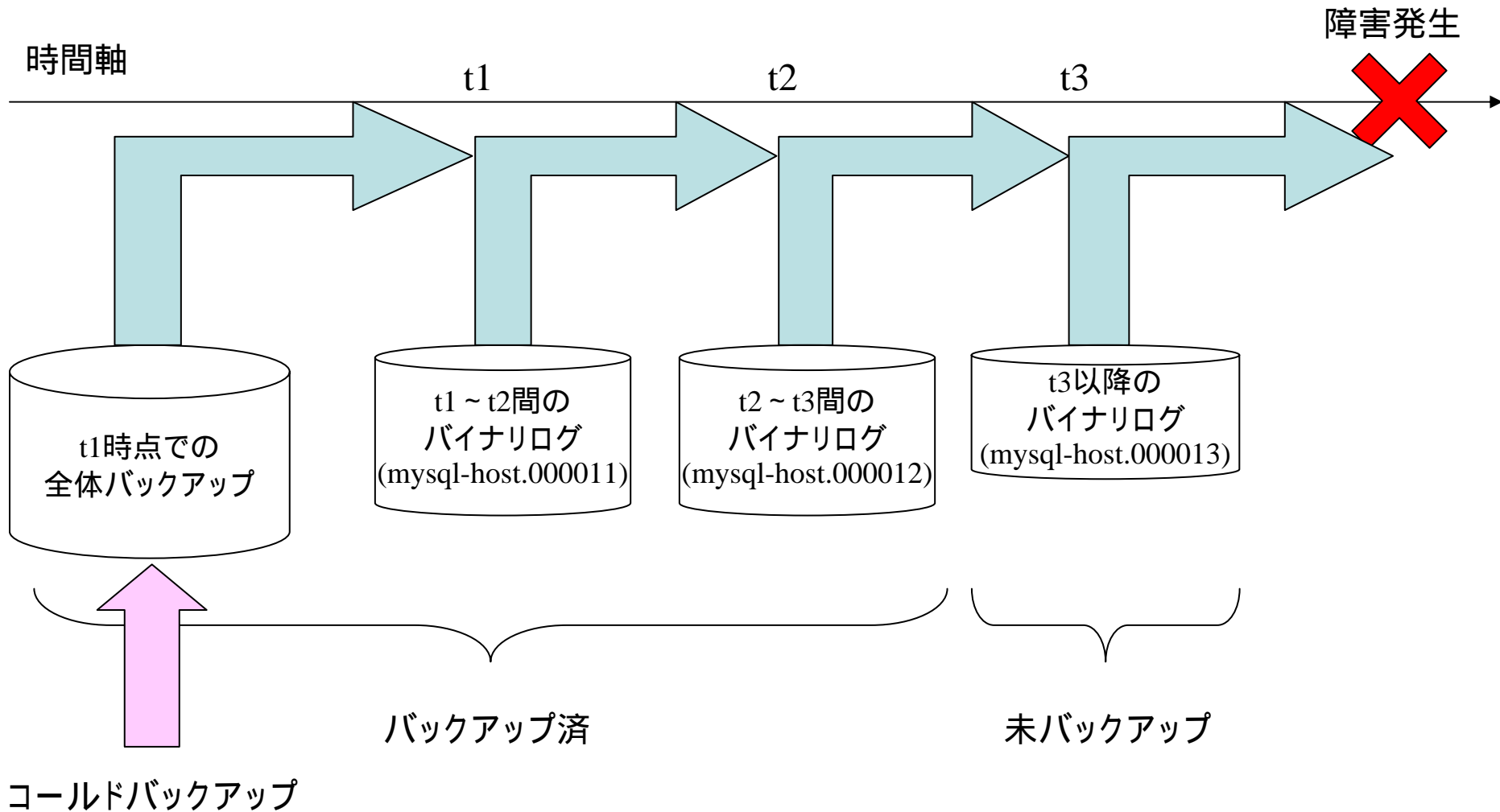
```
#cp -rp (データベース領域) (バックアップ先)
```

3 . mysqldを起動

```
$cd <MySQLインストールディレクトリ>
```

```
$/bin/mysqld_safe --defaults-file=(初期化パラメータファイル名)
```

コールドバックアップ・リカバリ(2)



コールドバックアップ・リカバリ(3)

■ リストア

- ◆ mysqldを停止した状態でリストアする

1. バックアップファイルをコピー
#cp -rp (バックアップ先) (データベース領域)

コールドバックアップ・リカバリ(4)

■ ロールフォワードリカバリ

- ◆ mysqlを起動した状態で行う(外部からの接続は遮断すべき)
- ◆ mysqlbinlogによってバイナリログをテキスト形式のSQL文に変換
- ◆ 適用の開始位置に注意
- ◆ 文字化けに注意

1. まだバックアップしていないバイナリログをバックアップする
#cp -p mysql-host.000013 (バックアップ先)

2. バイナリログの中身をテキスト形式のSQL文に変換
\$mysqlbinlog --disable-log-bin mysql-host.000011 mysql-host.000012
mysql-host.000013 > (ロールフォワード用SQLファイル名)

3. mysqlを起動
\$cd (MySQLインストールディレクトリ)
\$./bin/mysqld_safe --defaults-file=(初期化パラメータファイル名) --skip-networking

4. SQL文を適用
\$mysql --user=root --password=(パスワード) --socket=(ソケットファイル名)
--default-character-set=sjis < (ロールフォワード用SQLファイル名)

