MySQLのプロトコル解説

とみたまさひろ

日本MySQLユーザ会

MyNA会

2013/07/29

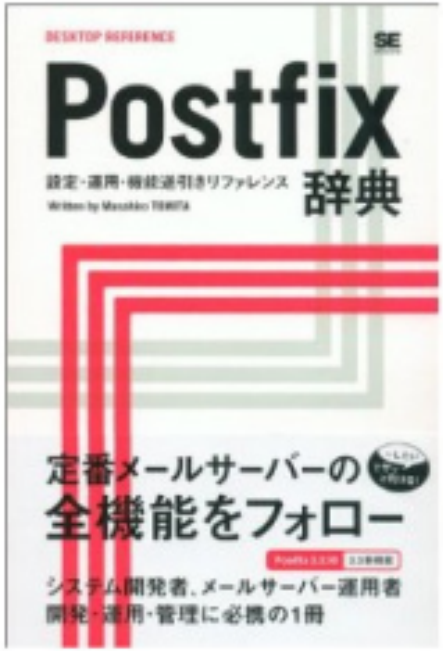
自己紹介

- とみた まさひろ
  - MySQLユーザ会(名ばかり代表)
  - 長野県北部在住
  - プログラマー (Ruby & C)
  - http://**tmtms**.hatenablog.com
  - http://twitter.com/**tmtms**
  - https://github.com/**tmtm**/ruby-mysql

趣味

- 誰も使わないRubyライブラリを作ったり
- MySQL の yacc ファイル読んだり
- マンガ読んだり
  - 聖悠紀 / 佐々木淳子 / 岡崎二郎 / 星野之宣 / 長谷川裕一 / 島本和彦 / 桑田乃梨子 / etc
- 技術書の積読

昔こんな本を書きました



OSS貢献者賞


第8回 日本OSS貢献者賞
とみた まさひろ 殿
貴殿は 優れた能力を発揮し国内外の
OSS（オープンソースソフトウェア）の
普及・促進に多大な貢献をされました
よってその功績を称えここに表彰
いたします
2013年2月22日
日本OSS推進フォーラム

本日の誰得枠

MySQLのプロトコル解説

MySQLプロトコルを理解すると何がうれしいの？

言語ネイティブなライブラリを作れる(かも)

- libmysqlclient に依存しない
- Cの制約に縛られない
- プリペアドステートメントの C API は複雑

Rubyの場合

libmysqlclient 制御下では

- 他のスレッドが止まる
  (やり方はあるみたいだけど知らない)
- GC が動かない

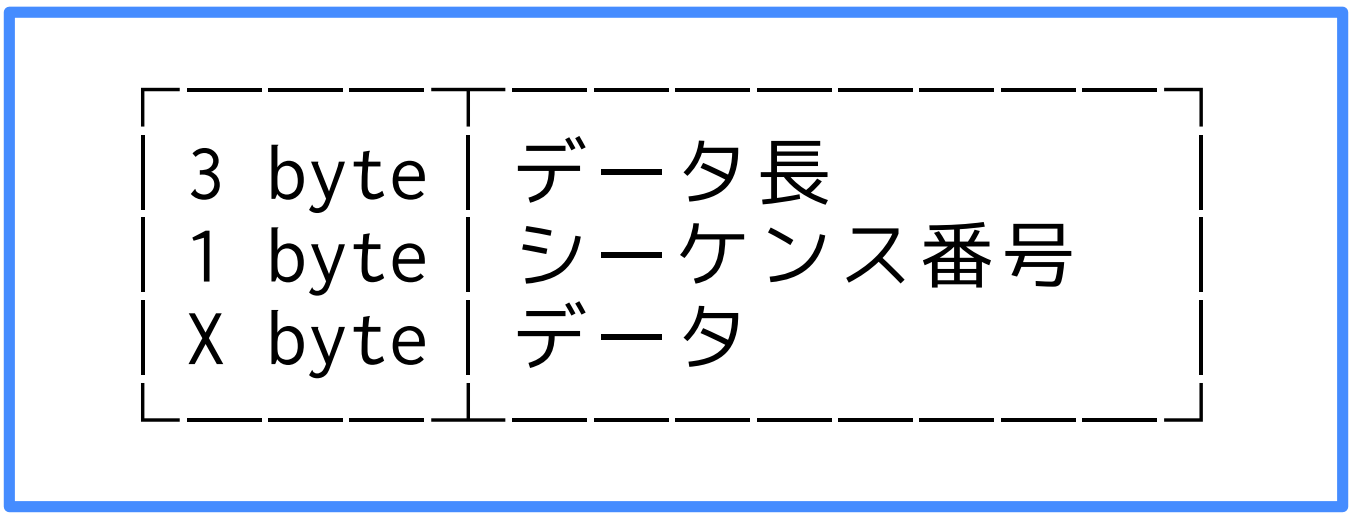
MySQLパケットを中継するプログラムを作れる(かも)

