
ORACLE®

运用MySQL Document Store 开发NoSQL应用 以MEAN框架为范例

杜修文
Ivan.Tu@Oracle.Com
MySQL全球事业部, Oracle

免责声明

以下内容旨在概述产品的总体发展方向。该内容仅供参考，不可纳入任何合同。本演示不承诺提供任何材料、代码或功能，也不应将其作为购买决策的依据。此处所述有关 Oracle 产品的任何特性或功能的开发、发布和时间安排均由 Oracle 自行决定。

IT面临的挑战

- 开发者想要更快的速度推出新应用系统
- 及时面市才能取得最佳的价值
- 雏型法,加速开发环境…
- 关系型数据库需事先建好数据模型
  - 可能在将来为您省下时间
  - 较没有弹性;但处理边际案例(edge case)时所用的程式码较少
  - 当Schema更改时会有额外的成本
- NoSQL 数据库不需要schema
  - 在开始时能节省一些时间
  - 但是一段时后有可能会增加操作成本
  - 更改Schema时没有额外的成本

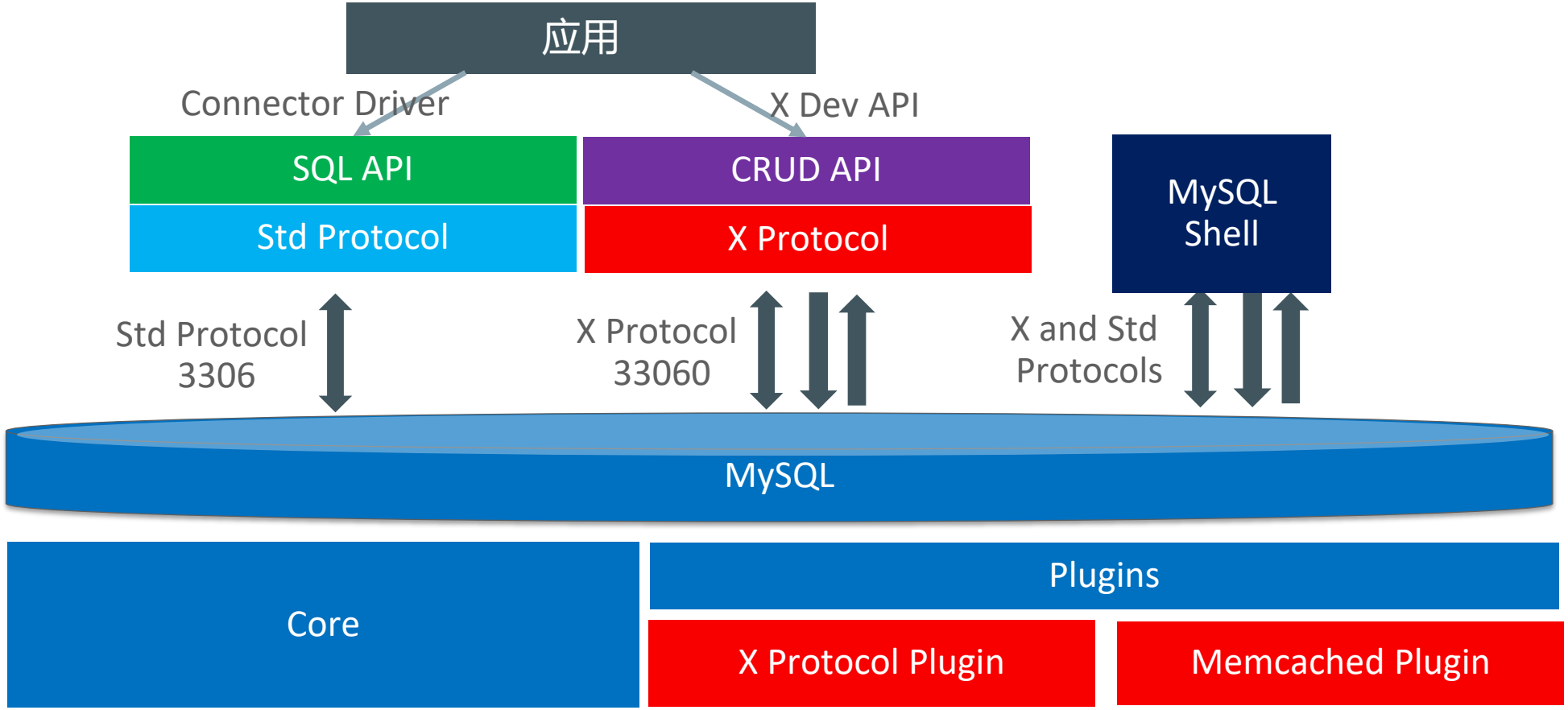
我们提供Schemaless

- MySQL as a Document Store (new!)
- 同时保有所有MySQL的功能
  - 复制
  - InnoDB
  - Performance Schema…
- 再加上schemaless
  - 使用JSON文件
  - 容易编程式的CRUD APIs

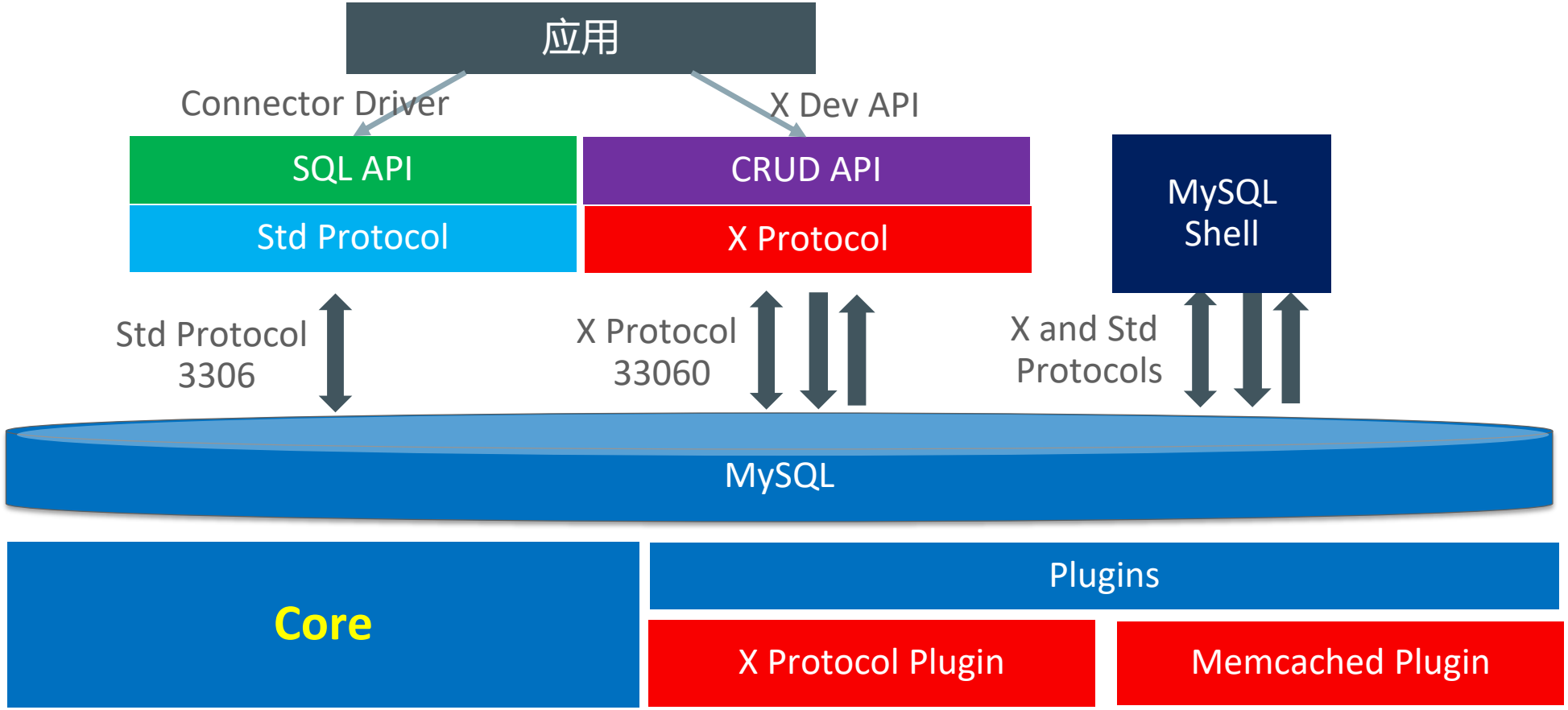
有那些新东西?