コールドバックアップ・リカバリ(5)

■ mysqldの再起動

◆ 初期化パラメータを正常に戻して再起動する

1 . mysqldを停止

```
$mysqladmin shutdown --user=root --password=(パスワード) --socket=(ソケットファイル名)
```

2 . mysqldを起動

```
$cd (MySQLインストールディレクトリ)
```

```
$/bin/mysqld_safe --defaults-file=(初期化パラメータファイル名)
```

■ 基本事項の整理

- ◆ MySQLの機能概要
- ◆ メディア障害とリカバリの流れ

■ バイナリログ

■ コールドバックアップ・リカバリ

■ オンラインバックアップ・リカバリ

- ◆ `mysqldump`
- ◆ FLUSH TABLES WITH READ LOCK

■ その他のトピック

- ◆ メディア障害を考慮したファイル配置
- ◆ 過去の一時点へのリカバリ
- ◆ 内部的な動作

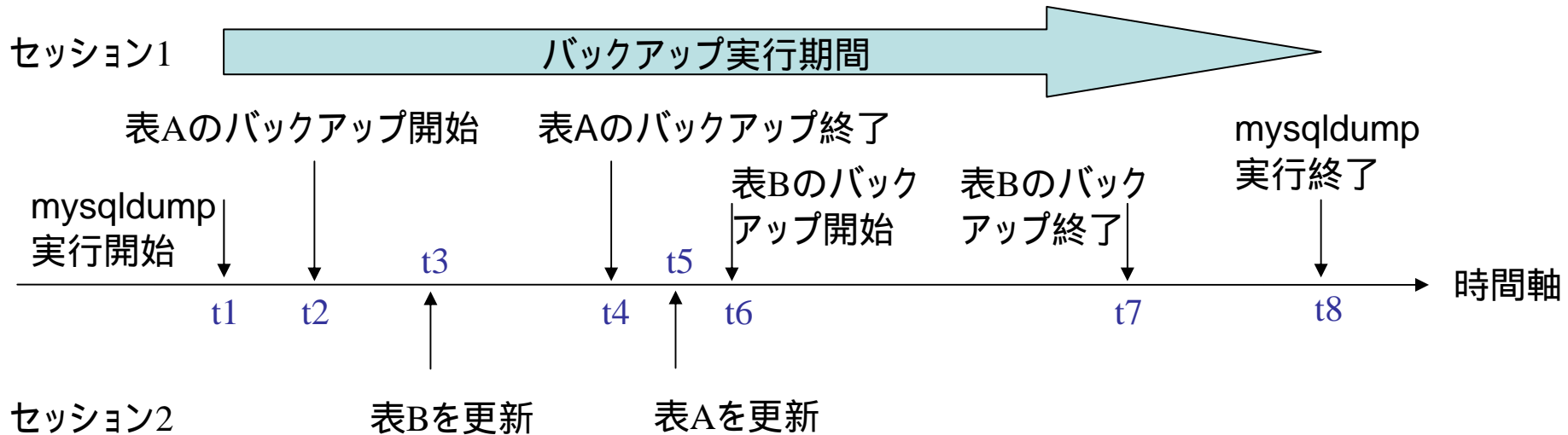
mysqldump(1)

- mysqldumpとは
 - ◆ MySQL標準のバックアップツール
 - ◆ バックアップ対象の全ての表に対してSELECT文を実行することで取得
 - ◆ バックアップ結果はSQL文の羅列
 - ◆ 引数の数が多く、それぞれが極めて重要
 - ◆ バージョンは4.1.11以降を推奨
 - ◆ InnoDB以外は、一貫性を保証するために、バックアップ中に共有ロックが必要
 - ◆ 部分バックアップも可能
 - ◆ 文字化けに注意

