本文是作者在ACMUG 2016 MySQL年会上的演讲内容，版权归作者所有。

中国MySQL用户组（China MySQL User Group）简称ACMUG。
ACMUG是覆盖中国MySQL技术爱好者的一个技术社区，是Oracle User Group Community和MairaDB Foundation共同认可的MySQL技术社区。

我们关注MySQL，MariaDB，以及其他一切周边的开源数据库和开源工具，我们交流使用经验，推广开源技术，为开源贡献力量。

我们是开放社区，欢迎任何关注MySQL及其相关技术的人加入，我愿意跟其他任何技术组织和团体保持沟通和展开合作。

我们期望在我们的活动中大家都能以开心的、轻松的姿态交流技术，分享技术，形成一个良性循环，从而每个人都可以有一份收获。

ACMUG的口号：开源，开放，开心

关注ACMUG公众号，参与社区活动，交流开源技术，分享学习心得，一起共同进步。

MySQL As a Document Store - Develop NoSQL Applications with New Generation MySQL

杜修文
Ivan.Tu@Oracle.Com
MySQL GBU, Oracle

ORACLE

MySQL

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Today's Challenges

- Developers want to move faster
- Time to market has a premium value
- Rapid prototyping, iterate fast…
- Relational databases ask for schema up front
  - Potentially saving you time in the future
  - Less variations; less code to handle edge cases
  - Added cost with each schema change
- NoSQL databases do not ask for schema
  - Saving you time up front
  - But potentially adding operational costs over time
  - No cost per schema change

Today's Challenges (cont.)

- The Most popular NoSQL database uses similar data structures to relational databases (B+TREE)
  - Data structure influences key performance characteristics not data model
  - No inherent scalability advantages
  - Yet behind on key functionality
- Standing up multiple technologies adds complexity for operations teams

Why can you not…

- Have both schema-less and schema in the same technology stack?
- One that checks all the boxes of all stakeholders:

Developers:
[x] Schemaless
[x] Rapid Prototyping/Simpler APIs
[x] Document Model

Operations:
[x] Performance Management/Visibility
[x] Robust Replication, Backup, Restore
[x] Comprehensive Tooling Ecosystem

Business Owner:
[x] Don’t lose my data = ACID transactions
[x] Capture all my data = Extensible/Schemaless
[x] Products On Schedule/Time to Market = Rapid Development

One Database Many Models VS Many Databases Many Models

- One Extensible database
  - Do more with the MySQL database
  - Many can manage (with deep skills)
  - Stable
  - Cost-effective
  - Easy to move data between like database types
  - Fewer Drivers
  - Few Tools
  - SQL works, CRUD works
  - Operational and Analytical Together

- Many different databases
  - Requires larger skill repertoire, more complex development …
    - Harder to find deep skills
  - Many Drivers
  - Many Tools
  - More effort to share and exchange data
  - It's a lot more work
  - Operational and Analytical Separate

