

XProtocol始めました、

Haskellで

Naoto Ogawa

# テーマ

- ✓ XProtocolとは

- ✓ 使い方をごく簡単に紹介します。

- ✓ XProtocolの内部を覗く

- ✓ バイナリを見ていきましょう。

- ✓ HaskellでのXProtocol実装状況報告

- ✓ <https://github.com/naoto-ogawa/h-xproto-mysql>

# XProtocol超入門

テーマ 1

# XProtocolとは

- ✓ DBサーバとクライアントの通信規約
- ✓ 通信データはProtocolBufferで規定
- ✓ CRUDスタイルとSQLスタイルの形式
- ✓ ドキュメントストアとの親和性のある規約
- ✓ パイプライン処理

個人的にはここにもっと着目されてもよいと思う。



# XDev API

- ✓ XProtocolを実装したAPI
- ✓ ドキュメントストアとの親和性のあるAPI
- ✓ EBNF定義(CRUD, セッション、結果セット)
- ✓ 複数DBサーバのコネクション管理も含む(はず)



# 使い方

- ✓ XPluginの有効化
- ✓ 対応ドライバ(XDevAPI)を使う
- ✓ <https://dev.mysql.com/doc/index-connectors.html>

# Tip

- ✓ world\_xデータベースをインストールしておくといいです。
- ✓ <https://dev.mysql.com/doc/refman/5.7/en/mysql-shell-tutorial-python-download.html>

```
mysql-sql> show tables;
```

Tables_in_world_x
city
country
countryinfo
countrylanguage

```
mysql-sql> desc countryinfo;
```

Field	Type	Null	Key	Default	Extra
doc	json	YES		null	
_id	varchar(32)	NO	PRI	null	STORED GENERATED

ドキュメントストアには必ず\_id列が必要

# XPlugin有効化

```
mysql-sql> INSTALL PLUGIN mysqlx SONAME 'mysqlx.so';
```

```
mysql-sql> show plugins;
```

Name	Status	Type	Library	License
mysqlx	ACTIVE	DAEMON	mysqlx.so	GPL



# MySQL Shell

```
$ mysqlsh -u root --port=33060
```

```
Creating a Session to 'root@localhost:33060'
```

```
Enter password:
```

```
Node Session successfully established. No  
default schema selected.
```

```
Welcome to MySQL Shell 1.0.9
```

```
...
```

```
mysql-js> db = session.getSchema('world_x');
```

```
<Schema:world_x>
```

通常のSQLクライアントとしても利用可

# MySQL Shell

```
mysql-js> db.countryinfo.find().limit(1);
```

```
[  
  {  
    "GNP": 828,  
    "IndepYear": null,  
    "Name": "Aruba",  
    ...  
    "government": {  
      ...  
      "HeadOfState": "Beatrix"  
    }  
  }  
]  
1 document in set (0.02 sec)
```

メソッド連鎖(chain)のAPI

# MySQL Shell

```
mysql-js>
```

```
db.countryinfo.find("$.GNP = 828").fields("$.Name")
```

```
[
```

```
{
```

```
  "$.Name": "Aruba"
```

```
}
```

```
]
```

```
1 document in set (0.03 sec)
```

# Java

```
import com.mysql.cj.api.xdevapi.*;
import com.mysql.cj.xdevapi.XSessionFactory;
public class Foo {
    public static void main(String[] args) {
        XSession session = new XSessionFactory().getSession(
            "mysqlx://localhost:33060?user=root&password=root");
        Schema schema = session.getSchema("world_x");
        Collection coll = schema.getCollection("countryinfo");
        FindStatement findStatement = coll.find(
            "$.demographics.Population = 278357000 &&
            $.demographics.LifeExpectancy > 77");
        DocResult docRet = findStatement.execute();
        System.out.println(docRet.next());
    }
}
```

ドライバが式や項をXProtocolに変換する必要がある



# XProtocolを覗く

テーマ2

# XProtocol実装に必要な知識

- ✓ XProtocolの仕様の理解
- ✓ ProtocolBufferの理解
- ✓ ネットワークプログラミングの理解
- ✓ 構文解析の理解
- ✓ 既存ライブラリの理解
- ✓ データベースの理解（暗黙の前提）

これから説明

対象外

# (寄り道) 2017/11月現在

✓ XProtocolの仕様の理解 名前の衝突の回避方法間違った😅(後述)

✓ ProtocolBufferの理解

TLSの仕組みの理解づらい

✓ ネットワークプログラミングの理解

ここの実装で立ち止まっている。テストやってるが、後回しにしていた実装は後回しにできないwwwww

✓ 構文解析の理解

✓ 既存ライブラリの理解

パーサのライブラリ選択間違ったかも😓

✓ データベースの理解

挙動の理解が進まない。特に日付時刻関連

# 仕様の理解

- ✓ ドキュメント
- ✓ メッセージ構造
- ✓ パケットの確認
- ✓ クライアント／サーバメッセージ
- ✓ コマンドの確認
- ✓ Find / 結果セット / Expr / JSON



# ドキュメント

- ✓ MySQL Internal Manual
  - ✓ Chapter 15 X Protocol
    - ✓ <https://dev.mysql.com/doc/internals/en/x-protocol.html>
- ✓ protoファイルのコメント👉
  - ✓ mysql-server -> rapid -> plugin -> x -> protocol
    - ✓ <https://github.com/mysql/mysql-server/tree/5.7/rapid/plugin/x/protocol>