1. バックアップを取得

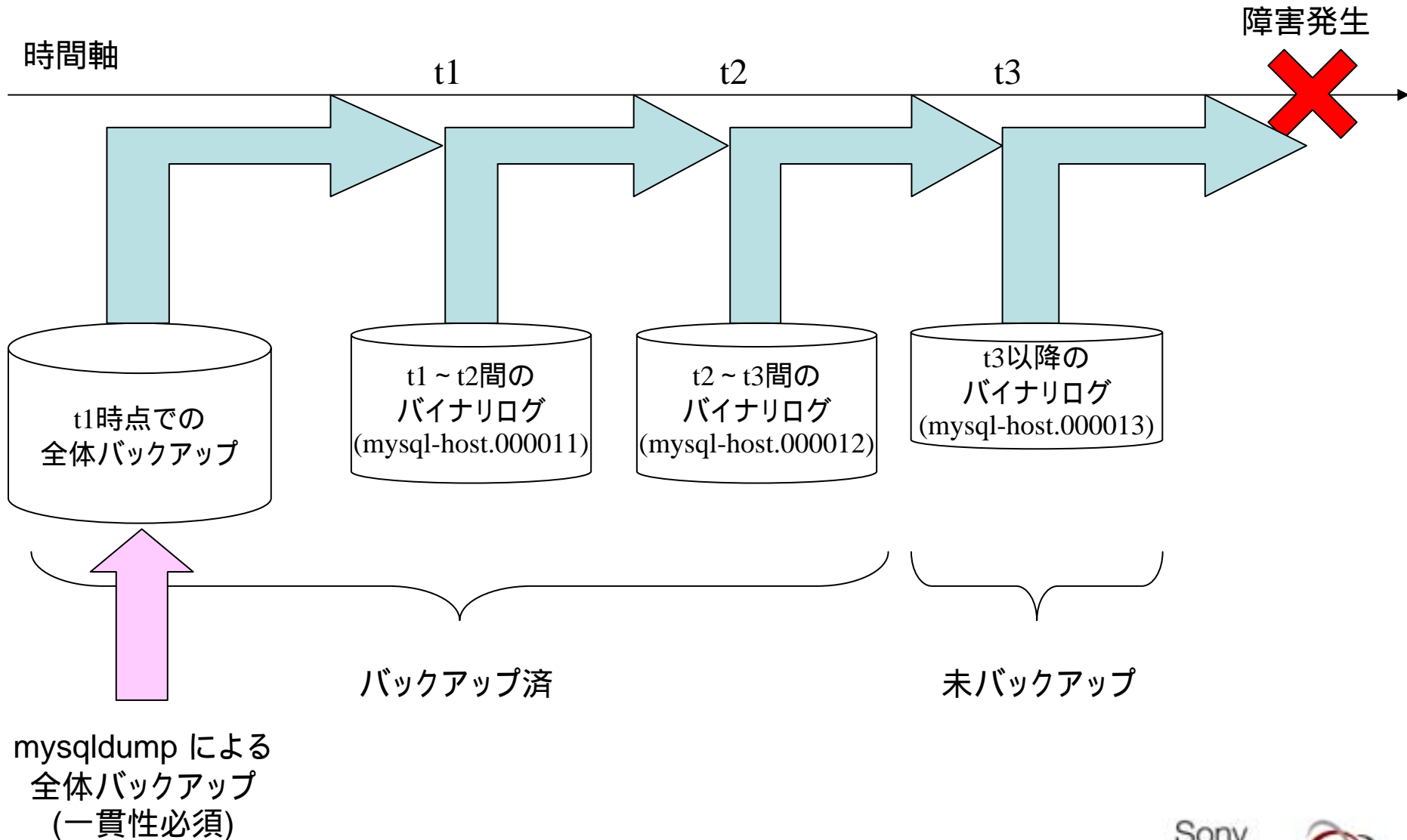
```
$mysqldump --user=root --password=(パスワード) --socket=(ソケットファイル名)  
--single-transaction  
--master-data  
--flush-logs  
--default-character-set=sjis  
--all-databases  
> (バックアップファイル名)
```

解説：一貫性のあるバックアップ



- バックアップデータは、いつの時点のものか？
 - ◆ MyISAM: SELECT文を実行した時刻にコミットされていたデータ
 - 表Aならt2、表Bならt6 **一貫性が無い**
 - ◆ InnoDB: START TRANSACTION WITH CONSISTENT SNAPSHOT文を実行した時刻にコミットされていたデータ
 - mysqlDumpならt1に実行するので、表A、表Bともt1 **一貫性がある**
- InnoDB以外をバックアップする場合、データベース全体に共有ロックをかけておかないと、一貫性のあるバックアップは取れない
 - ◆ mysqlDumpの引数--lock-all-tables

mysqldump(2)



mysqldump(3)

■ リストア

- ◆ mysqldを起動した状態で、SQL文を実行することでリストアする
- ◆ mysqlデータベースが消失している場合は、認証機能を無効にする必要がある

1. まだバックアップしていないバイナリログをバックアップ
#cp -p mysql-host.000013 (バックアップ先)

2. mysqldを起動
\$cd (MySQLインストールディレクトリ)
\$./bin/mysqld_safe --defaults-file=(初期化パラメータファイル名)
--skip-networking --skip-grant-tables

3. バックアップしておいたSQL文を実行
\$ mysql --user=root --password=(パスワード) --socket=(ソケットファイル名)
--default-character-set=sjis < (バックアップファイル名)

4. 作成されるファイルの一部を削除
\$ cd (データディレクトリ)
\$ rm master.info mysql-host-relay-bin.000001 mysql-host-relay-bin.index relay-log.info

mysqldump(4)

■ ロールフォワードリカバリ

- ◆ mysqldを起動した状態で行う
- ◆ mysqlbinlogによってバイナリログをテキスト形式のSQL文に変換
- ◆ 適用の開始位置に注意
- ◆ 文字化けに注意

1. バイナリログの中身をテキスト形式のSQL文に変換

```
$mysqlbinlog --disable-log-bin mysql-host.000011 mysql-host.000012  
mysql-host.000013 > (ロールフォワード用SQLファイル名)
```

2. SQL文を適用

```
$mysql --user=root --password=(パスワード) --socket=(ソケットファイル名)  
--default-character-set=sjis < (ロールフォワード用SQLファイル名)
```

- **基本事項の整理**
 - ◆ MySQLの機能概要
 - ◆ メディア障害とリカバリの流れ
- バイナリログ
- コールドバックアップ・リカバリ
- **オンラインバックアップ・リカバリ**
 - ◆ mysqldump
 - ◆ **FLUSH TABLES WITH READ LOCK**
- その他のトピック
 - ◆ メディア障害を考慮したファイル配置
 - ◆ 過去の一時点へのリカバリ
 - ◆ 内部的な動作

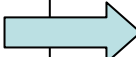
FLUSH TABLES WITH READ LOCK(1)