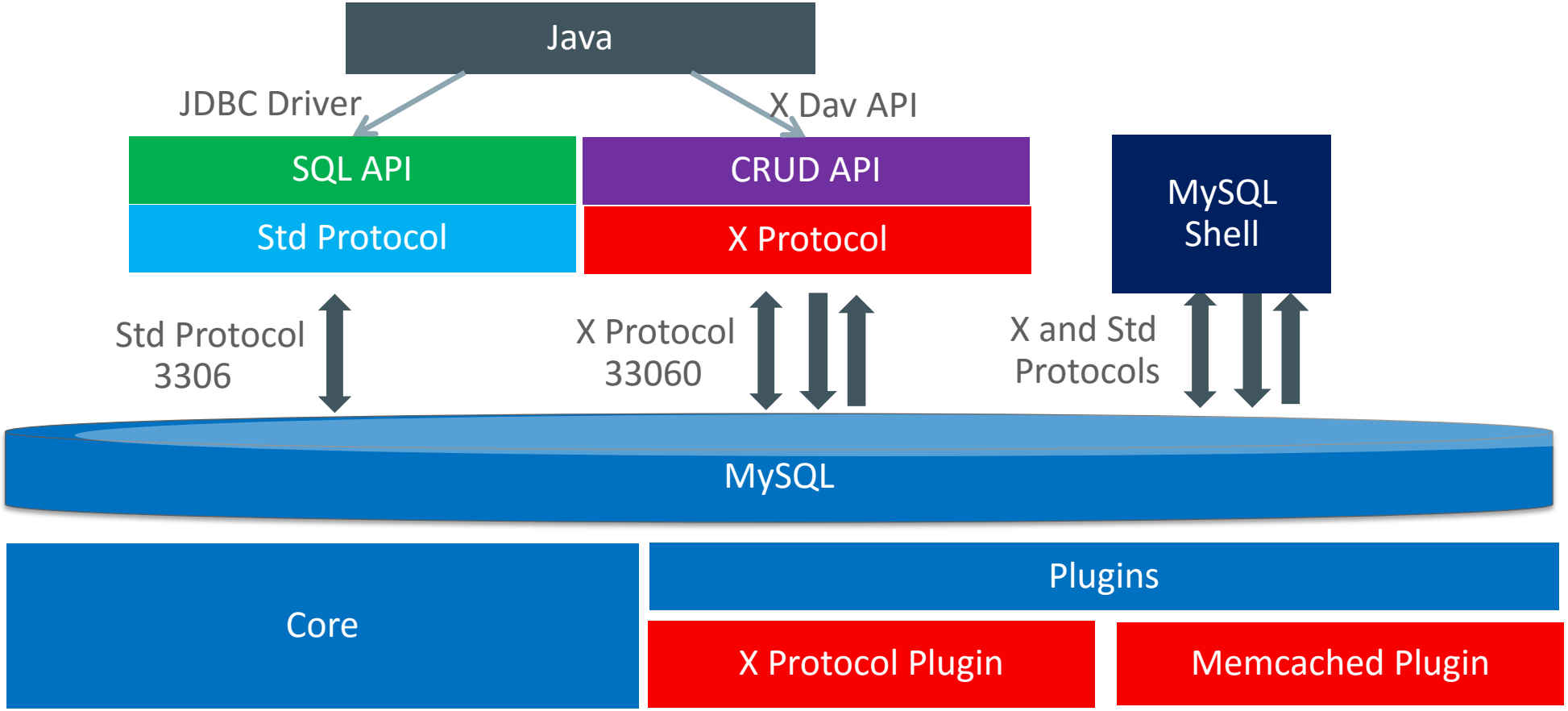
We <3 Schemaless

- MySQL as a Document Store (new!)
- All the existing features of MySQL
  - Replication
  - InnoDB
  - Performance Schema
- With the addition of schemaless
  - Documents using JSON
  - Easy to program CRUD APIs

Whats New?

- New Document APIs
  - SQL and NoSQL/Document CRUD APIs will cross all Connectors
    - For BOTH Collections of Documents and Relational Tables
  - Includes Node.js, Java, Python, C++, and .NET
- New Interactive Shell – with Javascript, Python, SQL modes
- New Server Features
  - Native JSON datatype and storage
  - Generated Columns with Indexing
  - 23 Native JSON functions
  - New X Protocol