# Protoファイル

```
$ ls -la mysql-x-protocol
```

```
..  
mysqlx.proto  
mysqlx_connection.proto  
mysqlx_crud.proto  
mysqlx_datatypes.proto  
mysqlx_expect.proto  
mysqlx_expr.proto  
mysqlx_notice.proto  
mysqlx_resultset.proto  
mysqlx_session.proto  
mysqlx_sql.proto
```

バイナリデータの定義  
定義されているオブジェクトは  
ファイル名から推測される通り

# オブジェクトの生成

```
> .cabal-sandbox/bin/hprotoc -d ./src/ -n mysql-  
server-rapid-plugin-x-protocol/*.proto
```

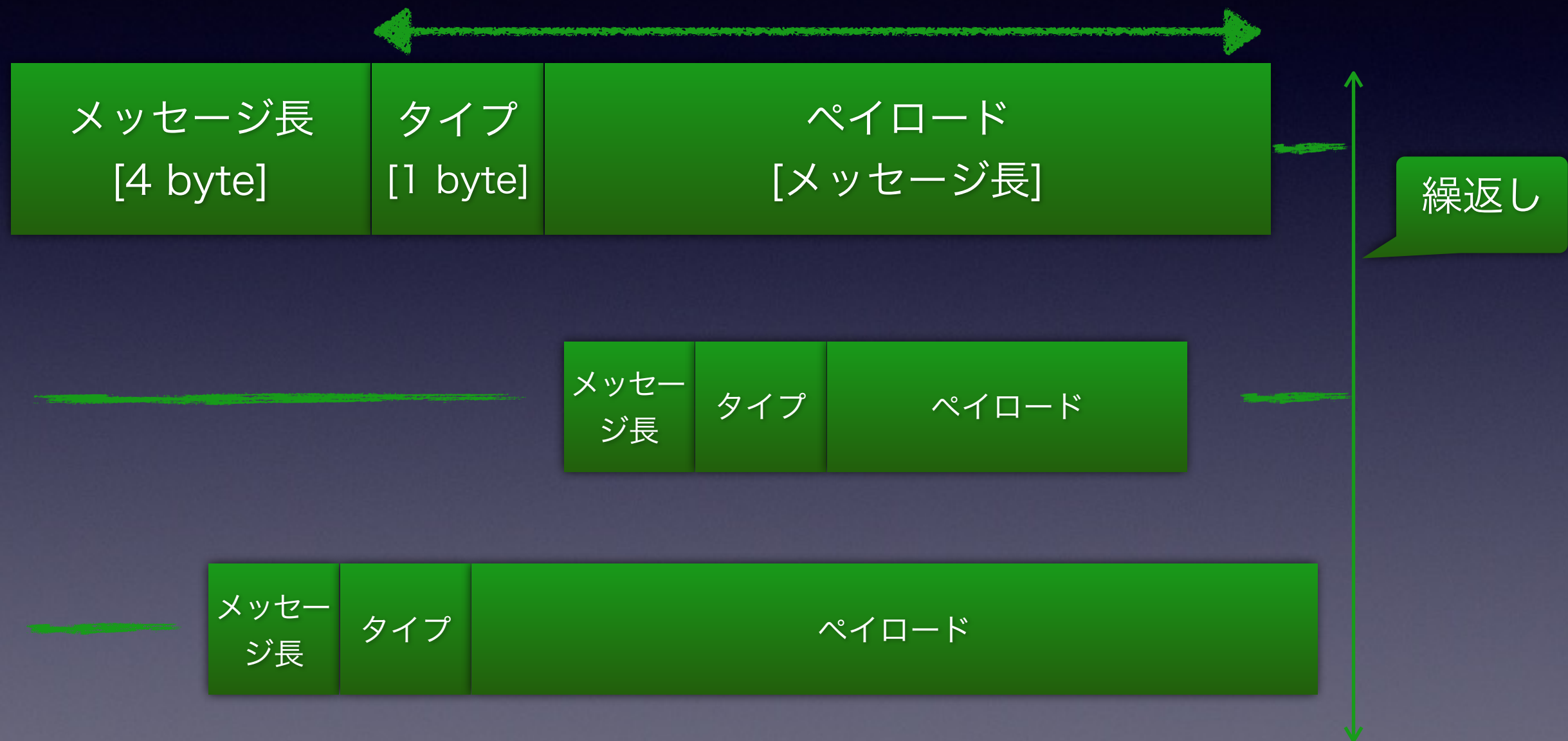
```
> ogawanaoto$ tree tmp/src/
```

```
tmp/src/  
├── Com  
│   └── Mysql  
│       ├── Cj  
│       │   └── Mysqlx  
│       │       ├── Protobuf  
│       │       │   ├── Any  
│       │       │   │   └── Type.hs  
│       │       │   ├── Any.hs  
│       │       │   ├── Any.hs-boot  
│       │       │   ├── Array.hs  
│       │       │   ├── AuthenticateContinue.hs  
│       │       │   ├── AuthenticateOk.hs  
│       │       │   ├── AuthenticateStart.hs  
│       │       │   ├── Capabilities.hs  
│       │       │   ├── CapabilitiesGet.hs  
│       │       │   ├── CapabilitiesSet.hs  
│       │       │   ├── Capability.hs  
│       │       │   ├── ClientMessages  
│       │       │   │   └── Type.hs  
│       │       │   ├── ClientMessages.hs  
│       │       │   ├── Close.hs  
│       │       │   ├── Collection.hs  
│       │       │   ├── Column.hs  
│       │       │   ├── ColumnIdentifier.hs  
│       │       │   ├── ColumnMetaData  
│       │       │   │   └── FieldType.hs  
│       │       │   ├── ColumnMetaData.hs  
│       │       │   ├── CreateView.hs  
│       │       │   ├── DataModel.hs  
│       │       │   ├── Delete.hs  
│       │       │   ├── DocumentPathItem  
│       │       │   │   └── Type.hs
```

```
├── DocumentPathItem.hs  
├── DropView.hs  
├── Error  
│   └── Severity.hs  
├── Error.hs  
├── Expr  
│   └── Type.hs  
├── Expr.hs  
├── Expr.hs-boot  
├── FetchDone.hs  
├── FetchDoneMoreOutParams.hs  
├── FetchDoneMoreResultsets.hs  
├── Find.hs  
├── Frame  
│   └── Scope.hs  
├── Frame.hs  
├── FunctionCall.hs  
├── Identifier.hs  
├── Insert  
│   └── TypedRow.hs  
├── Insert.hs  
├── Limit.hs  
├── ModifyView.hs  
├── Object  
│   └── ObjectField.hs  
├── Object.hs  
├── ObjectExpr  
│   └── ObjectFieldExpr.hs  
├── ObjectExpr.hs  
├── Ok.hs  
├── Open  
│   ├── Condition  
│   │   └── ConditionOperation.hs  
│   ├── Condition.hs  
│   └── CtxOperation.hs  
├── Open.hs  
└── Operator.hs
```