■ FLUSH TABLES WITH READ LOCKとは

- ◆ データベース全体を共有ロックする(=更新をブロックする)SQL文
- ◆ 共有ロックを解除するまで、全セッションから一切の更新が出来ない
 - 実行中のトランザクションは、以降の更新系SQL文の発行とCOMMITが出来ない
- ◆ MyISAM表データは全てディスクに書き込まれる
- ◆ バージョンは4.1.11以降を推奨

■ 「共有ロック 全体バックアップ 共有ロック解除」によって バックアップを取得できる

1. 共有ロックをかける
mysql> FLUSH TABLES WITH READ LOCK ;

2. バックアップを取得
#cp -rp (データベース領域) (バックアップ先) 

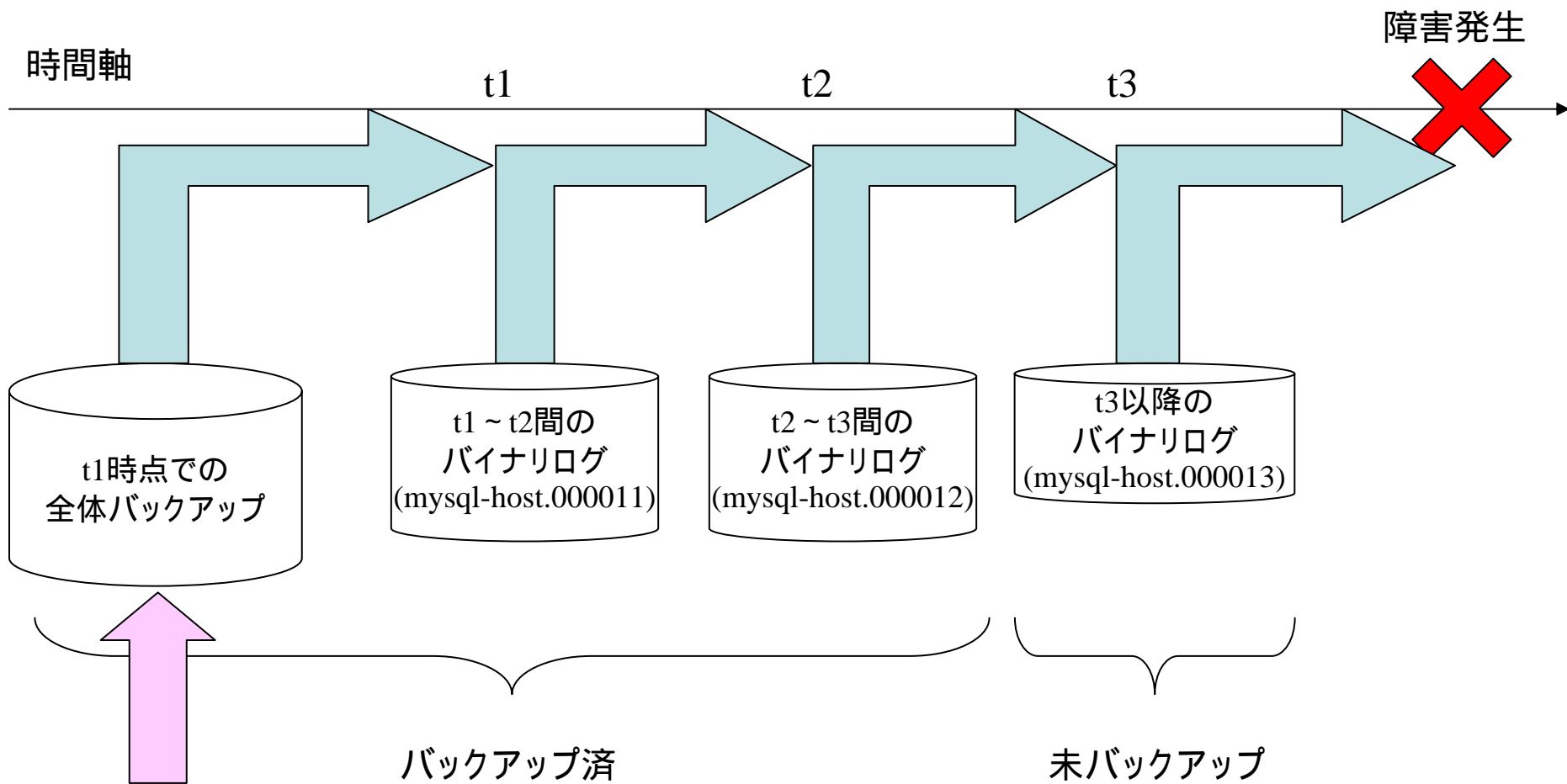
3. バイナリログを切り替える
mysql> FLUSH LOGS ;

4. 共有ロックを解除
mysql> UNLOCK TABLES ;

バックアップ時間をいかに短縮するかが鍵

一部のRAID対応バックアップソフトの
同期/スプリット機能や、
LVM、Win2003 VSS等の
スナップショット・バックアップ機能の活用

FLUSH TABLES WITH READ LOCK(3)



FLUSH TABLES WITH
READ LOCK による
全体バックアップ

■ リストア

- ◆ mysqldを停止した状態でリストアする
- ◆ リストアしたデータは、InnoDBデータファイルとInnoDBログファイルの同期が取れていないので、mysqld起動時にクラッシュリカバリが行われる