- 新的 Document APIs (X Dev API)
  - SQL and NoSQL/Document CRUD APIs will cross all Connectors
    - 同时能处理Document的Colleciton和Relational的表
  - 包括Node.js, Java , .NET 和 C++
- 新的 Interactive Shell – 有 Javascript, Python, SQL 模式可选择
- 新的 Server Features
  - 预设的 JSON数据型别和存储
  - 以Generated Columns 做索引
  - 超过 20个以上的预设 JSON功能
  - 新的X Protocol
- 新的 Connector功能

新的MySQL架构



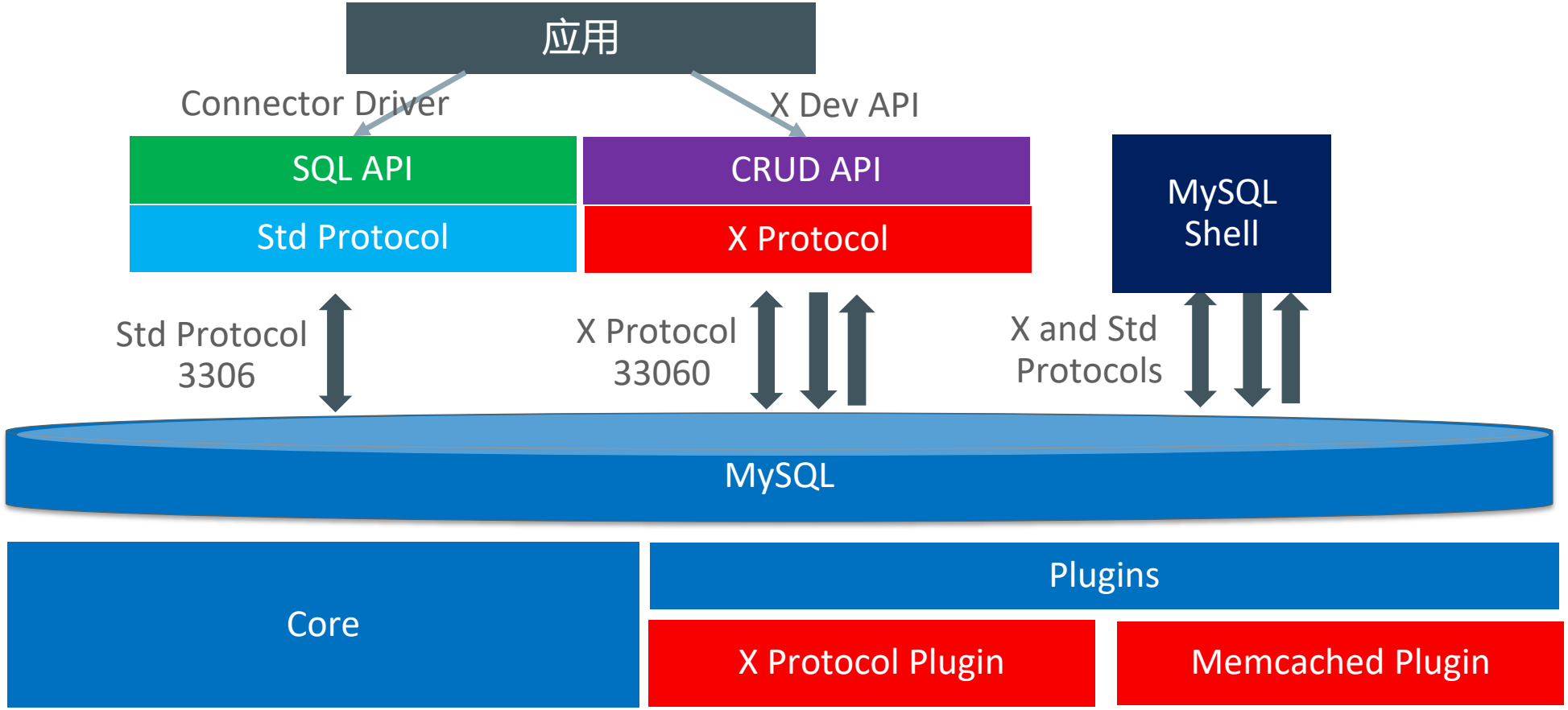
新的MySQL架构



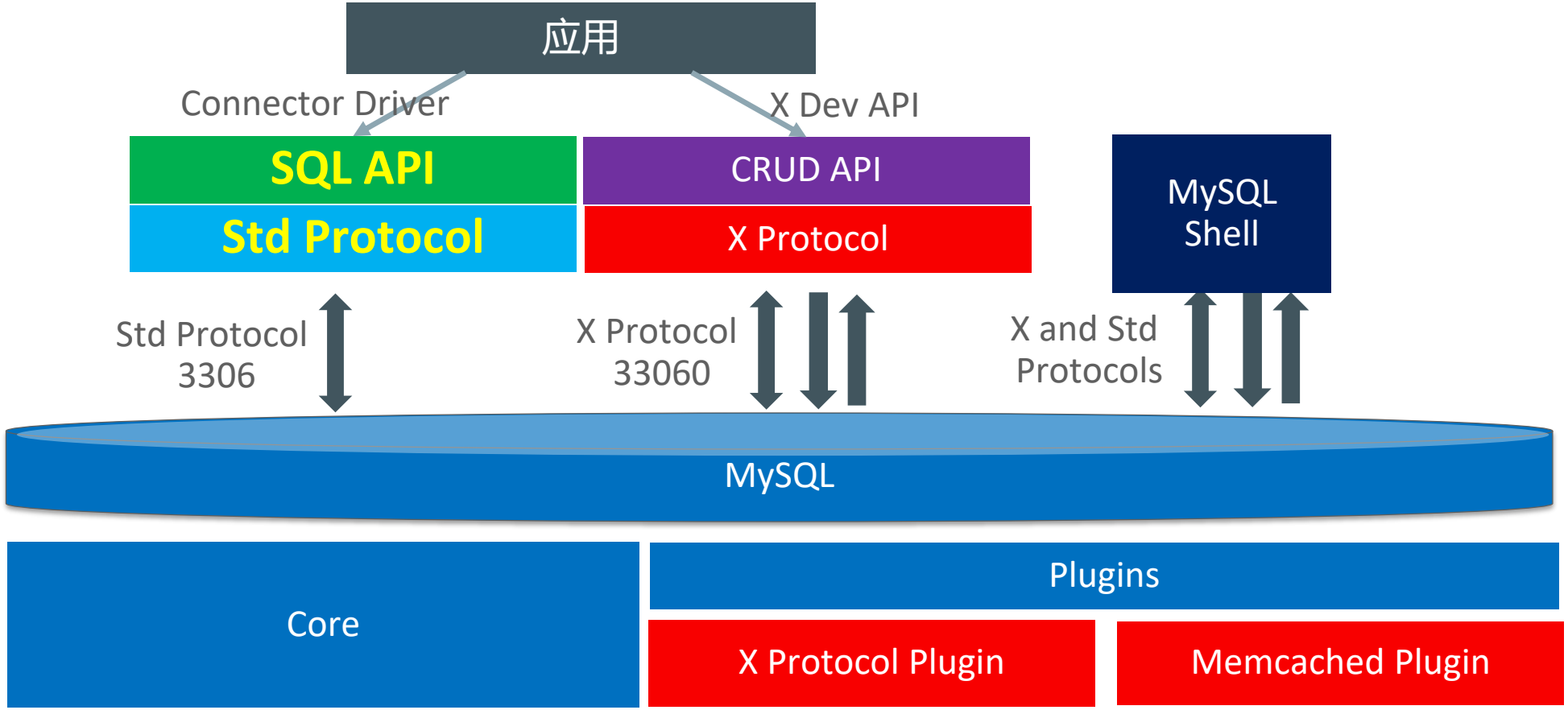
MySQL中SQL介面加入新的JSON的型別和处理

- New JSON data type
- New JSON function
- Indexing on JSON data

新的MySQL架构– SQL API



新的MySQL架构– SQL API



例如: CREATE 和 INSERT

```
CREATE TABLE t1 (data JSON);
INSERT INTO t1(data) VALUES
 (‘{ "series": 1}’), (‘{ "series": 7}’),  (‘{ "series": 3}’),
 (‘{ "series": 4}’), (‘{ "series": 10}’), (‘{ "series": 2}’),
 (‘{ "series": 6}’), (‘{ "series": 5}’),  (‘{ "series": 8}’),
 (‘{ "series": 11}’);
```

JSON Comparator: 例如

```
SELECT * FROM t1 WHERE
  json_extract(data,"$.series") >= 7 AND
  json_extract(data,"$.series") <=10;
```

```
+------------------+
| data             |
+------------------+
| {"series": 7}    |
| {"series": 10}   |
| {"series": 8}    |
+------------------+
```

处理JSON 数据的新功能: 路径

[[[database.]table.]column]$<path spec>

- Path expr
  - [[[database.] table.] field]
  - $
  - .identifier
  - [array]
  - .* and [*]
  - **
- 例如
  - db.phonebook.data (future extension)
  - document’s root
  - $.user.address.street
  - $.user.addresses[2].street
  - $.user.addresses[*].street
  - $.user**.phone

处理JSON 数据的新功能: 函式

- Info
  - JSON_VALID()
  - JSON_TYPE()
  - JSON_KEYS()
  - JSON_LENGTH()
  - JSON_DEPTH()
  - JSON_CONTAINS_PATH()
- Modify
  - JSON_REMOVE()
  - JSON_APPEND()
  - JSON_SET()
  - JSON_INSERT()
  - JSON_REPLACE()