- MySQL Proxy 風の何か

ということでMySQLプロトコルの解説

Ruby/MySQL を作った時に調べた知識に基づいてます。嘘書いてるかもしれません。

パケット



数値は基本的に Little Endian

データ長

- 0〜0xFFFFFF(約16MB)
- 0xFFFFFF の場合は継続あり。次のパケットのデータも結合する。

シーケンス番号

コマンド発行毎に 0 から始まり、read, write 毎に1ずつ増加。255 までいったらまた 0 に戻る。

可変長整数(VLI)

0x00〜0xFA(0〜250)	そのまま
0xFB	NULL
0xFC XX XX	2バイトの正の整数
0xFD XX XX XX	3バイトの正の整数
0xFE XX XX XX XX XX XX XX XX	8バイトの正の整数

長さつき文字列(LS)

文字列長(VLI) + 文字列のバイト列

接続

初期パケット
(サーバー→クライアント)

1 byte	プロトコルバージョン (5.6.10 では 10) (uchar)
X byte	サーバーバージョン (e.g. "5.6.10") (NUL終端文字列)
4 byte	スレッドID (ulong)
8 byte	パスワードハッシュ化のためのキー①
1 byte	0x00
2 byte	Capability①
1 byte	chraset (uchar)
2 byte	ステータス (ushort)
2 byte	Capability② (5.5以降？)
1 byte	パスワードハッシュ化のためのキー①+②の長さ+1 (5.5以降？)
10 byte	0x00
X byte	パスワードハッシュ化のためのキー② (12 byte)
1 byte	0x00
X byte	プラグイン名 "mysql_native_password" (5.5以降？)

パスワードのハッシュ化 (mysql_native_password)

- パスワードが与えられない場合は空文字列
- 平文を SHA1 でハッシュ化 … A
- A をさらに SHA1 でハッシュ化 … B
- パスワードハッシュ化のためのキー①+②+B を SHA1 でハッシュ化 … C
- A と C を XOR した文字列

認証用パケット（クライアント→サーバー）

4 byte	クライアントフラグ (ulong)
4 byte	最大パケット長 (ulong)
X byte	charset (VLI)
23 byte	00
X byte	ユーザー名 (NUL終端文字列)
X byte	ハッシュ化されたパスワード (LS)
X byte	データベース名 (NUL終端文字列)

クライアントフラグ

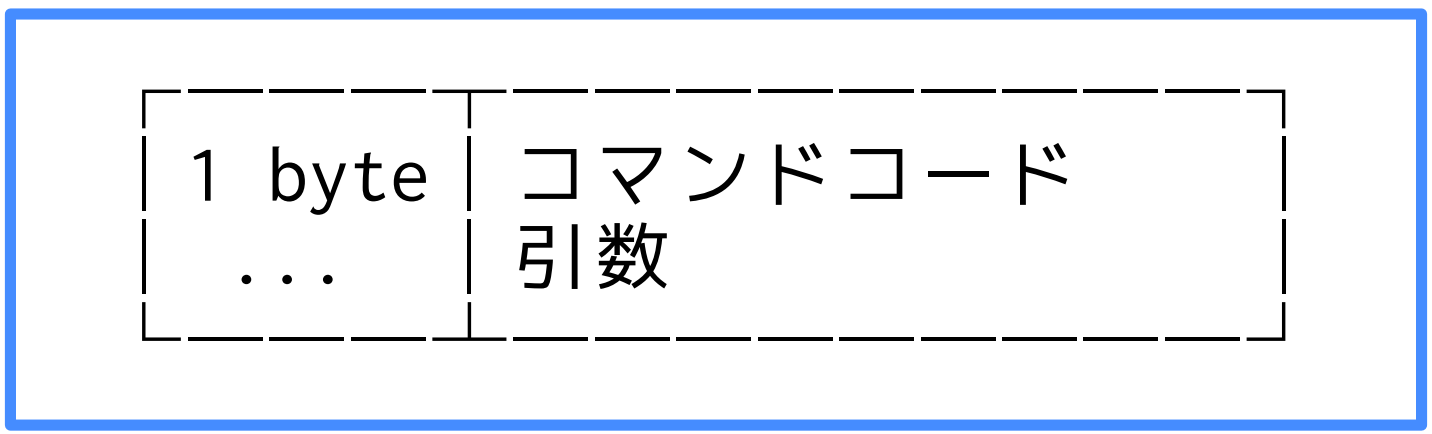
0x00001	長いパスワード
0x00002	更新した行数ではなく条件に一致した行数を返す
0x00004	全カラムフラグ
0x00008	DB名指定
0x00010	DB名.テーブル名.カラム名 指定を許可しない
0x00020	圧縮プロトコル
0x00080	LOAD DATA LOCAL INFILE を許可
0x00100	関数名と'('の間の空白を無視する
0x00200	4.1プロトコル
0x00400	wait_timeout の代わりに interactive_timeout を使う
0x02000	トランザクションあり
0x00800	SSL プロトコル
0x08000	4.1認証
0x10000	';' 区切りで複数のクエリを指定可能
0x20000	複数クエリの結果を返す