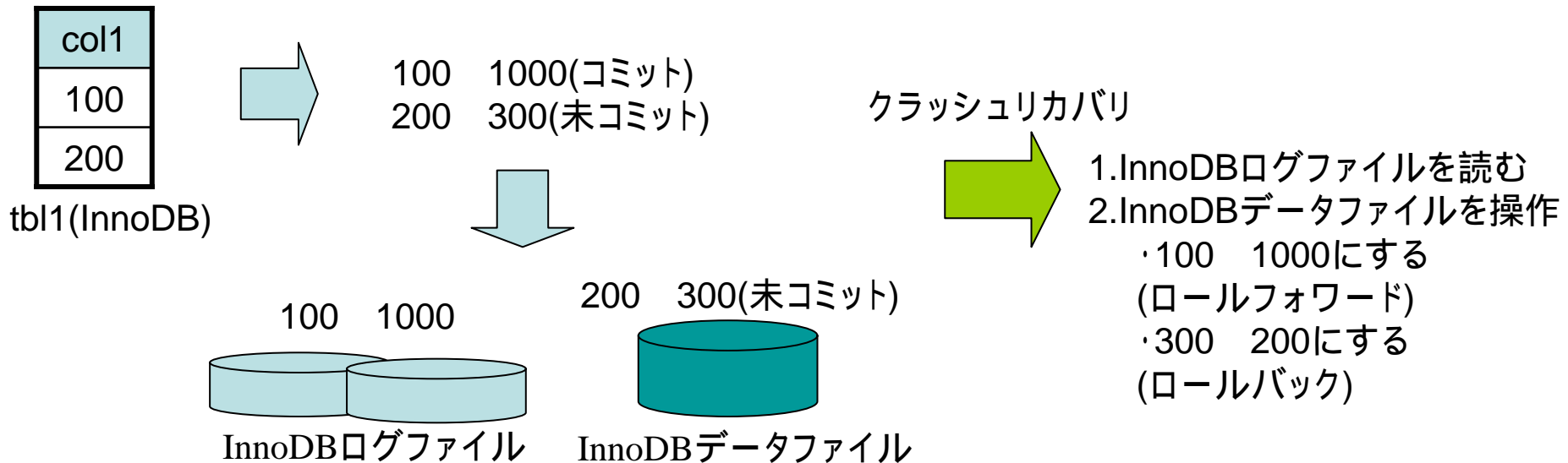
1. バックアップファイルをコピー
#cp -rp (バックアップ先) (データベース領域)

解説:クラッシュリカバリ

■ InnoDB表の回復処理

- ◆ mysql起動時に自動的に行われる、InnoDBログファイルとInnoDBデータファイルの同期をとる処理
- ◆ 同期が取れていない状態でmysqlを起動すると発生
 - 電源断などのmysql異常終了時
 - FLUSH TABLES WITH READ LOCK中に取得したバックアップを使う場合

■ InnoDB以外の表に対しては行われない



FLUSH TABLES WITH READ LOCK(5)

- ロールフォワードリカバリ(コールドバックアップの際と同様)
 - ◆ mysqlを起動した状態で行う(外部からの接続は遮断すべき)
 - ◆ mysqlbinlogによってバイナリログをテキスト形式のSQL文に変換
 - ◆ 適用の開始位置に注意
 - ◆ 文字化けに注意

1. まだバックアップしていないバイナリログをバックアップする
#cp -p mysql-host.000013 (バックアップ先)

2. バイナリログの中身をテキスト形式のSQL文に変換
\$mysqlbinlog --disable-log-bin mysql-host.000011 mysql-host.000012
mysql-host.000013 > (ロールフォワード用SQLファイル名)

3. mysqlを起動(クラッシュリカバリが自動的に発生)
\$cd (MySQLインストールディレクトリ)
\$./bin/mysql_safe --defaults-file=(初期化パラメータファイル名) --skip-networking

4. SQL文を適用
\$mysql --user=root --password=(パスワード) --socket=(ソケットファイル名)
--default-character-set=sjis < (ロールフォワード用SQLファイル名)