# メッセージ構造

✓ メッセージ長+タイプ+ペイロードの繰り返し



繰り返し全体でのメッセージサイズの情報はないので実装が、、、



# TCPパケット例

```
> tcpdump
```

```
IP localhost.60446 > localhost.33060: Flags [P.], seq  
664:760, ack 752, win 50943, options [nop,nop,TS val  
927166495 ecr 927166492], length 96
```

```
0x0000: 4500 0094 34a9 4000 4006 0000 7f00 0001  
0x0010: 7f00 0001 ec1e 8124 c7df 7f1a e59f 4f65  
0x0020: 8018 c6ff fe88 0000 0101 080a 3743 701f  
0x0030: 3743 701c 5c00 0000 1312 130a 0466 7567  
0x0040: 6112 0b74 6573 745f 7363 6865 6d61 1802  
0x0050: 2218 0805 3214 0a02 3d3d 1208 0801 1204  
0x0060: 1202 6964 1204 0806 3800 3a22 0a09 1207  
0x0070: 6d65 7373 6167 6510 011a 1308 0222 0f08  
0x0080: 084a 0b0a 096d 7367 332b 2b2b 2b2b 4204  
0x0090: 0801 1006
```

# TCPパケット例

```
> tcpdump
```

```
IP localhost.60446 > localhost.33060: Flags [P.], seq  
664:760, ack 752, win 50943, options [nop,nop,TS val  
9271664
```

メッセージ長(LE)

タイプ

0x0000:	4500	0094	34a9	4000	4006	0000	7f00	0001
0x0010:	7f00	0001	ec1e	8124	c7df	7f1a	e59f	4f65
0x0020:	8018	c6ff	fe88	0000	0101	080a	3743	701f
0x0030:	3743	701c	5c00	0000	1312	130a	0466	7567
0x0040:	6112	0b74	6573	745f	7363	6865	6d61	1802
0x0050:	2218	0805	3214	0a02	3d3d	1208	0801	1204
0x0060:	1202	6964	1204	0806	3800	3a22	0a09	1207
0x0070:	6d65	7373	6167	6510	011a	1308	0222	0f08
0x0080:	084a	0b0a	096d	7367	332b	2b2b	2b2b	4204
0x0090:	0801	1006						

# クライアントメッセージ

```
message ClientMessages {
  enum Type {
    CON_CAPABILITIES_GET = 1;
    CON_CAPABILITIES_SET = 2;
    CON_CLOSE = 3;

    SESS_AUTHENTICATE_START = 4;
    SESS_AUTHENTICATE_CONTINUE = 5;
    SESS_RESET = 6;
    SESS_CLOSE = 7;

    SQL_STMT_EXECUTE = 12;

    CRUD_FIND = 17;
    CRUD_INSERT = 18;
    CRUD_UPDATE = 19;
    CRUD_DELETE = 20;

    EXPECT_OPEN = 24;
    EXPECT_CLOSE = 25;

    CRUD_CREATE_VIEW = 30;
    CRUD_MODIFY_VIEW = 31;
    CRUD_DROP_VIEW = 32;
  }
}
```

交渉

認証

SQL

CRUD

パイプライン



# サーバメッセージ

```
message ServerMessages {  
    enum Type {  
        OK = 0;  
        ERROR = 1;  
  
        CONN_CAPABILITIES = 2;  
  
        SESS_AUTHENTICATE_CONTINUE = 3;  
        SESS_AUTHENTICATE_OK = 4;  
  
        // NOTICE has to stay at 11 forever  
        NOTICE = 11;  
  
        RESULTSET_COLUMN_META_DATA = 12;  
        RESULTSET_ROW = 13;  
        RESULTSET_FETCH_DONE = 14;  
        RESULTSET_FETCH_SUSPENDED = 15;  
        RESULTSET_FETCH_DONE_MORE_RESULTSETS = 16;  
  
        SQL_STMT_EXECUTE_OK = 17;  
        RESULTSET_FETCH_DONE_MORE_OUT_PARAMS = 18;  
    };  
}
```

応答ステータス

認証

補足情報

結果セット



# Find

```
message Find {  
    required Collection collection = 2;  
  
    optional DataModel data_model = 3;  
    repeated Projection projection = 4;  
    optional Mysqlx.Expr.Expr criteria = 5;  
    repeated Mysqlx.Datatypes.Scalar args = 11;  
    optional Limit limit = 6;  
    repeated Order order = 7;  
    repeated Mysqlx.Expr.Expr grouping = 8;  
    optional Mysqlx.Expr.Expr grouping_criteria = 9;  
};
```