Charset & Collation

#	charset	collation
11	ascii	ascii_general_ci
65	ascii	ascii_bin
33	utf8	utf8_general_ci
83	utf8	utf8_bin
45	utf8mb4	utf8mb4_general_ci
46	utf8mb4	utf8mb4_bin
95	cp932	cp932_japanese_ci
96	cp932	cp932_bin
97	eucjpms	eucjpms_japanese_ci
98	eucjpms	eucjpms_bin
63	binary	binary

コマンド

コマンドパケット
(クライアント→サーバー)



コマンド

0x01	切断
0x02	DB選択
0x03	クエリ
0x04	フィールドリスト
0x05	DB作成
0x06	DB破棄
0x07	リフレッシュ
0x08	シャットダウン
0x09	Statistics
0x0A	プロセス情報
0x0C	Kill
0x0E	Ping
0x11	ユーザー変更
0x1B	サーバーオプション設定

コマンド
(プリペアドステートメント)

0x16	作成
0x17	実行
0x18	巨大データの送信
0x19	クローズ
0x1A	リセット
0x1C	結果取り出し

応答パケット
(サーバー→クライアント)

コマンドによって異なる

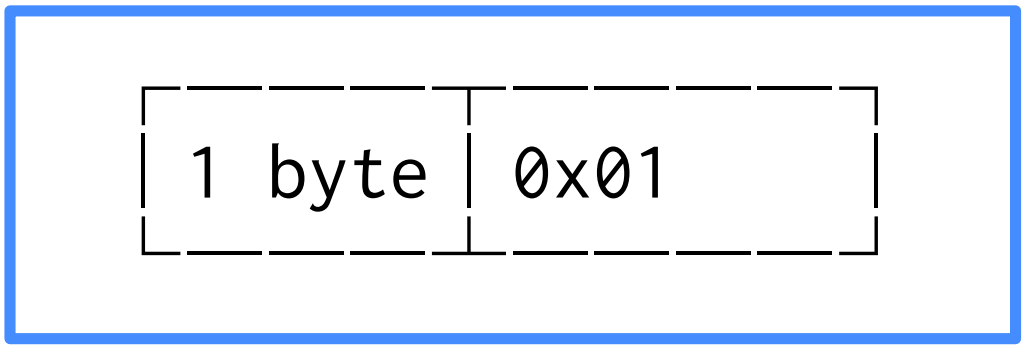
エラーパケット
(サーバー→クライアント)