■ 基本事項の整理

- ◆ MySQLの機能概要
- ◆ メディア障害とリカバリの流れ

■ バイナリログ

■ コールドバックアップ・リカバリ

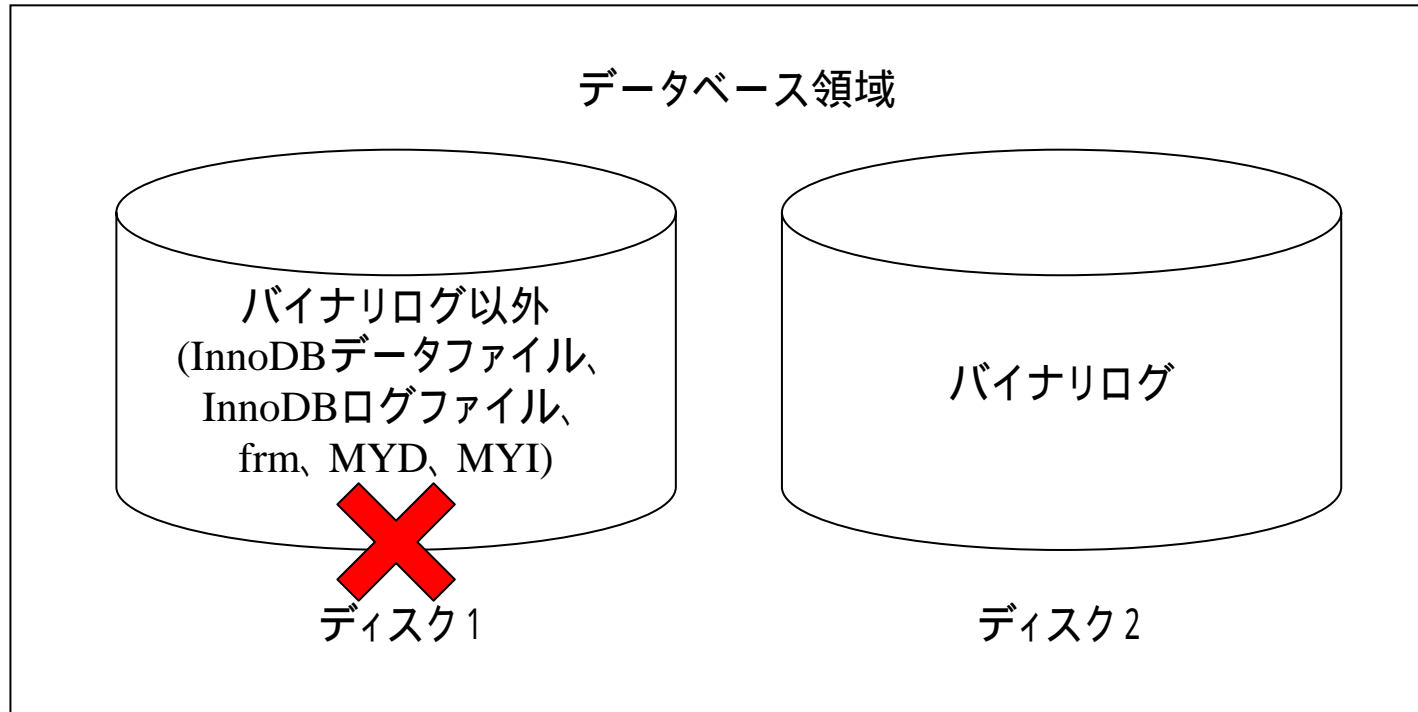
■ オンラインバックアップ・リカバリ

- ◆ mysqldump
- ◆ FLUSH TABLES WITH READ LOCK

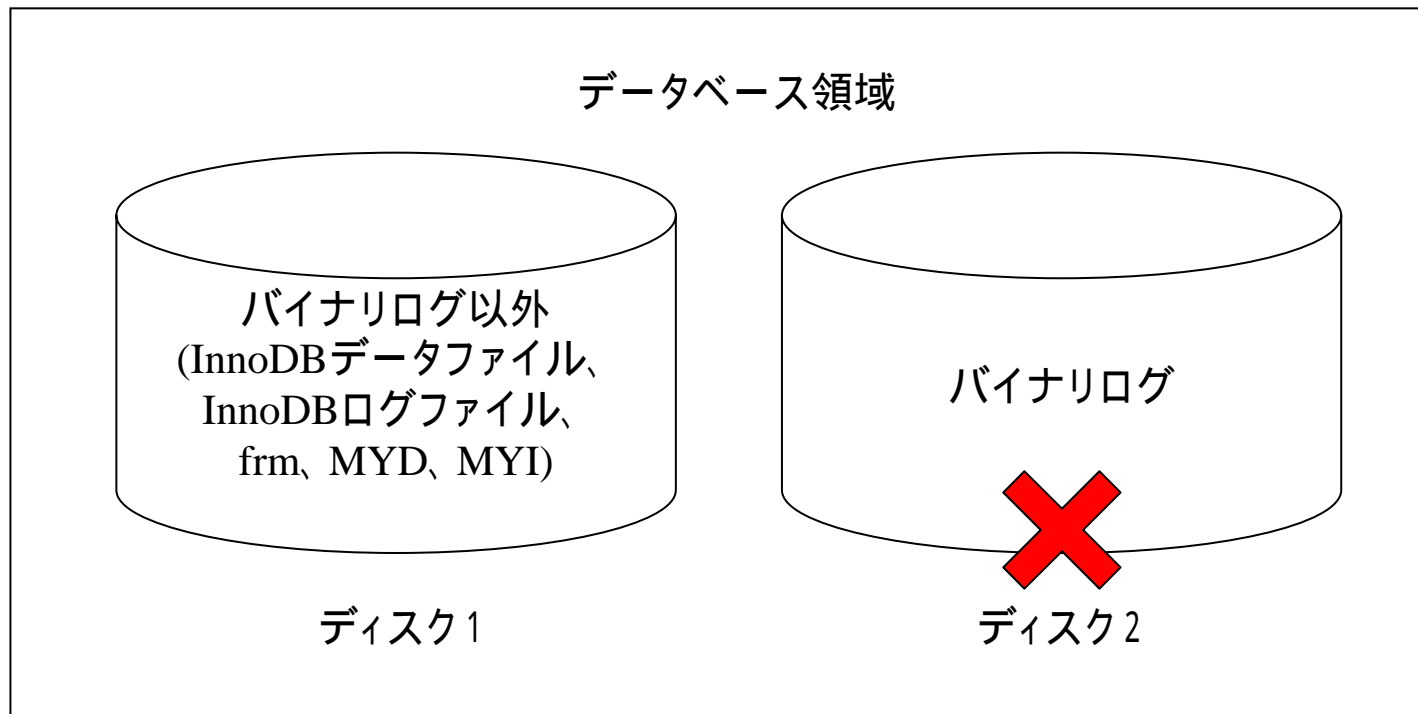
■ その他のトピック

- ◆ メディア障害を考慮したファイル配置
- ◆ 過去の一時点へのリカバリ
- ◆ 内部的な動作

メディア障害を考慮したファイル配置(1)