处理JSON 数据的新功能: 函式

- Create
  - JSON_MERGE()
  - JSON_ARRAY()
  - JSON_OBJECT()
- Get data
  - JSON_EXTRACT()
  - JSON_SEARCH()
- Helper
  - JSON_QUOTE()
  - JSON_UNQUOTE()

例如: UPDATE + join

```
UPDATE t1, t2
  SET t1.data=
    JSON_INSERT(t1.data,"$.inverted",
              11 – JSON_EXTRACT(t2.data,"$.b_series[0]"))
WHERE
  JSON_EXTRACT(t1.data, "$.series") =
    JSON_EXTRACT(t2.data,"$.b_series[0]");
```

为JSON数据建索引

- 用Functional Indexes, Luke 
- 支持STORED 和VIRTUAL 虚字段

```
CREATE TABLE t1
   (data JSON, id INT AS (JSON_EXTRACT(data,”$.id”)) STORED,
   PRIMARY KEY(id));
ALTER TABLE t1
   ADD COLUMN id INT AS (JSON_EXTRACT(data, "$.series")),
   ADD INDEX id_idx (id);
```

为JSON数据建索引: 范例

SELECT data FROM t1 WHERE

- JSON_EXTRACT(data,"$.series") BETWEEN 3 AND 5;
- id BETWEEN 3 AND 5;

```
+--------------------------------+----+
| data                           | id |
+--------------------------------+----+
| {"series": 3, "inverted": 8}   |  3 |
| {"series": 4, "inverted": 7}   |  4 |
| {"series": 5, "inverted": 6}   |  5 |
+--------------------------------+----+
```

为JSON数据建索引: 范例

> EXPLAIN SELECT data FROM t1 WHERE **JSON_EXTRACT(data,"$.series")** BETWEEN 3 AND 5;

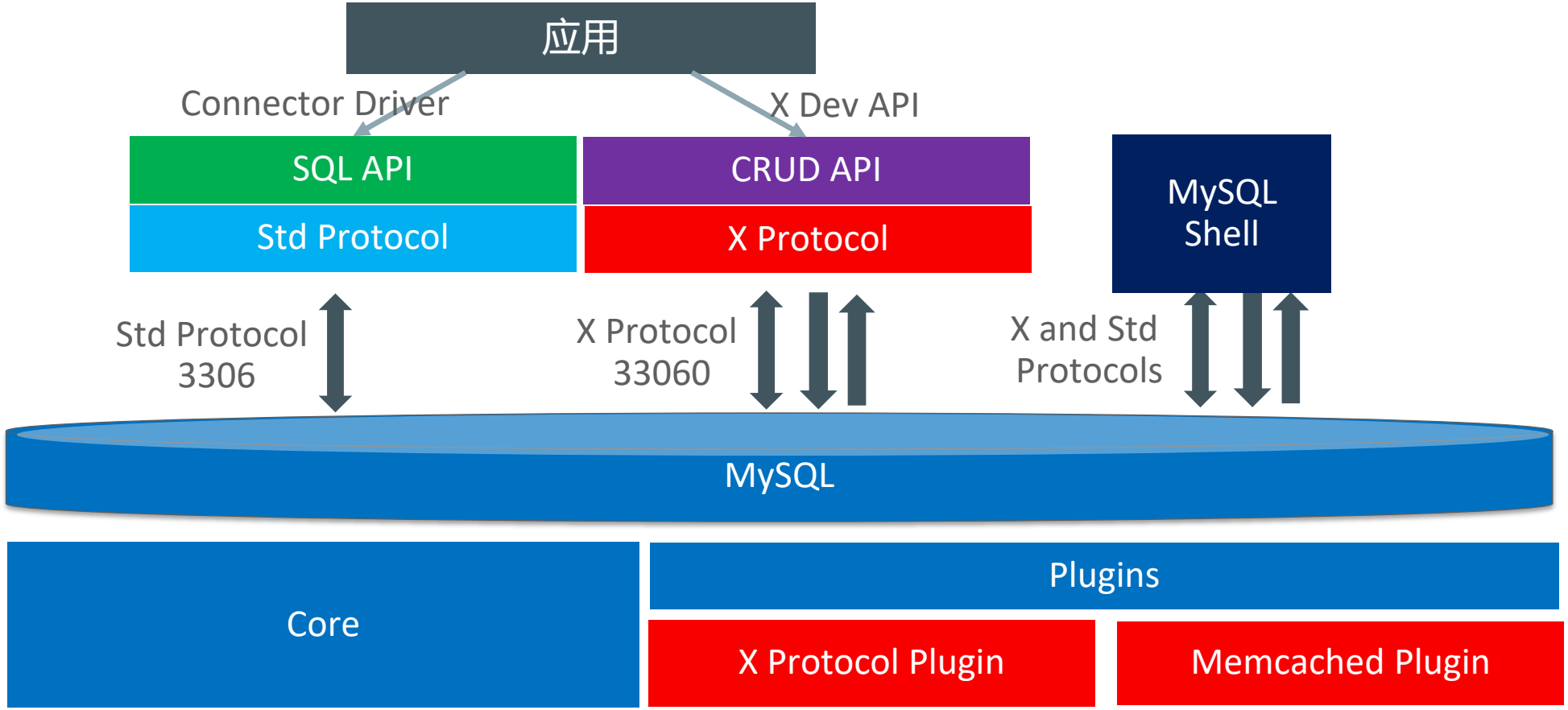
```
+----+-------------+-------+-------------+-------+...+-------------------------+
| id | select_type | table | partitions  | type  |   | Extra                   |
+----+-------------+-------+-------------+-------+...+-------------------------+
|  1 | SIMPLE      | t1    | NULL        | range |   | Using index condition |
+----+-------------+-------+-------------+-------+...+-------------------------+
```

```
select `test`.`t1`.`data` AS `data` from `test`.`t1`
where (`test`.`t1`.`id` between 3 and 5)
```