# Find

```
mysql-js>
  schema.getCollection("countryinfo")
  .find("$.geography.Continent= 'Asia'")
  .fields("avg(IndepYear)")
  .groupBy("geography.Continent")
[
  {
    "avg(IndepYear)": 1666.69
  }
]
1 document in set (0.00 sec)
```

※クエリ自体は冗長です(例のための例です)。

# Findの送信パケット

```
IP localhost.62195 > localhost.33060: Flags [P.], seq 70:75, ack 148, win
12981, options [nop,nop,TS val 776143993 ecr 776143992], length 5
0x0000:  4500 0039 8ebe 4000 4006 0000 7f00 0001  E..9..@.@.....
0x0010:  7f00 0001 f2f3 8124 808d c477 6bea 89cc  ....$...wk...
0x0020:  8018 32b5 fe2d 0000 0101 080a 2e43 0479  ..2..-.....C.y
0x0030:  2e43 0478 b100 0000 11                .C.x.....      IP
localhost.62195 > localhost.33060: Flags [P.], seq 75:251, ack 148, win 12981,
options [nop,nop,TS val 776143993 ecr 776143992], length 176
0x0000:  4500 00e4 50d8 4000 4006 0000 7f00 0001  E...P.@.@.....
0x0010:  7f00 0001 f2f3 8124 808d c47c 6bea 89cc  ....$...|k...
0x0020:  8018 32b5 fed8 0000 0101 080a 2e43 0479  ..2.....C.y
0x0030:  2e43 0478 1216 0a0b 636f 756e 7472 7969  .C.x....countryi
0x0040:  6e66 6f12 0777 6f72 6c64 5f78 1801 2232  nfo..world_x.."2
0x0050:  0a20 0804 2a1c 0a05 0a03 6176 6712 1308  ....*.....avg...
0x0060:  0112 0f0a 0d08 0112 0949 6e64 6570 5965  ....IndepYe
0x0070:  6172 120e 6176 6728 496e 6465 7059 6561  ar..avg(IndepYea
0x0080:  7229 2a3c 0805 3238 0a02 3d3d 1222 0801  r)*<..28..=="..
0x0090:  121e 0a0d 0801 1209 6765 6f67 7261 7068  ....geograph
0x00a0:  790a 0d08 0112 0943 6f6e 7469 6e65 6e74  y.....Continent
0x00b0:  120e 0802 220a 0804 2a06 0a04 4173 6961  ...."*...Asia
0x00c0:  4222 0801 121e 0a0d 0801 1209 6765 6f67  B".....geog
0x00d0:  7261 7068 790a 0d08 0112 0943 6f6e 7469  raphy.....Conti
0x00e0:  6e65 6e74                nent
```



# Protocol Bufferデコード

```
$ protoc-3/bin/protoc --  
decode_raw < memo/  
dump_shell_find_field_groupby_  
2.bin
```

```
2 {  
  1: "countryinfo"  
  2: "world_x"  
}  
3: 1  
4 {  
  1 {  
    1: 4  
    5 {  
      1 {  
        1: "avg"  
      }  
      2 {  
        1: 1  
        2 {  
          1 {
```

```
1: 1  
2 {  
  9:  
    0x7261655970656446e  
  }  
}  
}  
}  
}  
}  
}  
2: "avg(IndepYear)"  
}
```

続<



# Protocol Bufferデコード

```
5 {
  1: 5
  6 {
    1: "=="
    2 {
      1: 1
      2 {
        1 {
          1: 1
          2: "geography"
        }
        1 {
          1: 1
          2: "Continent"
        }
      }
    }
  }
}
2 {
  1: 2
  4 {
    1: 4
  }
}

5 {
  1: "Asia"
}
}
}
8 {
  1: 1
  2 {
    1 {
      1: 1
      2: "geography"
    }
    1 {
      1: 1
      2: "Continent"
    }
  }
}
```

# オブジェクトデコード

```
>readObj "memo/  
dump_shell_find_field_groupby_  
2.bin" :: IO PF.Find
```

```
collection {  
  name: "countryinfo"  
  schema: "world_x"  
}  
data_model: DOCUMENT  
projection {  
  source {  
    type: FUNC_CALL  
    function_call {  
      name {  
        name: "avg"  
      }  
      param {  
        type: IDENT  
        identifier {
```

```
document_path {  
  type: MEMBER  
  value: "IndepYear"  
}  
}  
}  
}  
}  
alias: "avg(IndepYear)"  
}
```

続<

# オブジェクトデコード

```
criteria {  
  type: OPERATOR  
  operator {  
    name: "=="  
    param {  
      type: IDENT  
      identifier {  
        document_path {  
          type: MEMBER  
          value: "geography"  
        }  
        document_path {  
          type: MEMBER  
          value: "Continent"  
        }  
      }  
    }  
  }  
  param {  
    type: LITERAL  
    literal {  
      type: V_OCTETS
```

```
      v_octets {  
        value: "Asia"  
      }  
    }  
  }  
  grouping {  
    type: IDENT  
    identifier {  
      document_path {  
        type: MEMBER  
        value: "geography"  
      }  
      document_path {  
        type: MEMBER  
        value: "Continent"  
      }  
    }  
  }  
}
```