Architecture

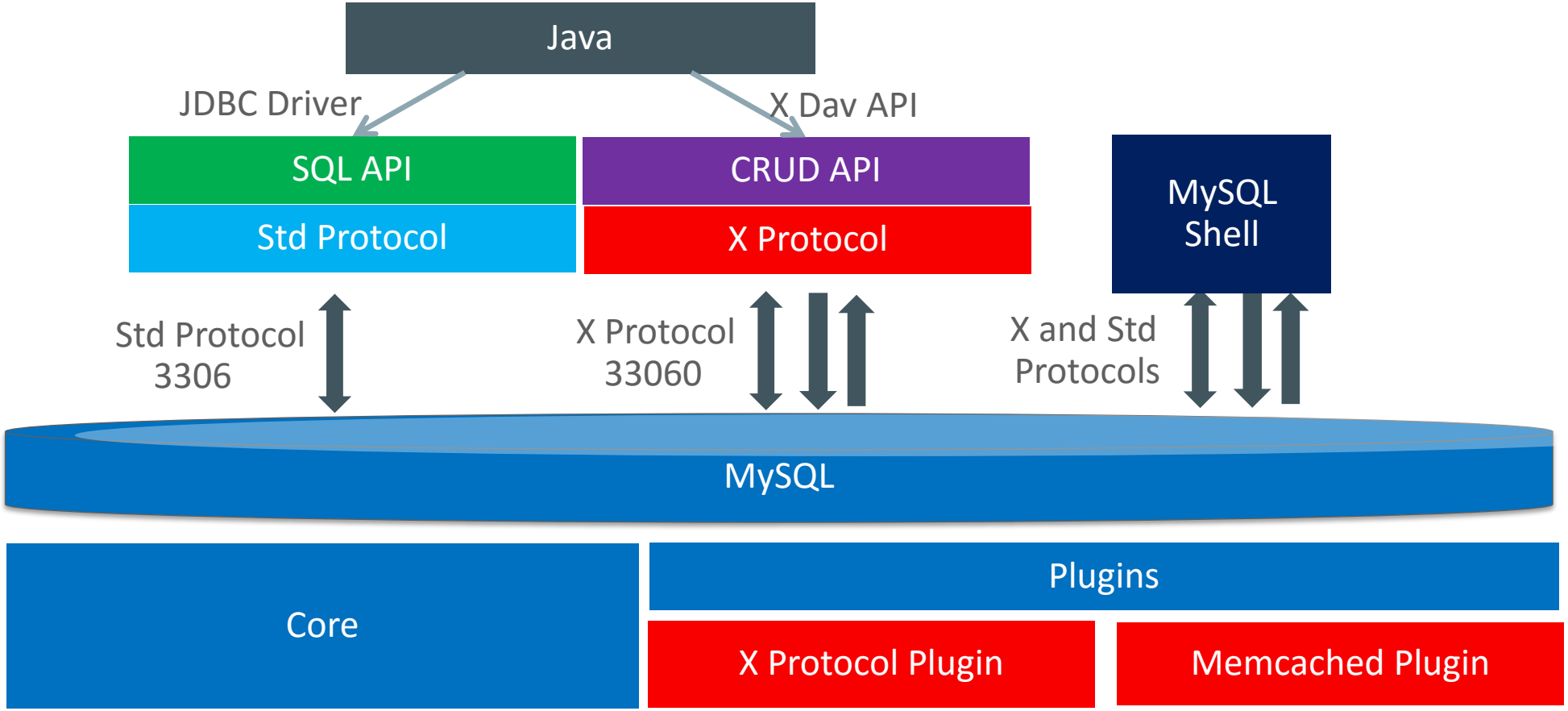


MySQL as a Document Store – a FULL Stack

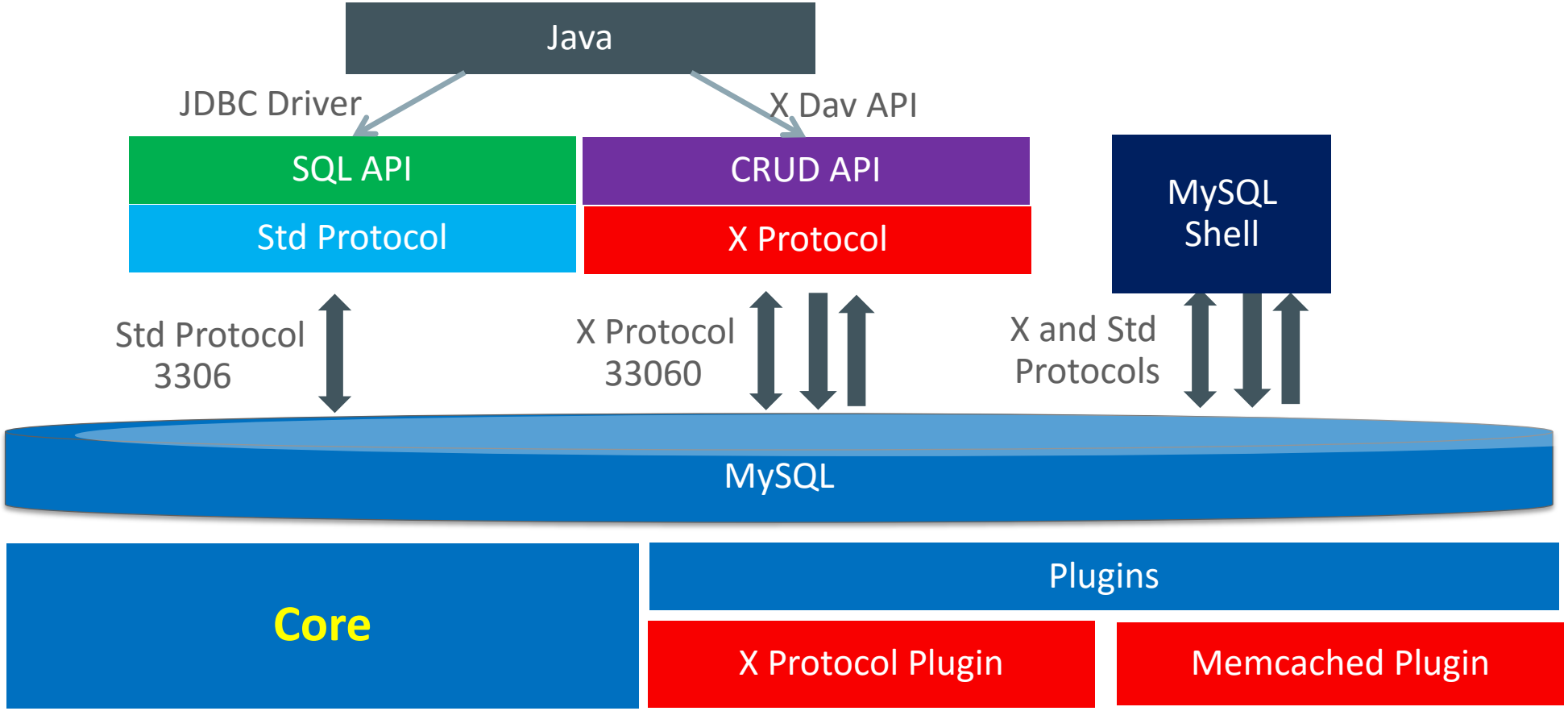
Store, Retrieve, Search and Managing JSON documents