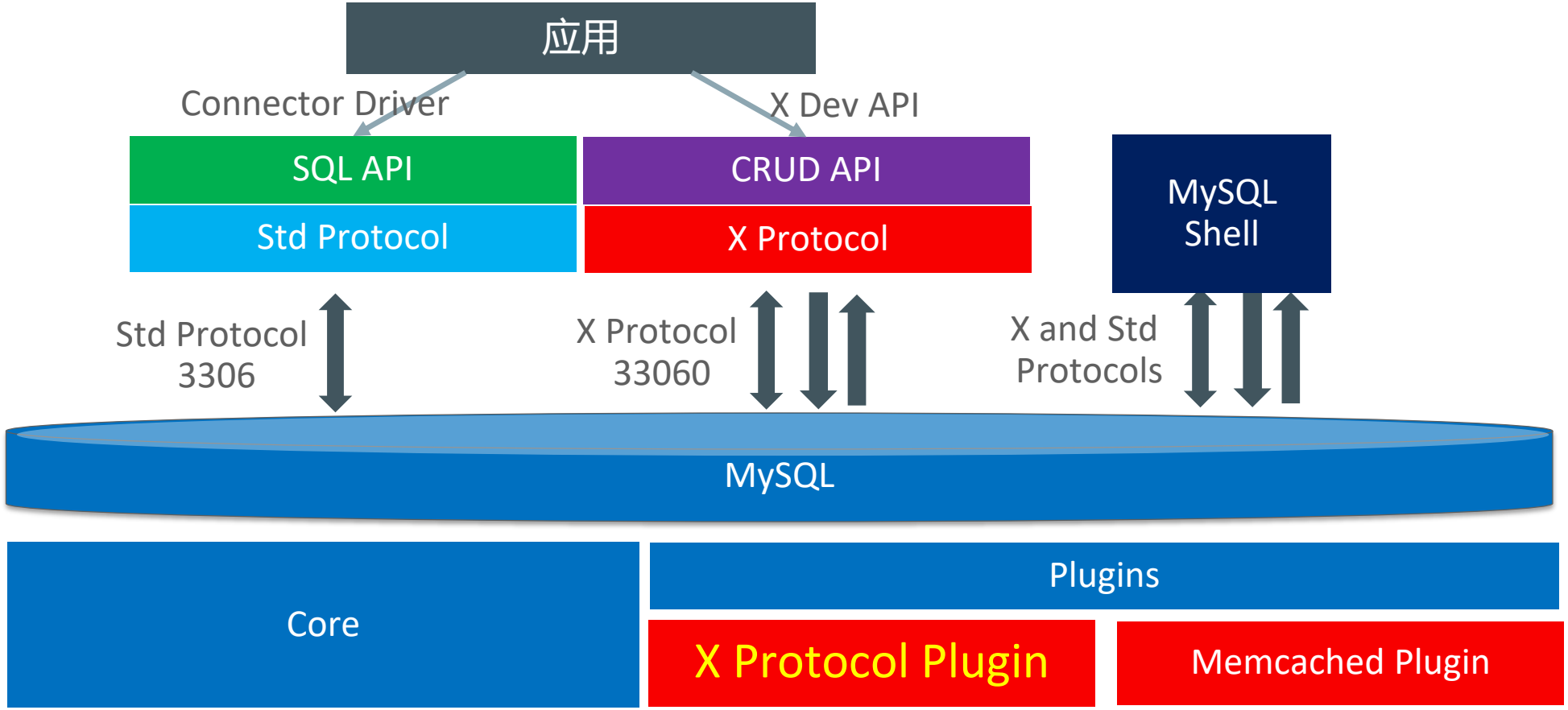
这还不够牛

我们您带来全套的NoSQL

新的MySQL架构– mysqlx plugin



新的MySQL架构– mysqlx plugin



新的! MySQL Server Plugin

让Connectors能使用Document APIs

- 设计成具创新的API能快速开发新的Protocol
- 支持新的protocol
  - 有CRUD和其他新加入的介面
- 包括Performance Schema加入新的instruments以协助监督
- 和标准的SQL APIs一起运作
- 运行于新的端口- 33060

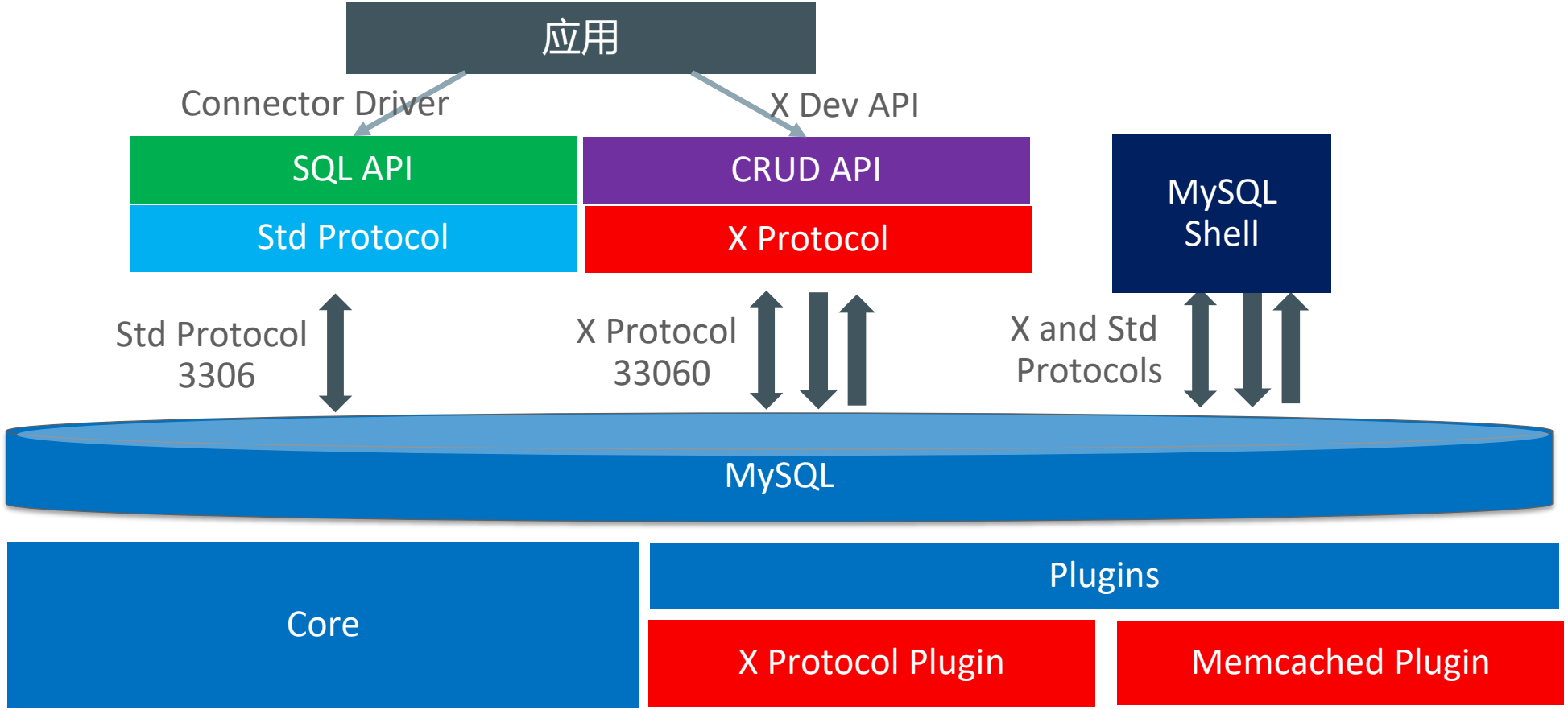
在数据库安装mysqlx 插件

- 在MySQL 5.7.12以后的版本中包中可找到该插件
  - 在lib/plugin/mysqlx.so(或mysqlx.dll)