先頭が 0xFF の応答パケットはエラーパケット

1 byte	0xFF
2 byte	エラー番号(ushort)
1 byte	0x23(#)
5 byte	SQLSTATE
X byte	エラーメッセージ

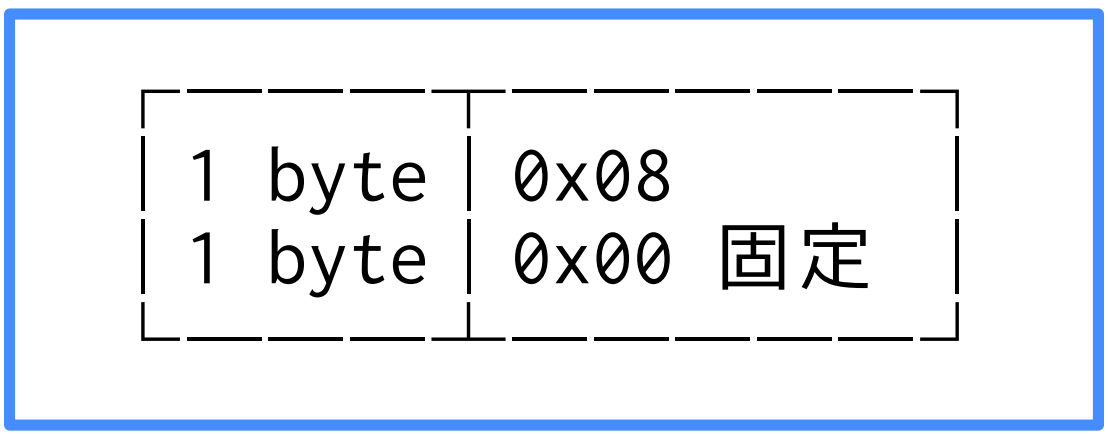
引数なしコマンド

QUITパケット
（クライアント→サーバー）



引数ありコマンド

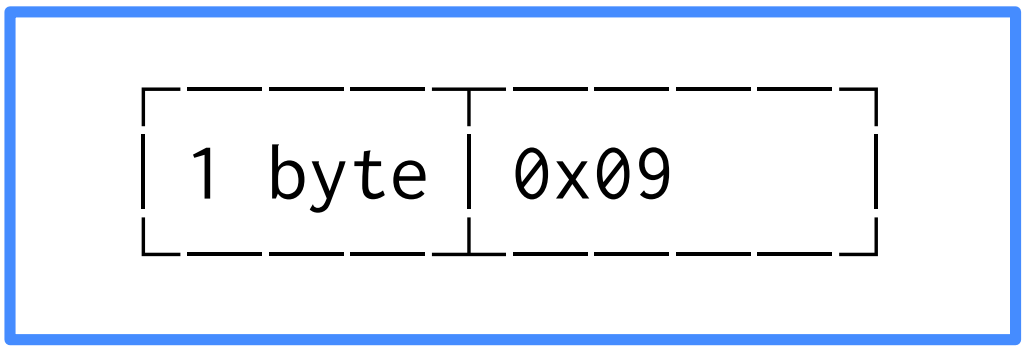
シャットダウンパケット (クライアント→サーバー)



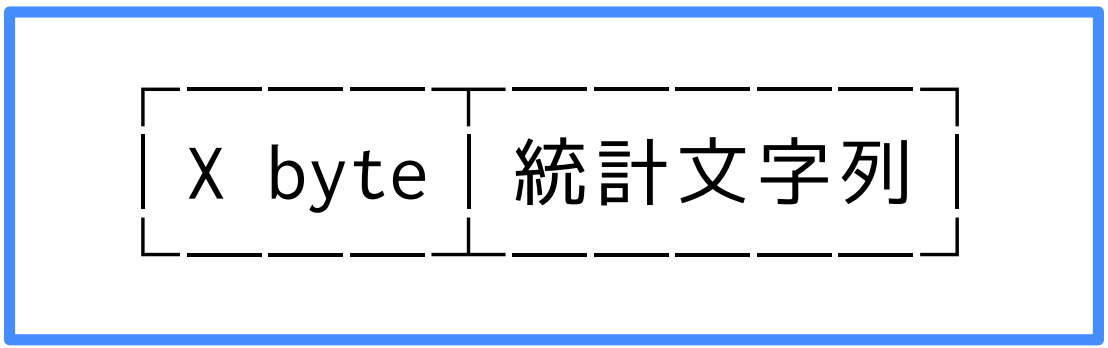
1 byte	0x08
1 byte	0x00 固定

応答ありコマンド

Statisticsパケット (クライアント→サーバー)



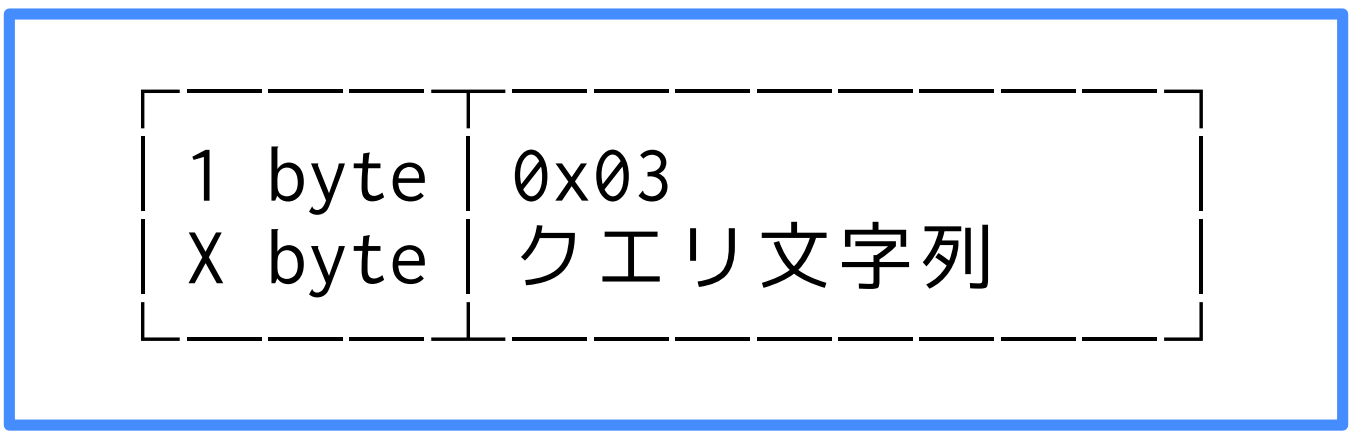
Statistics応答パケット (サーバー→クライアント)