# FindのTree

Find

collection :: Collection

name :: Utf8

countryinfo

schema :: Utf8

world\_x

data\_model :: DataModel

DOCUMENT

projection :: Projection

criteria :: Expr

grouping :: Expr



projection :: Projection

source :: Expr

function\_call :: FunctionCall

name :: Identifier

name :: Utf8

avg

param :: Expr

name :: ColumnIdentifier

document\_path :: DocumentPathItem

type :: Type

value :: Utf8

IndepYear

alias :: Utf8

criteria :: Expr

operator :: Operator

name :: Utf8

==

param :: Expr

name :: ColumnIdentifier

document\_path :: DocumentPathItem

type :: Type

geography

value :: Utf8

Continent

literal :: Scalar

type :: Type

v\_octets :: Octets

value :: ByteString

Asia

grouping :: Expr

name :: ColumnIdentifier

document\_path :: DocumentPathItem

type :: Type

value :: Utf8

geography

Continent

# 結果セット

```
message ColumnMetaData {
```

```
  enum FieldType {
```

```
    SINT    = 1;
```

```
    UINT    = 2;
```

```
    DOUBLE  = 5;
```

```
    FLOAT   = 6;
```

```
    BYTES   = 7;
```

```
    TIME    = 10;
```

```
    DATETIME = 12;
```

```
    SET     = 15;
```

```
    ENUM    = 16;
```

```
    BIT     = 17;
```

```
    DECIMAL = 18;
```

```
  }
```

```
  // datatype of the field in a row  
  required FieldType type = 1;
```

```
  optional bytes name = 2;
```

```
  optional bytes original_name = 3;
```

```
  optional bytes table = 4;
```

```
  optional bytes original_table = 5;
```

```
  optional bytes schema = 6;
```

```
  optional bytes catalog = 7;
```

```
  optional uint64 collation = 8;
```

```
  optional uint32 fractional_digits = 9;
```

```
  optional uint32 length = 10;
```

```
  optional uint32 flags = 11;
```

```
  optional uint32 content_type = 12;
```

```
}
```

```
message Row {
```

```
  repeated bytes field = 1;
```

```
}
```



# 結果セット

```
message Row {  
  repeated bytes field = 1;  
}
```

- ・ データはbyte列で来るので、ライブラリ側で、（メタデータを元に）型を復元したうえで、オブジェクトに変換する必要がある。💧😓

# Decimal

- ✓ Decimal型はBinary Coded Decimal
- ✓ [https://en.wikipedia.org/wiki/Binary-coded\\_decimal](https://en.wikipedia.org/wiki/Binary-coded_decimal)
- ✓ scale | BCD | sign | [0x0]
  - ✓ scale – 8bit 少数部分の桁数
  - ✓ BCD – 4 bits
  - ✓ sign – 4 bits (0xc="+", 0xd="-")
  - ✓ 0x0 – 4 bit パディング

結果セットにあるカラムのデータはbyte列なので、デコードはライブラリ側でおこなう必要がある。

# Decimal

03	10	00	c0			→	1.000
03	99	99	90	00	c0	→	99999.000
03	99	99	91	00	c0	→	99999.100
03	99	99	91	23	c0	→	99999.123
03	99	98	76	d0		→	-999.874
04	12	34	01	d0		→	-12.3401
02	03	4c				→	0.34

参考 (1.000のmessage)

07 00 00 00 0d 0a 04 03 10 00 c0

+-->len                   |   |   +-->len(PB)

          type<--+   +-->(PB) 0 0001(tag) 010(type)

# Expr

```
message Expr {
  enum Type {
    IDENT          = 1;
    LITERAL        = 2;
    VARIABLE       = 3;
    FUNC_CALL      = 4;
    OPERATOR       = 5;
    PLACEHOLDER   = 6;
    OBJECT         = 7;
    ARRAY          = 8;
  };

  required Type type = 1;

  optional ColumnIdentifier identifier = 2;
  optional string      variable = 3;
  optional Mysqlx.Datatypes.Scalar literal = 4;
  optional FunctionCall function_call = 5;
  optional Operator    operator = 6;
  optional uint32      position = 7;
  optional Object      object = 8;
  optional Array       array = 9;
}
```



# Literal Expr

```
message Expr {  
  enum Type {  
    IDENT          = 1;  
    LITERAL       = 2;  
    VARIABLE       = 3;  
    FUNC_CALL      = 4;  
    OPERATOR       = 5;  
    PLACEHOLDER   = 6;  
    OBJECT         = 7;  
    ARRAY          = 8;  
  };  
};
```

**required Type type = 1;**

```
optional ColumnIdentifier identifier = 2;  
optional string      variable = 3;  
optional Mysqlx.Datatypes.Scalar literal = 4;  
optional FunctionCall function_call = 5;  
optional Operator    operator = 6;  
optional uint32      position = 7;  
optional Object      object = 8;  
optional Array       array = 9;  
};
```

# Literal Scalar

```
message Scalar {
  message String {
    required bytes value = 1;
    optional uint64 collation = 2;
  };
  message Octets {
    required bytes value = 1;
    optional uint32 content_type = 2;
  };
  enum Type {
    V_SINT = 1; V_UINT = 2; V_NULL = 3; V_OCTETS = 4;
    V_DOUBLE = 5; V_FLOAT = 6; V_BOOL = 7; V_STRING = 8;
  };
  required Type type = 1;

  optional sint64 v_signed_int = 2;
  optional uint64 v_unsigned_int = 3;
  // 4 is unused, was Null which doesn't have a storage anymore
  optional Octets v_octets = 5;
  optional double v_double = 6;
  optional float v_float = 7;
  optional bool v_bool = 8;
  optional String v_string = 9;
}
```