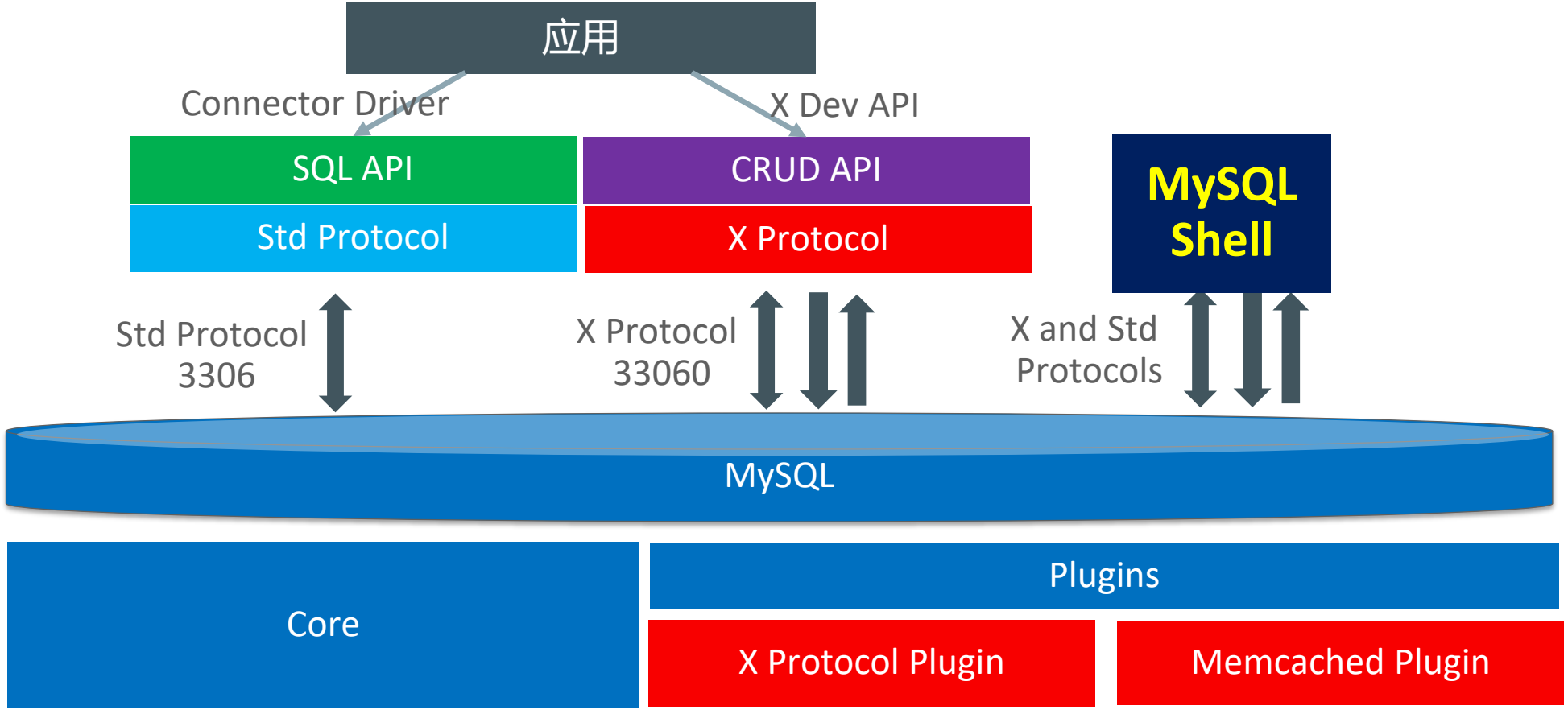
```
INSTALL PLUGIN mysqlx  SONAME 'mysqlx.so';
```

- 安装MySQL Shell
  - 自http://dev.mysql.com/downloads/shell/下载MySQL Shell
  - 或自MySQL Repository安装MySQL Shell
  - 执行MySQL Shell – ex: mysqlsh --uri mysqlxsys@localhost:33060/world_x

新的MySQL架构 – MySQL Shell



新的MySQL架构 – MySQL Shell



新的! MySQL Shell做管理和临时性的操作

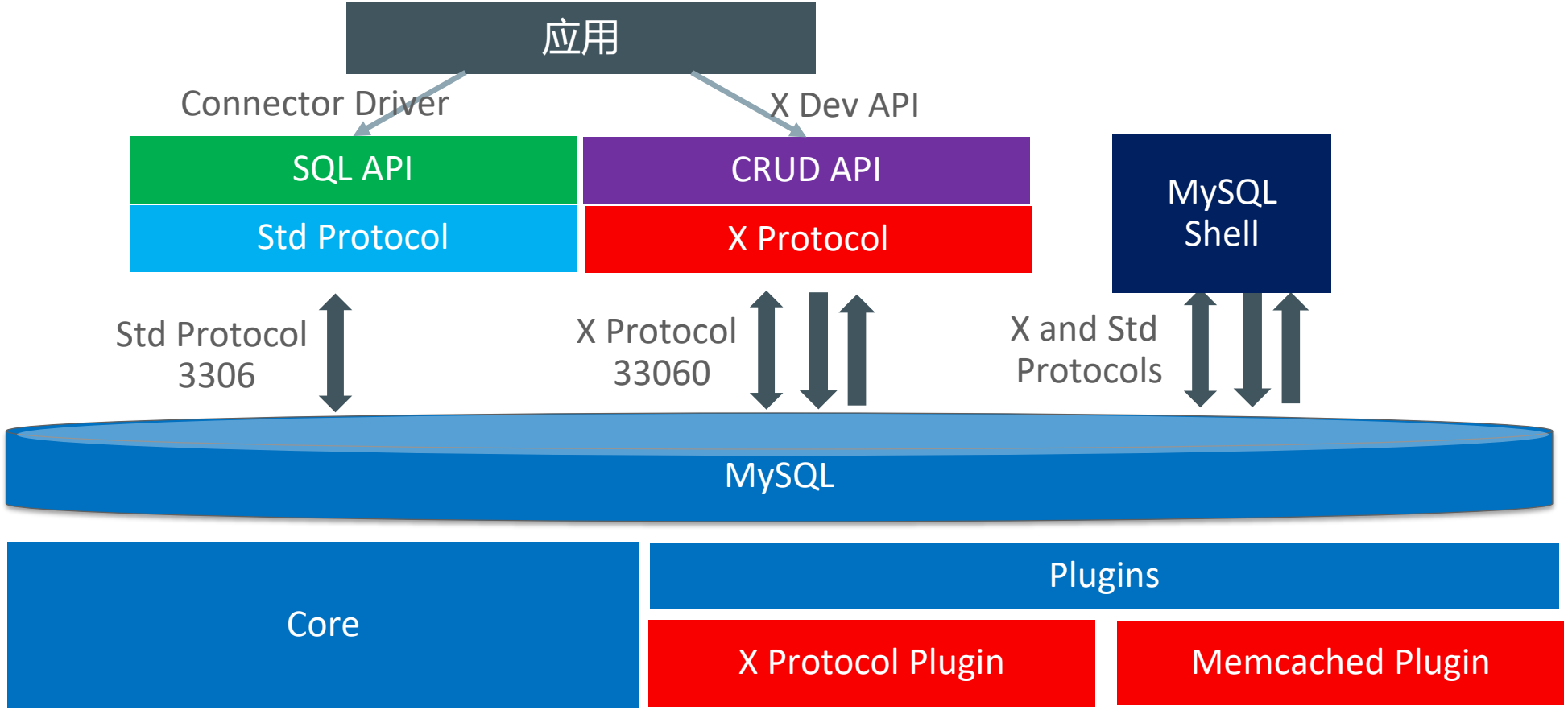
Global Sessions

- Stored Sessions – for 1 to Many servers
- 每一个Session可以支持多个Connections
- 支援三种新的Session types
  - Xsession – CRUD only session (Connected Server is “Abstracted”)
  - Node – SQL and CRUD (Connected Server is “Fixed”)
  - Standard MySQL session – SQL (Connected Server is “Fixed”)
- 三种语法 – JavaScript, Python, SQL
- Optional Session Logging
  - Set Logging Level, Verify state, Review history