バイナリログが全て無事であれば、全体バックアップと
バイナリログによって、最新の状態にロールフォワードリカバリできる。



バイナリログ以外が無事であれば、データの消失は起こらない。
ロールフォワードリカバリも不要。

更新はできなくなる。ただし正常シャットダウンが可能なので、
停止した上で全体バックアップを取得すれば良い。

過去の一時点へのリカバリ

- mysqlbinlogの引数--stop-datetimeによって、バイナリログの変換範囲を指定できる
 - ◆ DATETIME/TIMESTAMP型のフォーマットを指定
 - ◆ 指定した時刻までのバイナリログがSQL文に変換される
 - 指定した時刻までロールフォワードリカバリできる

3/26の13:30までのバイナリログをSQL文に変換

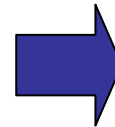
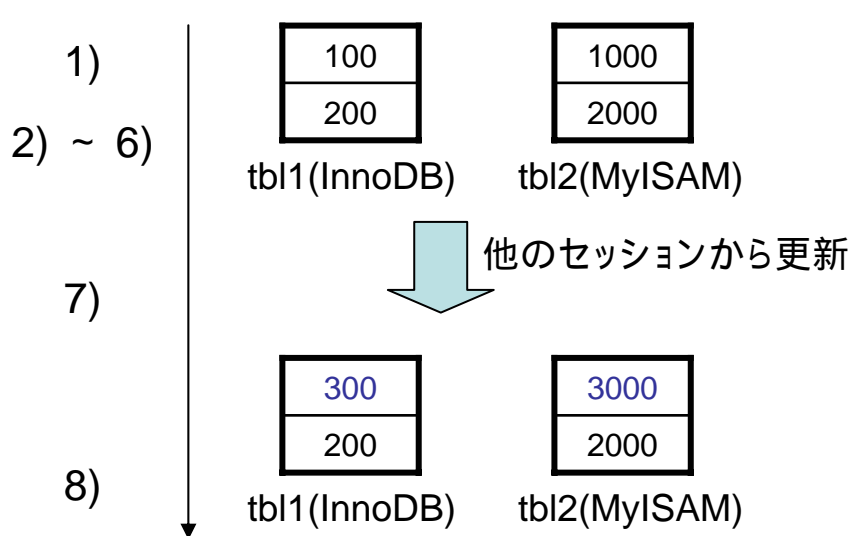
```
$ mysqlbinlog --stop-datetime="2005-03-26 13:30:00" (バイナリログファイル名)
```

mysqldumpの動作

```
$ mysqldump --single-transaction --master-data
--flush-logs --all-databases
```

- 1) mysqldump実行開始
- 2) FLUSH TABLES WITH READ LOCK;
- 3) SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
- 4) START TRANSACTION WITH CONSISTENT SNAPSHOT;
- 5) FLUSH LOGS;
- 6) UNLOCK TABLES;
- 7) 全ての表に対してSELECT文を実行
- 8) mysqldump実行終了

2) ~ 6)の間は更新不可
(通常は一瞬で終わる)



バックアップデータの中身

100
200

3000
2000

tbl1(InnoDB) tbl2(MyISAM)

- ・InnoDBは2)~6)時点でコミットされていた行データを選んでバックアップする。また、5)以降のバイナリログを適用してロールフォワードリカバリが可能。
 - ・MyISAM他は7)のSELECT文実行時点でコミットされていた行データをバックアップする。
- 全体で見れば一貫性が保証されない。また、どこからロールフォワードリカバリすれば良いか特定できない。

FLUSH TABLES WITH READ LOCKの動作

1. グローバルな共有ロックを確保(以降の更新SQL文を全てブロック)
2. 未終了の更新SQL文を待つ(SQL文レベル。トランザクションレベルではない)
3. COMMIT文をブロックするためのロックを確保
 タイムアウトはしない(innodb_lock_wait_timeoutパラメータは機能しない)
 3の部分で不具合があり、4.1.11で解決予定