# Example Literal Expr

```
mysql-js> db.country.select("99").limit(1);
```



```
+-----+
```

```
| 99 |
```

```
+-----+
```

```
| 99 |
```

```
+-----+
```

```
1 row in set (0.13 sec)
```

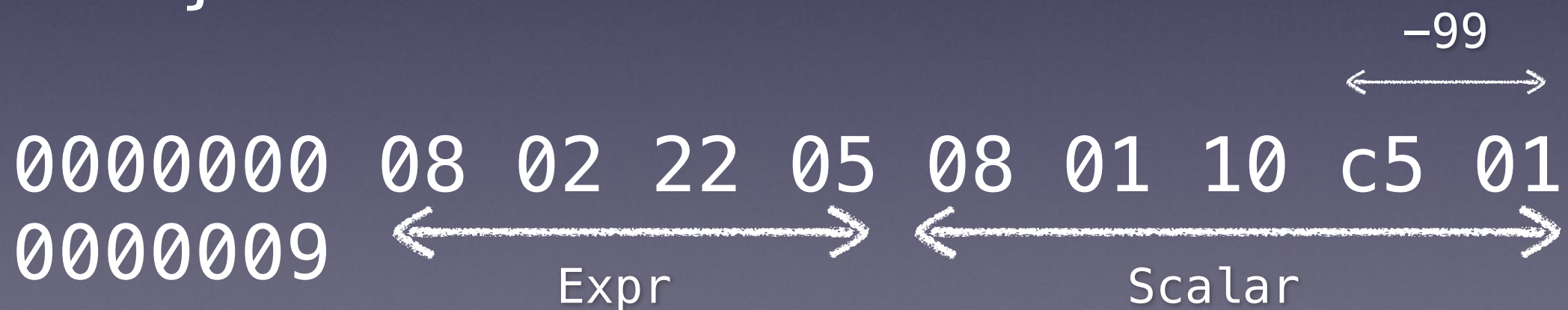
```
mysql-js>
```

※例のための例です。

# Literal Expr

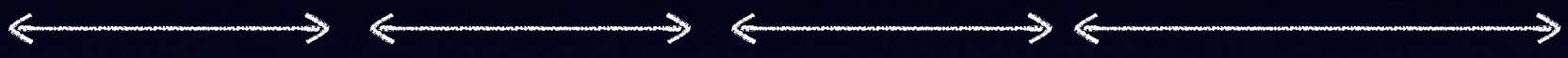
Expr

```
{ type' = LITERAL
  , literal = Just
    ( Scalar
      { type' = V_SINT
        , v_signed_int = Just(-99)
      }
    )
}
```





# note : Protocol Buffer

00000000 08 02 22 05 08 01 10 c5 01  
00000009   
key = value key = value key = value key = value

08 : 0000 1000 > 0-0001-000 > 1-0 >  
tag1-type0(Variant(enum))

02 : 0000 0010 > 0-0000010 > 2

22 : 0010 0010 > 0-0100-010 > 4-2 >  
tag4-type2(Letnngth delimited)

05 : 0000 0101 > 0-0000101 > 5

# note : Protocol Buffer

00000000 08 02 22 05 08 01 10 c5 01  
00000009  $\longleftrightarrow$   $\longleftrightarrow$   $\longleftrightarrow$   $\longleftrightarrow$   
key = value key = value key = value key = value

**08** : 0000 1000 > 0-0001-000 > 1-0 >  
tag1-type0(Variant(enum))

**01** : 0000 0001 > 0-0000001 > 1

**10** : 0001 0000 > 0-0010-000 > 2-0 >  
tag2-type0(Variant(sint64))

**c5 01** : 1100 0101 0000 0001 > 1-1000101  
0-0000001 > 0000001 1000101 > 197 > -99

zigzag encoding

# JSON

```
{  
  'a' : 11  
  , 'b' : 'steve'  
  , 'c' : [9,99]  
}
```

# JSON

```
type: OBJECT
object {
  fld {
    key: "a"
    value {
      type: LITERAL
      literal {
        type: V_SINT
        v_signed_int: 11
      }
    }
  }
  fld {
    key: "b"
    value {
      type: LITERAL
      literal {
        type: V_STRING
        v_string {
          value: "steve"
        }
      }
    }
  }
}
```

Expr

Expr

```
fld {
  key: "c"
  value {
    type: ARRAY
    array {
      value {
        type: LITERAL
        literal {
          type: V_SINT
          v_signed_int: 9
        }
      }
      value {
        type: LITERAL
        literal {
          type: V_SINT
          v_signed_int: 99
        }
      }
    }
  }
}
```

Expr

Expr



# 関数

- ✓ドライバが、文字列をパースし、XProtocolの表現に変換する
  - ✓変換後表現が、妥当なシンタックス、セマンティックスであるかは、ドライバは関知せず
  - ✓ドライバ開発者は構文解析の知識が必要
- ✓SQLで利用出来る関数が全てXDevAPIでサポートされているとは限らない

# 関数

```
mysql-sql> select * from items;
```

id	price
1	100
2	200
3	300

```
mysql-js> db.items.select('sum(price)');
```

sum(`price`)
600

# 関数

Find

```
{ collection = Collection {name="items",schema=Just "x_protocol_test"}
, data_model = Just TABLE
, projection = fromList
  [ Projection
    { source = Expr
      { type' = FUNC_CALL
      , function_call = Just
        ( FunctionCall
          { name = Identifier { name = "sum" }
          , param = fromList
            [ Expr
              { type' = IDENT
              , identifier = Just
                ( ColumnIdentifier
                  { document_path = fromList []
                  , name = Just "price"
                  }
                )
              }
            ]
          }
        )
      }
    ]
  }
, args = fromList [], order = fromList [], grouping = fromList []}
```