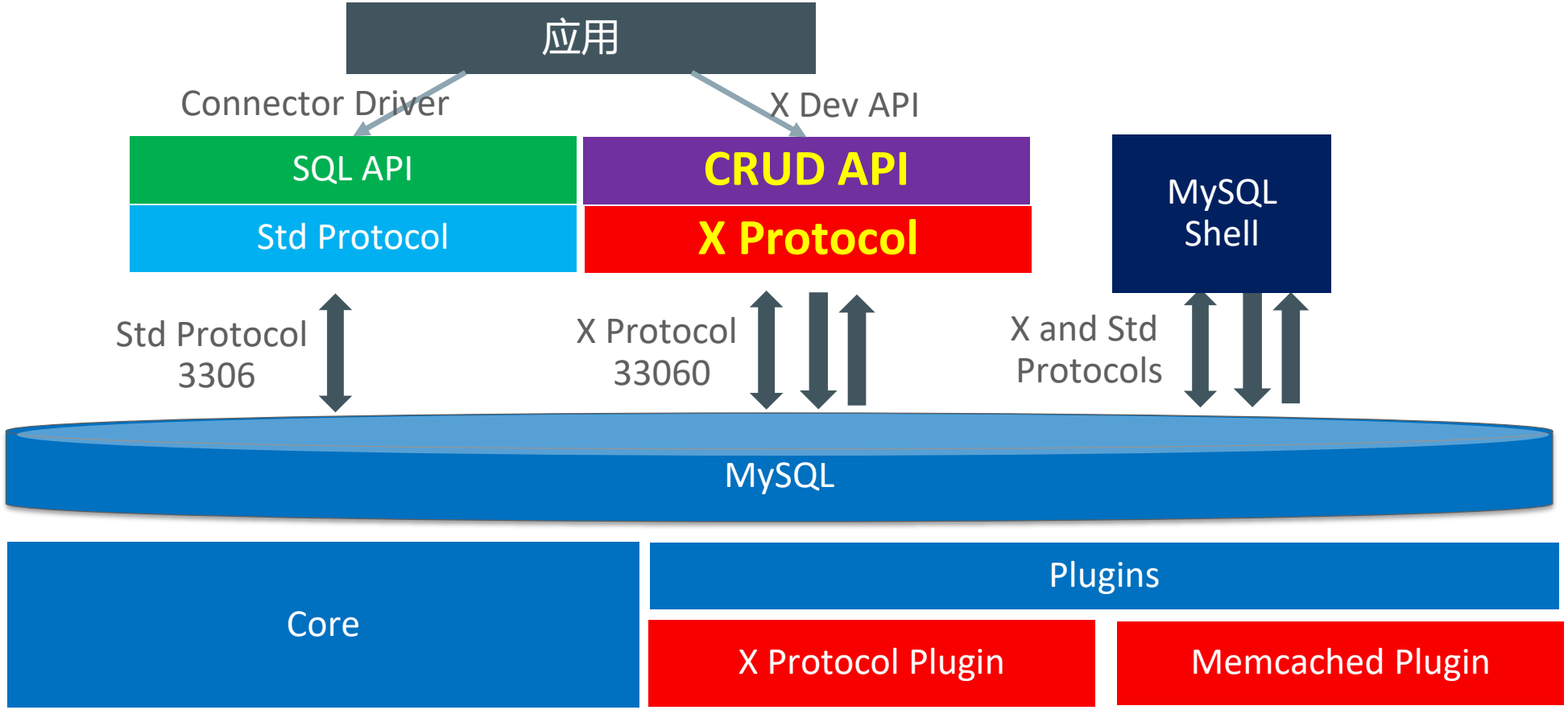
MySQL Shell的操作

- 连线
  - mysql-js> \C root@127.0.0.1/mysqlnews
- Schema的操作
  - mysql-js > \u mysqlnews
  - mysql-js > db
- 查当前的Schema有那些Collections?
  - mysql-js > db.getCollections()
- 查指定的Collection中有那些Document
  - mysql-js > db.news.find()
- 筛选Document
  - mysql-js > db.news.find(“introduction like ‘%8.0%’”)

新的MySQL架构



新的MySQL架构 – X Dev API 和 Connector



新的! MySQL Connectors让前端应用能使用X Protocol

- 弹性
  - 包括JavaScript, **Node.js**, Python, C#, C++, Java
  - 促成快速而穩定的演进
  - 用 SQL, CRUD APIs – Document 和 Relational, 或 “两者都有”
    - 所有都加在原有的 API 之上
- 支持
  - Concurrency, Asynchronous, Pipeline, Batch, ...
- 支持新的现代语言能力和Framework
  - Method Chaining, Promises, Asynchronous constructs, JSON results

新的! MySQL Connector 新特性,

新的 API

- 新的CRUD (Create, Read, Update, Delete) 即S(Search)CRUD
- 不仅是 CRUD – 还有新的SQL APIs
- 加上 新的 MySQL Connector/Node.js !
- 新的概念和能力
  - Sessions
  - Method Chaining
  - Optimized Network Round Trips
    - Single Round trip for Prepare/Execute, Streaming Inserts, Bulk Statements (16MB chunks)

MySQL Documents 和Collections

- Documents 存于collection中
  - 这些documents 有一个共同的目的
  - Collection可有一个以上索引
  - 每个collection有一个唯一的名称
- 存于单一个schema中
- 在collection内可以:
  - Add(), Find(), Modify(), 和 Remove() - JSON **documents**
- Collection可以被
  - Create(), List(), Drop()

Collection Search – find(), bind(), fields()

- 支持多种搜寻运算元
  - ||, &&, XOR, IS, NOT, BETWEEN, IN, LIKE, !=, <>, >, >=, <, <=, &, |, <<, >>, +, -, *, /, ~, %.
- 搜寻范例
  - db.CountryInfo.find("GNP > 500000 and demographics.Population < 100000000")
  - db.CountryInfo.find("GNP*1000000/demographics.Population > 30000")
- 动态绑定值- bind()
  - db.CountryInfo.find("Name = :country").bind("country", "Italy")
- Project Results – fields() –传回特定的值
  - db.CountryInfo.find("Name = :country").bind("country", "Italy")

MEAN 范例

Access Document based Database from Node.js - preparing

- 下载及安装Connector/Node.js
  - $ **npm install mysql-connector-nodejs-1.0.6.tar.gz**
- 安装Angular.js 和 Express
  - $ npm install angular@1.5.8 angular-route@1.5.8
  - $ npm install express-generator –g
- 以Express建立框架
  - $ express **mysql-news** –ejs # use ejs as a template instead of jade (default)
- 安装上列所建的模板相依部份
  - $ cd **mysql-news** && npm install