"Uptime: 7314 Threads: 1 Questions: 4 Slow queries: 0 Opens: 67
Flush tables: 1 Open tables: 60 Queries per second avg: 0.000"

クエリ

クエリパケット
（クライアント→サーバー）



1 byte	0x03
X byte	クエリ文字列

クエリ応答パケット（サーバー→クライアント）

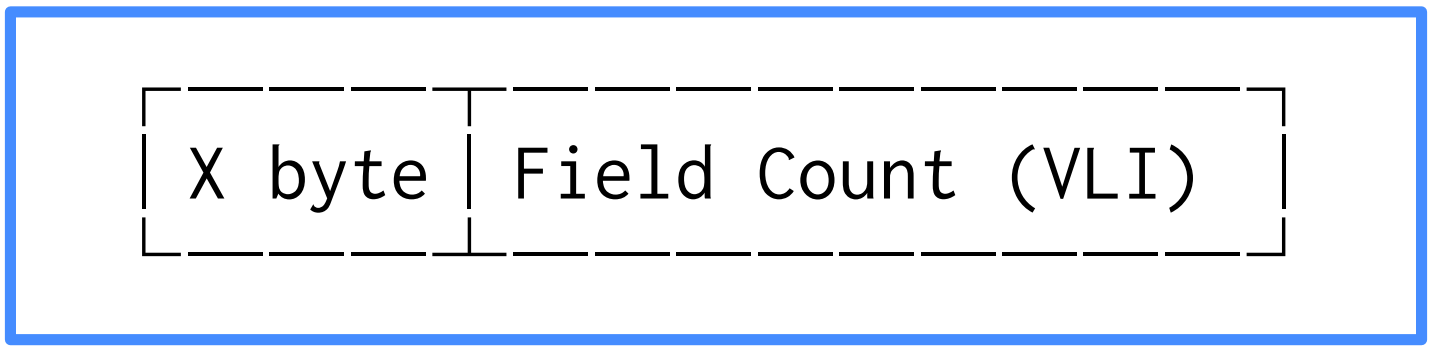
3種類

- 結果セットなし(UPDATE, INSERT 等)
- 結果セットあり(SELECT 等)
- LOAD DATA LOCAL INFILE

クエリ応答パケット (結果セットなし)

X byte	0 (VLI)
X byte	Affected Rows (VLI)
X byte	Insert ID (VLI)
2 byte	Server Status (ushort)
2 byte	Warning Count (ushort)
X byte	Message (LS)

クエリ応答パケット
(結果セットあり)



結果セット

- フィールドパケット * フィールド数
- レコードパケット * レコード数
- EOFパケット

フィールドパケット

4 byte	"def" (LS)
X byte	データベース名 (LS)
X byte	テーブル名 (LS)
X byte	オリジナルテーブル名 (LS)
X byte	カラム名 (LS)
X byte	オリジナルカラム名(LS)
1 byte	12
2 byte	charset (ushort)
4 byte	長さ (ulong)
1 byte	型 (uchar)
2 byte	フラグ (ushort)
1 byte	少数桁数 (uchar)
2 byte	00
X byte	デフォルト値 (LS) (フィールドリストコマンド時)