# 関数

```
mysql-js> db.foo.select('XXX(price)');  
ERROR: 1305: FUNCTION x_protocol_test.XXX does not exist
```

```
{ source = Expr  
  { type' = FUNC_CALL  
    , function_call = Just  
      ( FunctionCall  
        { name = Identifier { name = "XXX" }  
          , param = fromList  
            [ Expr  
              { type' = IDENT  
                , identifier = Just  
                  ( ColumnIdentifier  
                    { document_path = fromList []  
                      , name = Just "price"  
                    }  
                  )  
                }  
            ]  
          }  
        )  
      }  
    }  
  }  
}
```

サーバのエラーメッセージ

XProtocol上は妥当な形式なのでドライバではエラーとならない

XXXというユーザ定義関数があれば、正常に処理されるはず。



# パイプライン

- ✓Nリクエスト : 1レスポンス
- ✓失敗時の挙動を制御できる
  - ✓例：失敗したら後続は実行しない
- ✓挙動をネストできる
- ✓トランザクションとは独立

# パイプライン

case\_a : 1->2->3->4->5 =>[1,2,3,4,5], []

case\_b : 1->2->3->4->5' =>[1,2,3,4], [err\_msg]

case\_c : 1->2->3'->4->5 =>[1,2], [err\_msg]

case\_d : 1'->2->3->4->5 =>[], [err\_msg]

カンマなし : 正常なinsert文

カンマあり : 不正なinsert文

# パイプライン実装

クエリ関数(monad)

送信関数(monad)

受信関数(monad)

Haskellのパイプライン実装  
の基本アイデアはモナドの組  
み替え

パイプラインクエリ  
関数(monad)

送信関数(monad)

送信関数(monad)

送信関数(monad)

.....

受信関数(monad)

受信関数(monad)

受信関数(monad)

.....

# エラー

```
message Error {  
    optional Severity severity = 1 [ default = ERROR ];  
    required uint32 code       = 2;  
    required string sql_state  = 4;  
    required string msg        = 3;  
  
    enum Severity { ERROR = 0; FATAL = 1; };  
}
```

Type	Meaning	Used For
0	Varint	int32, int64, uint32, uint64, sint32, sint64, bool, enum
1	64-bit	fixed64, sfixed64, double
2	Length-delimited	string, bytes, embedded messages, packed repeated fields
3	Start group	groups (deprecated)
4	End group	groups (deprecated)
5	32-bit	fixed32, sfixed32, float

<https://developers.google.com/protocol-buffers/docs/encoding>



# エラー

3a 00 00 00 01 08 00 10 99 0a 1a 2b 46 55 4e 43  
54 49 4f 4e 20 78 5f 70 72 6f 74 6f 63 6f 6c 5f  
74 65 73 74 2e 58 58 58 20 64 6f 65 73 20 6e 6f  
74 20 65 78 69 73 74 22 05 34 32 30 30 30

08 -> 0000-1000 -> 0-0001-000 -> tag1-type0

10 -> 0001-0000 -> 0-0010-000 -> tag2-type0

1a -> 0001-1010 -> 0-0011-010 -> tag3-type2

22 -> 0010-0010 -> 0-0100-010 -> tag4-type2

99 -> 1001-1001 -> 1-0011001 -----+

0a -> 0000-1010 -> 0-0001010 -> 001010 0011001 = 1305

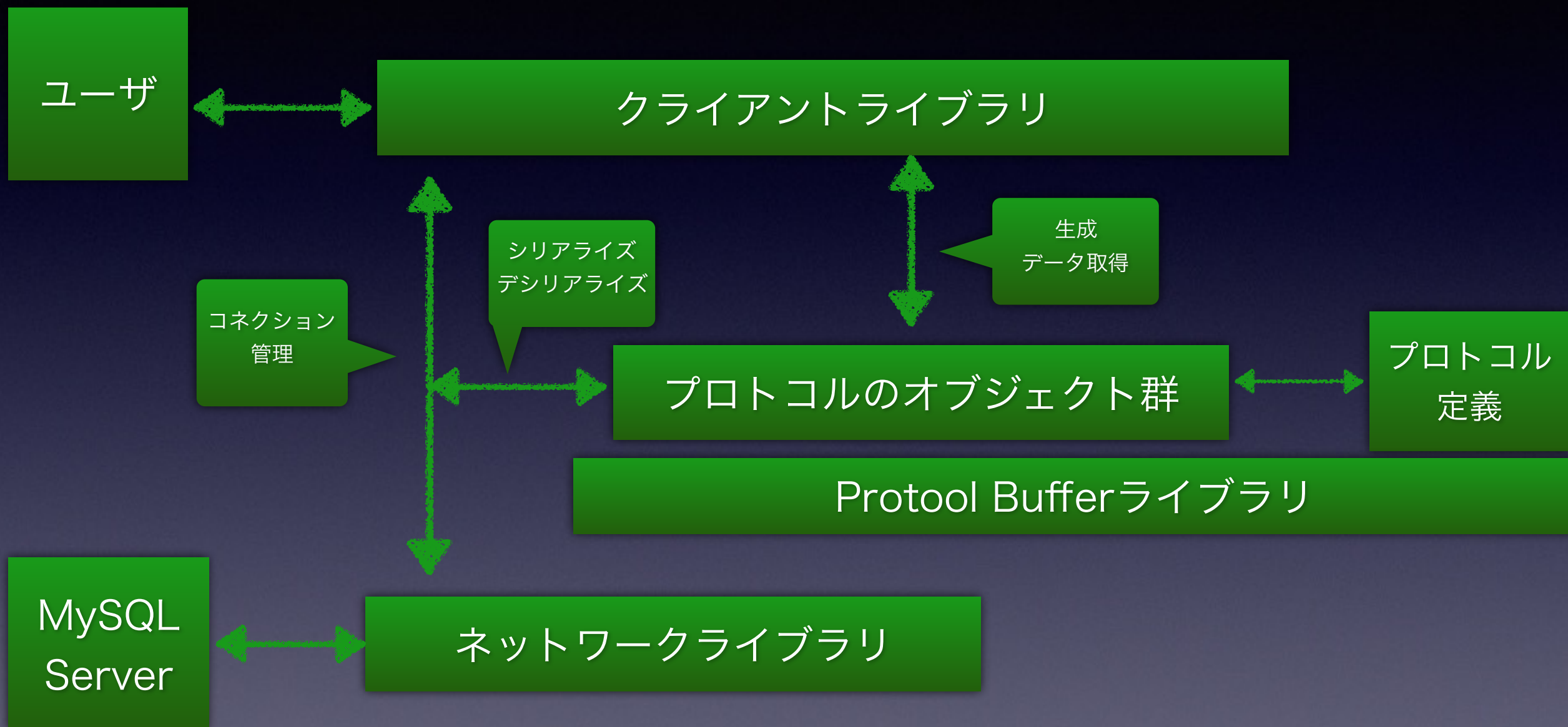
Error

```
{ severity = Just ERROR
, code = 1305
, sql_state = "42000"
, msg = "FUNCTION x_protocol_test.XXX does not exist"
}
```