安装好MEAN和Connector/Node.js后,您可以得到的结构

```
mysql-news/
├── app.js
├── mysql-connector-nodejs-1.0.6.tar.gz
├── package.json
├── bin/
│   └── www
├── data/
├── models/
├── node_modules/
├── public/
│   ├── images/
│   ├── javascripts/
│   └── stylesheets/
│   └── style.css
├── routes/
│   ├── index.js
│   └── users.js
└── views
    ├── error.ejs
    └── index.ejs
```

在Node.js的Template批次载入数据

- bin/www之中加入:
  - 载入数据和XDevAPI library:

```
//设模块相依程序,建立HTTP Server
var app = require('../app');
var debug = require('debug')('mysql-news:server');
var http = require('http');
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);
var server = http.createServer(app)
// 载入内含初始代数据库的JSON文件
var initialData = require('../data/news.json');
//载入XDevAPI library
var mysql = require('@mysql/xdevapi');
```

  - Emitter

```
var eventEmitter = new EventEmitter();
eventEmitter.once('initDb', initializeDatabase);
eventEmitter.emit('initDb');
```

  - Initialize Database

```
function initializeDatabase() {
configureDataBase(function (resultMsg) {
// configureDatabase中真正建Schema和载入数据
  server.listen(port);
   server.on('error', onError);
   server.on('listening', onListening);
   //列示数据库中的数据
});
```

在configureDatabase中建Schema和载入数据

```
function configureDataBase(callback) {
mysql.getSession({
// 连结数据库
    host: 'localhost',
    port: '33060',
    dbUser: 'root',
    dbPassword: 'Welcome1'
  }).then(function (session) {
    //取得Schema
    var schema = session.getSchema('mysqlNews');
    schema.existsInDatabase().then(function (exists) {
```

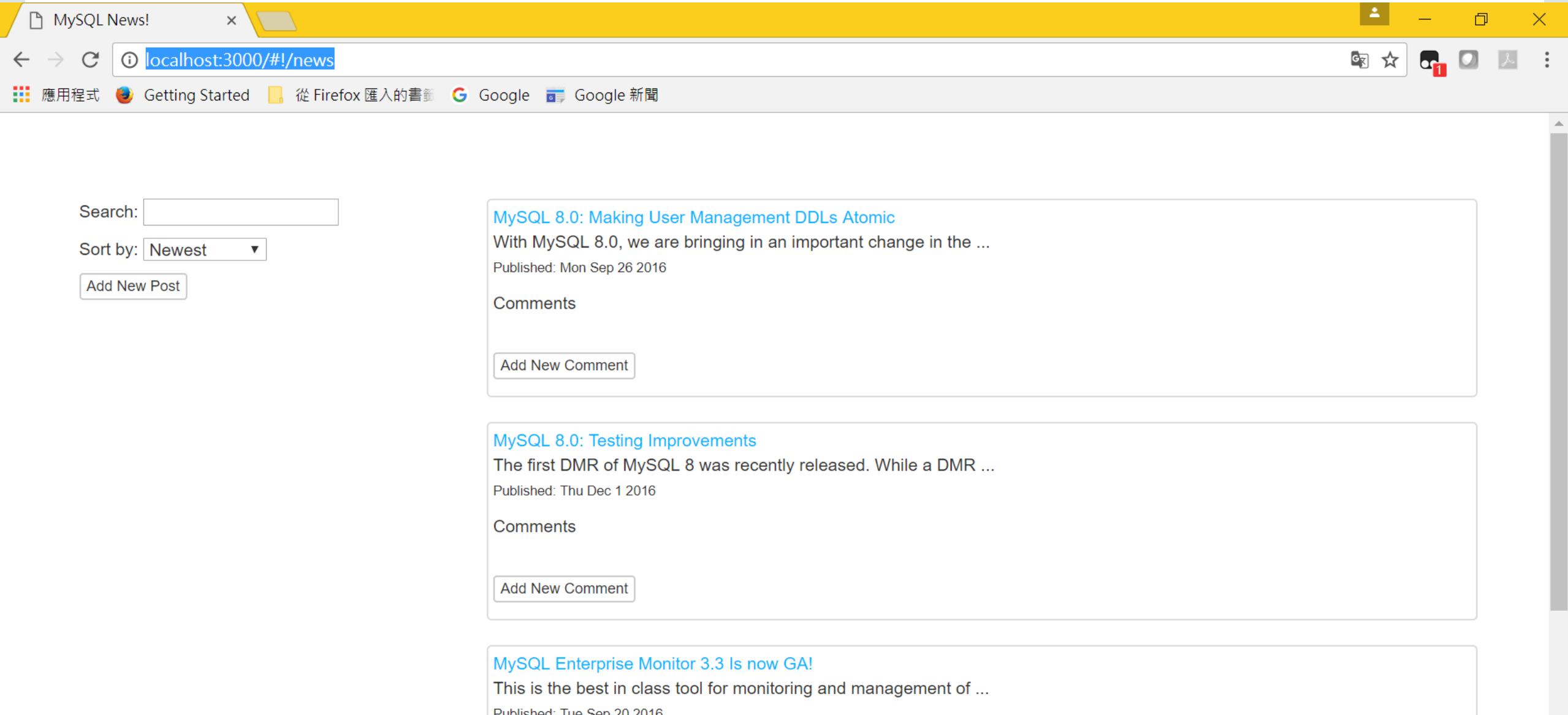
```
— 如果mysqlNews Schema不存在则建立之,并插入数据
if (!exists) {
  //建立Schema
  session.createSchema('mysqlNews').then(function (newSchema) {
  Promise.all([
    //建立新的Collection - news
    newSchema.createCollection('news').then(function (newsColl) {
     // 加入初始数据到数据库
     newsColl.add(initialData).execute().then(function (newsAdded) {
       var rowsAffected = newsAdded.getAffectedItemsCount();
       if (rowsAffected <= 0) {
         console.log('No news Added');
       }
       else {
         console.log(rowsAffected + ' news Added');
       }
     }).catch(function (err) {
```

Node.js template中以JSON文件批次导入数据

- Data目录之下:
  - news.sjon

```
[
{
"title": "MySQL 8.0: Testing Improvements",
"link": "http://mysqlserverteam.com/mysql-8-0-testing-improvements/",
"introduction": "The first DMR of MySQL 8 was recently released. While a DMR ...",
"published": "Thu Sep 29 2016",
"comments":[]
},
…
```

鏈接指向本地3000端口看到的完整MEAN范例



在MySQL Shell 看到的结果

MySQL Shell

```
mysql-js> db.news.find();
[
    {
        "_id": "05353cf6-72e1-1f10-d400-c933c4dd",
        "comments": [],
        "introduction": "With MySQL 8.0, we are bringing in an important change in the ...",
        "link": "http://mysqlserverteam.com/mysql-8-0-making-user-management-ddls-atomic/",
        "published": "Mon Sep 26 2016",
        "title": "MySQL 8.0: Making User Management DDLs Atomic"
    },
    {
        "_id": "45766589-fb29-ad95-5366-17a28bb0",
        "comments": [],
        "introduction": "In this blog post we will present a first look at the ...",
        "link": "http://mysqlhighavailability.com/an-overview-of-the-group-replication-performance/",
        "published": "Wed Sep 21 2016",
        "title": "An overview of the Group Replication performance"
    },
    {
        "_id": "62337aed-c9e8-cf4b-1de0-82cd29f1",
        "comments": [],
        "introduction": "The first DMR of MySQL 8 was recently released. While a DMR ...",
        "link": "http://mysqlserverteam.com/mysql-8-0-testing-improvements/",
        "published": "Thu Dec 1 2016",
        "title": "MySQL 8.0: Testing Improvements"
    },
    {
        "_id": "aac9d0fe-76c6-2729-f216-7c1fb202",
        "comments": [],
        "introduction": "This is the best in class tool for monitoring and management of ...",
        "link": "https://blogs.oracle.com/MySQL/entry/mysql_enterprise_monitor_3_3",
        "published": "Tue Sep 20 2016",
        "title": "MySQL Enterprise Monitor 3.3 Is now GA!"
    }
]
```