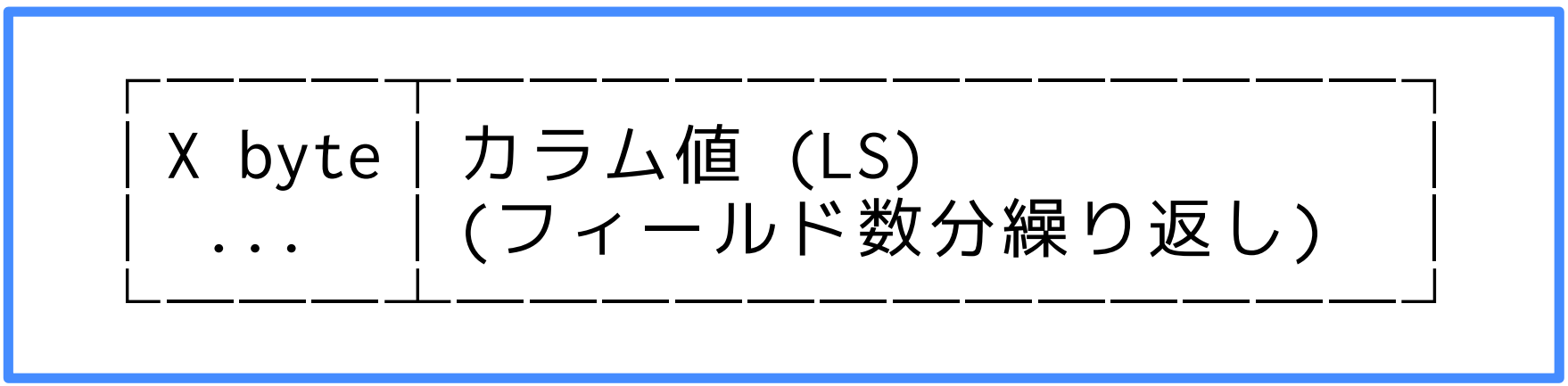
フィールドの型

TINYINT	0x01	TIME	0x0B
SMALLINT	0x02	YEAR	0x0D
MEDIUMINT	0x09	TIMESTAMP	0x07
INT	0x03	CHAR	0xFE
BIGINT	0x08	VARCHAR	0xFD
FLOAT	0x04	BLOB	0xFC
DOUBLE	0x05	BIT	0x10
DECIMAL	0xF6	ENUM	0xF7(0xFE?)
DATETIME	0x0C	SET	0xF8(0xFE?)
DATE	0x0A	GEOMETRY	0xFF

フィールドフラグ

NOT NULL	0x0001
PRIMARY KEY	0x0002
UNIQUE	0x0004
INDEX	0x0008
BLOB	0x0010
UNSIGNED	0x0020
ZEROFILL	0x0040
BINARY	0x0080
AUTO_INCREMENT	0x0200
ENUM	0x0100
SET	0x0800
NO DEFAULT VALUE	0x1000

レコードパケット

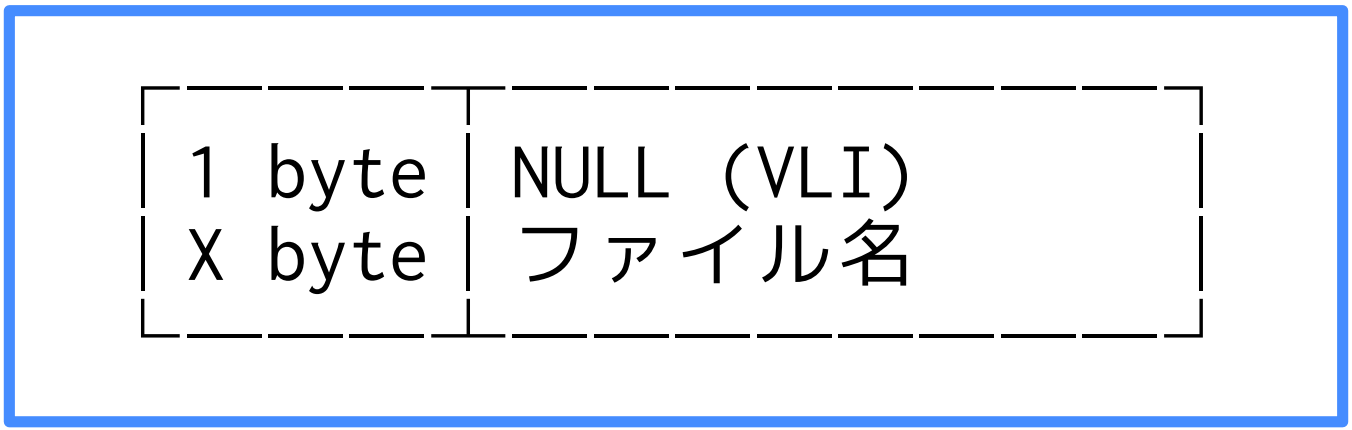


EOFパケット

- 0xFE で始まる8バイト以内のパケット

1 byte	0xFE
2 byte	Warning Count
2 byte	Server Status

クエリ応答パケット (LOAD DATA LOCAL INFILE)