# XProtocolドライバ開発

- ✓ protoファイルより各言語の定義を生成
  - ✓ 各言語用のProtocol Buffer用のツールを使う
- ✓ オブジェクトを生成/からデータを取得する実装
- ✓ オブジェクトをネットワークで送受信する実装
  - ✓ シリアライズ、デシリアライズはProtocol Bufferライブラリで提供される

# ドライバの構造



# X Protocol 注意点

- ✓用語の重複に注意
  - ✓Objectの定義が2箇所ある
    - ✓<https://github.com/mysql/mysql-server/search?utf8=✓&q=ObjectField&type=>
  - ✓Errorが2箇所ある
    - ✓ErrorメッセージとWarning内のERRORタイプ
- ✓同じ概念だが異なる名称
  - ✓notice(ドキュメントの定義) と  
frame(protoファイルの定義)



# ドライバ設計の観点

- ✓ MySQLの型と言語側の型の整合性
  - ✓ 例：MySQLのTIME型はマイナス値を取りうる
- ✓ 結果取得のAPIをどうするか？
  - ✓ 例：正常だがメッセージがあるケース（不正な値だがMySQLサーバ側が補正するケース）
- ✓ ドライバのレイアリング（立ち位置）
  - ✓ 例：隠蔽かをどこまでやるか、プロトコルの詳細を意識させないAPI
- ✓ 依存関係を少なくしたい(どのライブラリを使うか)

# HaskellでXProtocol

テーマ3

# Haskellでの実装

- ✓h-xproto-mysql

- ✓<https://github.com/naoto-ogawa/h-xproto-mysql>

- ✓XProtocolの詳細をできるだけユーザに意識させず、XProtocolのメリットを提供することを目標

- ✓JavaやJavaScriptのXDevAPIとは異なるAPI

# 実装

- ✓コネクション管理
- ✓トランザクション管理
- ✓SQL実行
- ✓CRUD実行
- ✓Pipeline実行
- ✓結果セットのレコードへのバインディング
- ✓JSON連携
- ✓開発用のプロキシ



# 今後

- ✓交渉の実装
- ✓TLSでの接続の実装
- ✓使いやすいAPI（プロトコルの詳細をユーザに意識させない）
- ✓ストリーミング対応
- ✓結果セットが分割される場合の実装
- ✓データ型サポート: UTC, Unicode, GeometryType, etc.

# 課題

- ✓インストールを簡単にする。

- ✓テスト

  - ✓品質を確保するにはどの程度テストが必要？

- ✓パフォーマンス計測、改善

# 疑問点

- ✓交渉のメッセージ解析
- ✓日付型のバイナリ構造
- ✓細かいSQLの機能

<https://github.com/naoto-ogawa/h-xproto-mysql/issues>

# 情報源まとめ

- ✓ Reference Manual Chapter 19 Using MySQL as a Document Store
  - ✓ <https://dev.mysql.com/doc/refman/5.7/en/document-store.html>
- ✓ Internal Manual Chapter 15 X Protocol
  - ✓ <https://dev.mysql.com/doc/internals/en/x-protocol.html>
- ✓ Introducing the MySQL Document Store
  - ✓ [https://downloads.mysql.com/presentations/20160510\\_02\\_MySQLDocumentStore.pdf](https://downloads.mysql.com/presentations/20160510_02_MySQLDocumentStore.pdf)
- ✓ MySQL Document Store: Under the Hood
  - ✓ [https://downloads.mysql.com/presentations/20160510\\_03\\_MySQL\\_Document\\_Store\\_Under\\_the\\_Hood.pdf](https://downloads.mysql.com/presentations/20160510_03_MySQL_Document_Store_Under_the_Hood.pdf)
- ✓ Connectors and APIs
  - ✓ <https://dev.mysql.com/doc/index-connectors.html>
- ✓ X Protocol MySQL workload
  - ✓ <https://dev.mysql.com/worklog/task/?id=8639>
- ✓ Protocol Buffer (Encoding)
  - ✓ <https://developers.google.com/protocol-buffers/docs/encoding>



End