在MySQL 客户端看到的结果

MySQL 5.7 Command Line Client

```
mysql> use mysqlnews;
Database changed
mysql> show tables;
+--------------------+
| Tables_in_mysqlnews |
+--------------------+
| news               |
+--------------------+
1 row in set (0.00 sec)

mysql> select * from news\G
*************************** 1. row ***************************
doc: {"_id": "05353cf6-72e1-1f10-d400-c933c4dd", "link": "http://mysqlserverteam.com/mysql-8-0-making-user-management-ddls-atomic/", "title": "MySQL 8.0: Making User Manage
ment DDLs Atomic", "comments": [], "published": "Mon Sep 26 2016", "introduction": "With MySQL 8.0, we are bringing in an important change in the ..."}
_id: 05353cf6-72e1-1f10-d400-c933c4dd
*************************** 2. row ***************************
doc: {"_id": "45766589-fb29-ad95-5366-17a28bb0", "link": "http://mysqlhighavailability.com/an-overview-of-the-group-replication-performance/", "title": "An overview of the
Group Replication performance", "comments": [], "published": "Wed Sep 21 2016", "introduction": "In this blog post we will present a first look at the ..."}
_id: 45766589-fb29-ad95-5366-17a28bb0
*************************** 3. row ***************************
doc: {"_id": "62337aed-c9e8-cf4b-1de0-82cd29f1", "link": "http://mysqlserverteam.com/mysql-8-0-testing-improvements/", "title": "MySQL 8.0: Testing Improvements", "comments
": [], "published": "Thu Dec 1 2016", "introduction": "The first DMR of MySQL 8 was recently released. While a DMR ..."}
_id: 62337aed-c9e8-cf4b-1de0-82cd29f1
*************************** 4. row ***************************
doc: {"_id": "aac9d0fe-76c6-2729-f216-7c1fb202", "link": "https://blogs.oracle.com/MySQL/entry/mysql_enterprise_monitor_3_3", "title": "MySQL Enterprise Monitor 3.3 Is now
GA!", "comments": [], "published": "Tue Sep 20 2016", "introduction": "This is the best in class tool for monitoring and management of ..."}
_id: aac9d0fe-76c6-2729-f216-7c1fb202
4 rows in set (0.01 sec)

mysql>
```

和MySQL其他功能结合

- 对事务的支援
- MVCC和ACID
- 高可用方案
- 管理及监督介面
  - MySQL Enterprise Monitor
  - MySQL Enterprise Backup
  - MySQL Enterprise Audit
  - MySQL Firewall
- 数据库分片

需要您的参与

- 试用并将您的建议反馈给我们(bugs.mysql.com)
- 加入您的framework中
  - Java
  - MEAN - M(ySQL)E(xpress)A(agular.js)N(ode.js) 和更多的框架
  - C#/.Net
  - C++
  - Python

Resources

Topic	Link(s)
MySQL as a Document Database	http://dev.mysql.com/doc/refman/5.7/en/document-database.html
MySQL Shell	http://dev.mysql.com/doc/refman/5.7/en/mysql-shell.html http://dev.mysql.com/doc/refman/5.7/en/mysqlx-shell-tutorial-javascript.html http://dev.mysql.com/doc/refman/5.7/en/mysqlx-shell-tutorial-python.html
X Dev API	http://dev.mysql.com/doc/x-devapi-userguide/en/
X Plugin	http://dev.mysql.com/doc/refman/5.7/en/x-plugin.html
MySQL JSON	http://mysqlserverteam.com/tag/json/ https://dev.mysql.com/doc/refman/5.7/en/json.html https://dev.mysql.com/doc/refman/5.7/en/json-functions.html
Demo database – world_x	http://downloads.mysql.com/docs/world_x-db.zip
Learn MySQL from Oracle	https://education.oracle.com/pls/web_prod-plq-dad/ou_product_category.getPage?p_cat_id=159


ORACLE®

MySQL使用JSON案例-Qunar

行政套房

面积62㎡ 位于4-20层
大床 (可以加床)
独立卫浴 有窗

酒店直销	行政套房-双早(大床)	双早	取消扣款	¥1878 在线付¥1942减¥64	预付 预订 支持礼品卡
酒店直销	行政套房(提前30天预订)(双人入住)(大床)	双早	取消扣款	¥2006	担保 预订
Ctrip 携程	行政套房(双人入住)(大床)	双早	限时取消	¥2507	预订
酒店直销	Executive Suite	无早	限时取消	¥2502	预付 预订 支持礼品卡
嫣然旅行社	行政套房(提前7天预订)[双早](大床)	双早	取消扣款	低价 ¥1750 在线付¥1752减¥2	预付 预订 支持礼品卡

查看其他12条报价

取消规则

预订人因自身原因或因其他非因法定原因要求变更或取消，将扣收全额预付款。（"法定原因"以《合同法》和《消费者权益保护法》规定为准，一般包括自然灾害、政府征用、酒店方过错等导致预订人不能正常入住的情形。）

如订单无法确认，房费全额退还

保险

6月已为购险用户挽回72万元损失

☑ 建议购买《因故取消险》 ¥167.50 （总房费5%），因故未入住可获得90%房费赔付，避免您的房费损失，购买前请仔细阅读理赔条款

☑ 建议购买 《出行意外险》 ¥23 为您的出行保驾护航。意外伤害、急病医疗、航班延误、财产丢失等超值综合保障，购买前请仔细阅读理赔条款

份数 1 ¥23 (¥23×1)

保额 ¥23/份 最高保额20万

姓名 周彦伟 身份证 证件号码

北京中国大饭店

北京市朝阳区建国门外大街一号

房型	超豪华客房(提前7天预订)[双早]
床型	大床
宽带	无宽带
WIFI	无wifi
最大可住	2人

需在线支付

房费	¥ 3354.00
因故取消险	¥ 167.50
出行意外险	¥ 23.00
房费优惠	减去 ¥ 4.00
总计	¥3540.50

收起单价和早餐信息

08-24(四)	08-25(五)
¥ 1852.00 双早	¥ 1502.00 双早

预订须知

1. 该房型不支持加床
2. 目前北京全城禁烟，酒店均为无烟房。

qunar_order_store

- qunar_order_index
- qunar_order_detail

Field	Type
id	bigint(20) unsigned
order_id	bigint(20)
version	int(11)
section	varchar(16)
data	text
create_date	datetime
update_time	timestamp

- qunar_order_detail:data

- {"orderPay":{"createDate":"2015-03-13 00:00:01","updateTime":null,"orderId":100363989850,"payAmount":null,"realPayAmount":null,"refundAmount":null,"realRefundAmount":null,"guaranteeAmount":null,"guaranteeConfirmAmount":null,"guaranteeRevokeAmount":null,"paySubtractAmount":0,"payModeList":null,"status":"PAY","preAuthType":null,"actionRecorderStoreData":{"recorderStamps":[],"recorderMap":{}},"orderPayRecordList":[]},"cashPay":null,"promotionPay":null,"version":0}


```
{
  "orderPay": {
    "createDate": "2015-03-13 00:00:01",
    "updateTime": null,
    "orderId": 100363989850,
    "payAmount": null,
    "realPayAmount": null,
    "refundAmount": null,
    "realRefundAmount": null,
    "guaranteeAmount": null,
    "guaranteeConfirmAmount": null,
    "guaranteeRevokeAmount": null,
    "paySubtractAmount": 0,
    "payModeList": null,
    "status": "PAY",
    "preAuthType": null,
    "actionRecorderStoreData": {
      "recorderStamps": [],
      "recorderMap": {}
    },
```

现状与改进

- 使用自建索引
- 整体存取TEXT字段
- 编程解析JSON
- 未来用MySQL Doc Store解决