1 byte	NULL (VLI)
X byte	ファイル名

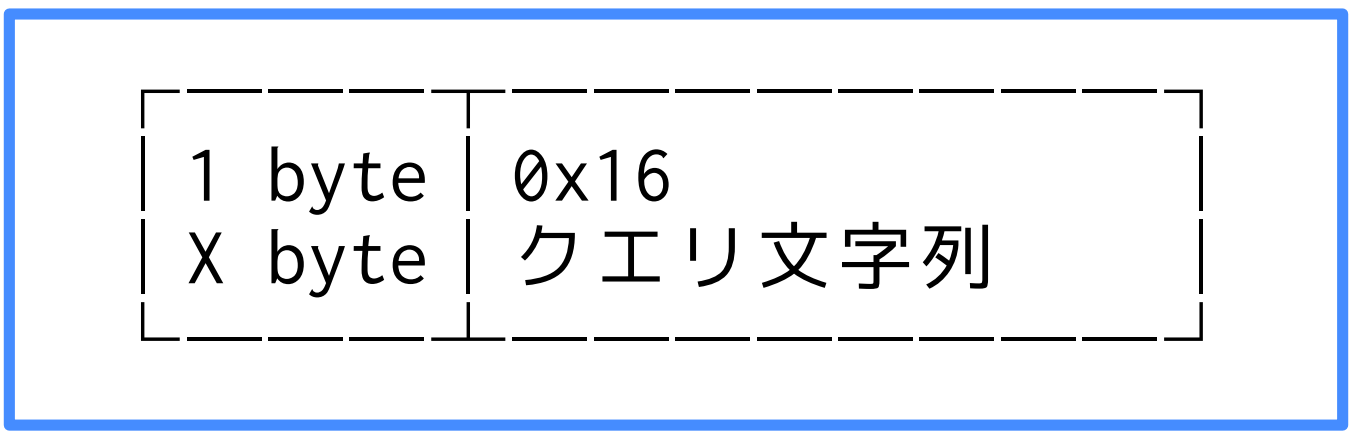
このパケットを受け取った後、クライアントはローカルファイルのデータをサーバーに送信する。

LOAD DATA LOCAL INFILE

```
C→S  LOAD DATA LOCAL INFILE 'filename.tsv'
     INTO TABLE ...

C←S  'filename.tsv'

C→S  filename.tsv の内容

C→S  EOF(長さ0のパケット)
```

プリペアドステートメント

クエリ準備パケット（クライアント→サーバー）



クエリ準備結果パケット (サーバー→クライアント)

4 byte	Statement ID (ulong)
2 byte	Field Count (ushort)
2 byte	Parameter Count (ushort)
1 byte	0x00
2 byte	Warning Count (ushort)

クエリ実行パケット (クライアント→サーバー)

1 byte	0x17
4 byte	Statement ID (ulong)
1 byte	Cursor type (uchar)
4 byte	1 (ulong)
X byte	NULL Bitmap
1 byte	1 (uchar)
X byte	パラメータ値の型
X byte	パラメータ値

NULL Bitmap

- パラメータの値のうち NULL のものを 1、NULL 以外を 0 で表したビットマップ
- (パラメータ数-1)/8+1 バイト（ただしパラメータがない場合は 0 バイト)
- 例: パラメータが (1, nil, 2, 3, nil) の場合 0x12(0001_0010)

パラメータ値の型

- 2 byte * パラメータ数(NULLを除く)
- フィールドの型の2バイト表現
- 符号なし整数値は 0x8000 を加算

  例) 符号なし2byte整数: 0x8002

パラメータ値

値	データ長と表現
NULL	0 byte
1 byte 整数	1 byte
2 byte 整数	2 byte
4 byte 整数	4 byte
8 byte 整数	8 byte
8 byte float	8 byte (IEEE754?)
文字列	1 byte + 文字列長 (LS)
日時	X byte 後述

日時表現

DATE	4, 年,月,日
DATETIME	11, 年,月,日,時,分,秒,マイクロ秒
DATETIME	7, 年,月,日,時,分,秒
TIMESTAMP	11, 年,月,日,時,分,秒,マイクロ秒
TIMESTAMP	7, 年,月,日,時,分,秒
TIME	12, 符号,日,時,分,秒,マイクロ秒
TIME	8, 符号,日,時,分,秒
YEAR	2, 年