col1
100
200

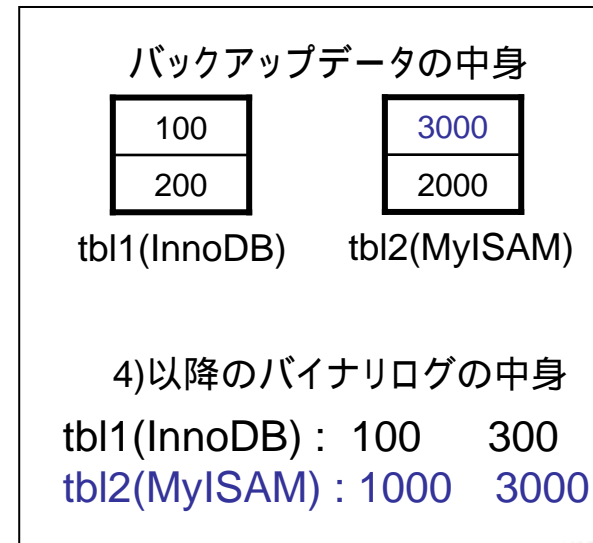
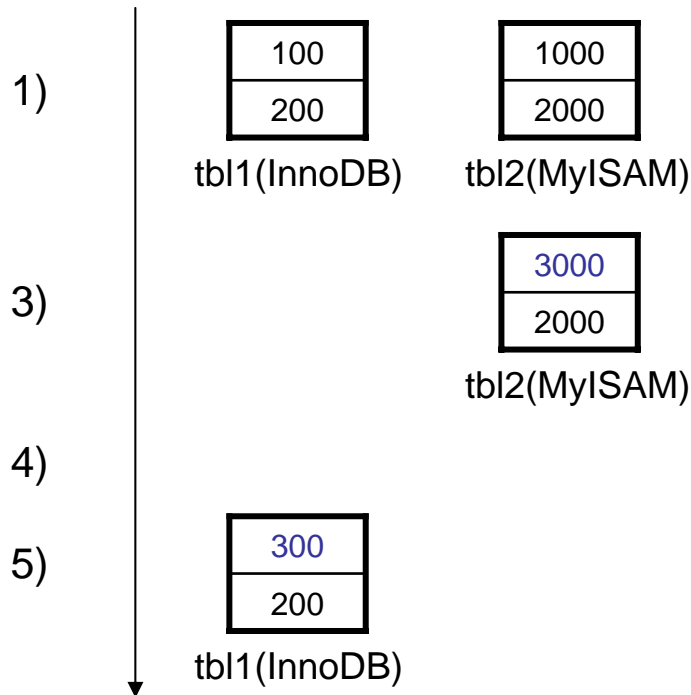
tbl1(InnoDB)

	トランザクション1	トランザクション2	トランザクション3	トランザクション4
1	START TRANSACTION;	START TRANSACTION;		
2	UPDATE tbl1 SET col1=300;			
3		UPDATE tbl1 SET col1=400; (行ロックによる待ち)		
4			FLUSH TABLES WITH READ LOCK; (1段階目まで終了し、2段階目で待つ)	
5				CREATE TABLE tbl2 (col1 INTEGER); (ロック待ち)
6.1	COMMIT;			
6.2		UPDATE文の実行終了(未コミット)		
6.3			FLUSH TABLES WITH READ LOCK の実行終了	
7		COMMIT; (ロック待ち)		

バイナリログの不具合(5.1で解決予定)

- 同一トランザクション内で、InnoDB表を更新した後にMyISAM表を更新すると、MyISAMへの更新結果が、バイナリログに即時反映(コミット時同期書込)されない。

- 1) START TRANSACTION;
- 2) InnoDB表を更新
- 3) MyISAM表を更新
- 4)--ここで全体バックアップ取得
- 5) COMMIT;



まとめ(1)

■ バックアップの手段

- ◆ コールドバックアップ、オンラインバックアップ、増分バックアップが可能
- ◆ オンラインバックアップには様々な方法がある

■ ロールフォワードリカバリ

- ◆ バイナリログを有効にすることで可能(デフォルトでは無効)
- ◆ 4.1以降では文字コードの指定に注意

■ ファイル配置

- ◆ バイナリログとそれ以外でディスクを分けることを検討すると良い

■ アプリケーション処理

- ◆ 同一トランザクション内でInnoDB表とMyISAM表を同時に更新することは避ける

まとめ(2) オンラインバックアップの手法比較

バックアップ方法	対象ストレージエンジン	特徴、特記事項
mysqldump	全て(条件付)	<p>バックアップに時間がかかる</p> <p>リストアに時間がかかる</p> <p>InnoDB以外は、バックアップ中に共有ロックをかけないと一貫性を保証できない</p> <p>ロックをかける期間は実質皆無に近い</p> <p>バグのため、バージョン4.1.11以降を推奨</p>
FLUSH TABLES WITH READ LOCK +OSコマンドでコピー	全て	<p>即時バックアップ機能がほぼ前提</p> <p>バックアップ取得時間が短い</p> <p>ロックをかける期間は実質皆無に近い</p> <p>バグのため、バージョン4.1.11以降を推奨</p>
InnoDB Hot Backup (参考)	原則InnoDBのみ	<p>バックアップに時間がかかる</p> <p>有料</p> <p>frmファイルのバックアップは別途行う必要がある</p> <p>ロックはかけない</p>
非同期レプリケーション (参考)	全て	<p>スレーブ機が別途必要 コスト増につながる</p> <p>ロックはかけない</p>

- 翔泳社 DB Magazine 「本気で取り組むMySQL入門」
2005年6月号、7月号 (バックアップ・リカバリ前編、後編)
 - ◆ 前編: コールドバックアップ、バイナリログ、増分バックアップ
 - ◆ 後編: オンラインバックアップ

- MySQLのマニュアル
 - ◆ The mysqldump Database Backup Program
 - <http://dev.mysql.com/doc/mysql/en/mysqldump.html>
 - 日本語の情報はバージョンが古いので注意
 - ◆ InnoDB Hot Backup Online Manual
 - <http://www.innodb.com/manual.php>

- オライリー「実践ハイパフォーマンスMySQL」
 - ◆ レプリケーションの解説が深い。

- MySQLメーリングリスト「クラッシュからのリカバリ」(翻訳記事)
 - ◆ <http://www.mysql.gr.jp/mysqlml/mysql/msg/10758>