年:2byte, マイクロ秒:4byte, TIMEの日:4byte, 他:1byte

クエリ応答パケット（サーバー→クライアント）

- 通常クエリと同じ

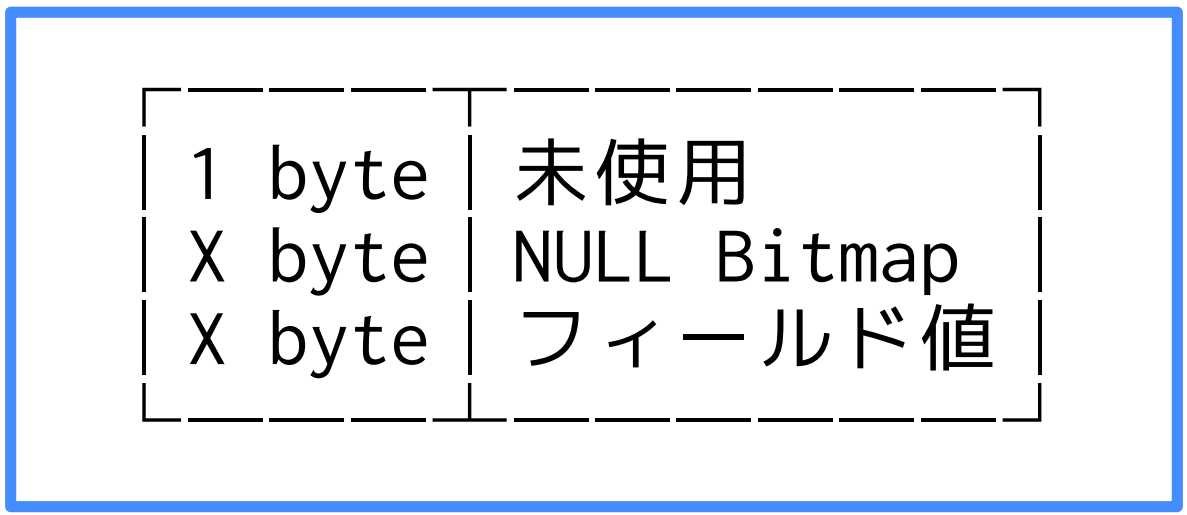
結果セット

- フィールドパケット * フィールド数
- レコードパケット * レコード数
- EOFパケット

フィールドパケット

- 通常クエリと同じ

レコードパケット



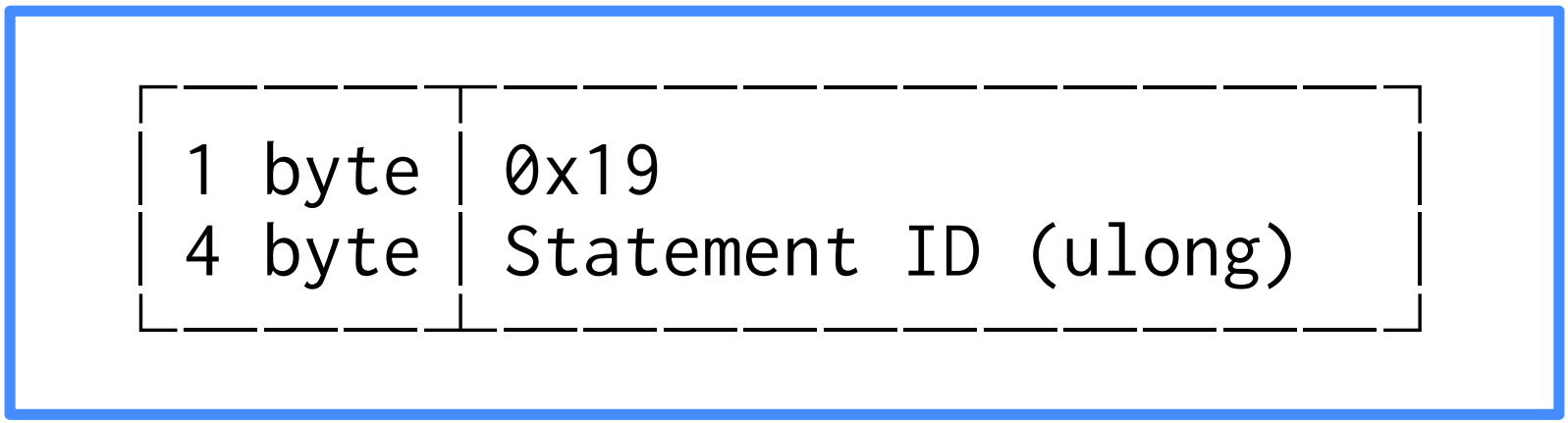
NULL bitmap

- パラメータの値のうち NULL のものを 1、NULL 以外を 0 で表したビットマップ
- 下位２ビットは未使用
- (フィールド数 + 1)/8+1 バイト
- 例: パラメータが (1, nil, 2, 3, nil) の場合 0x48(0100_10XX)

フィールド値

- パラメータの表現と同じ

クエリ破棄パケット (クライアント→サーバー)



1 byte	0x19
4 byte	Statement ID (ulong)

サーバーからの応答は無い

まとめ

触れなかったこと

- SSLプロトコル
- 圧縮プロトコル
- 複数クエリ
- プリペアドステートメント
  - サーバーサイド カーソル
  - 巨大データ送信
- MySQL 5.6

まとめ

- MySQLのプロトコルは結構複雑
- 素直に標準API使ったほうが吉
- 特殊なサーバー＆クライアント作りたい人は頑張ってください

ご清聴ありがとうございました