- Native JSON Datatype, Indexes on JSON Documents
- Native Collections with Key Value Semantics
- Interactive Shell “MySQL Shell” – Javascript, Python, SQL modes
- Connectors include NoSQL CRUD APIs
  - Java, New NodeJS, NET, C++/C, Python
  - Method Chaining and Pipelining
  - Supports Combined Document and Relational

Architecture



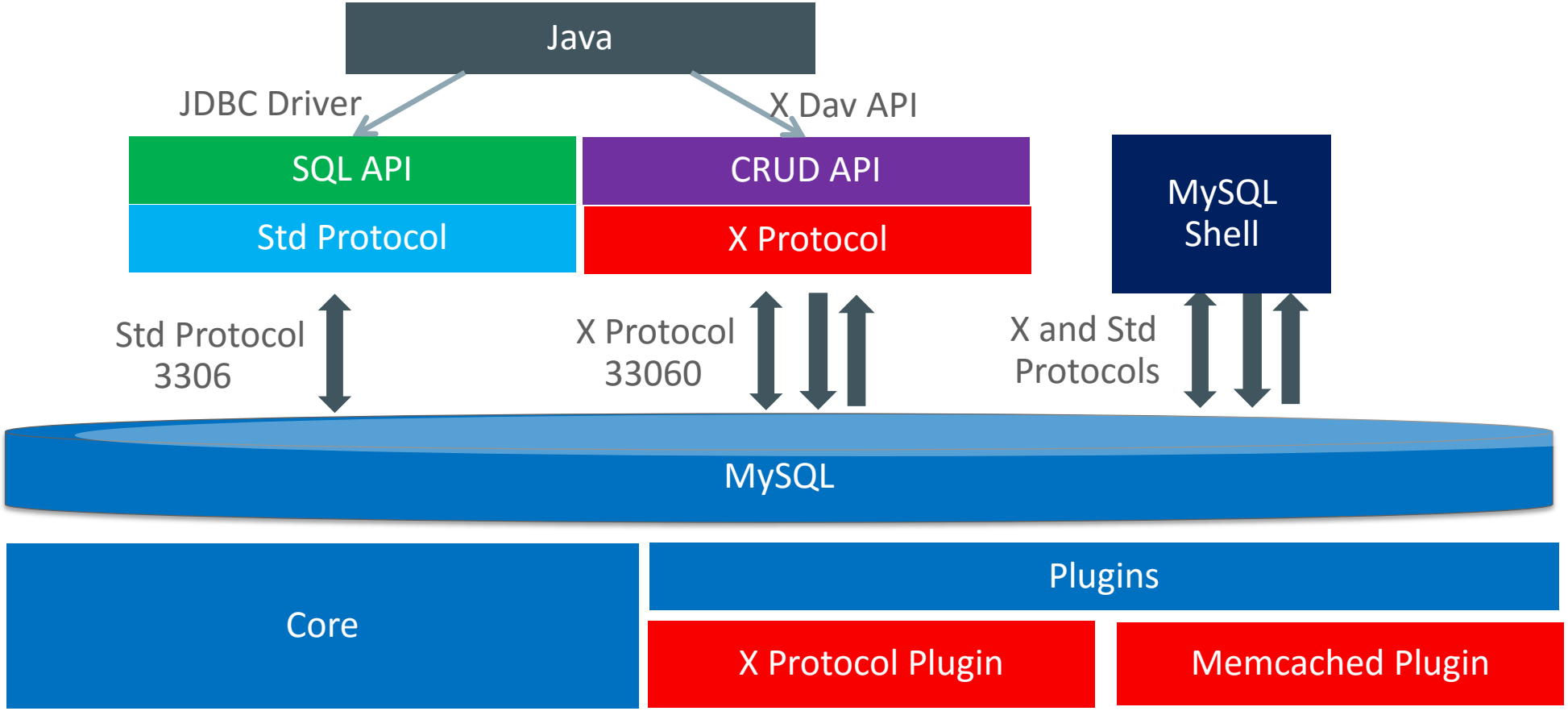
Architecture – Core, JSON



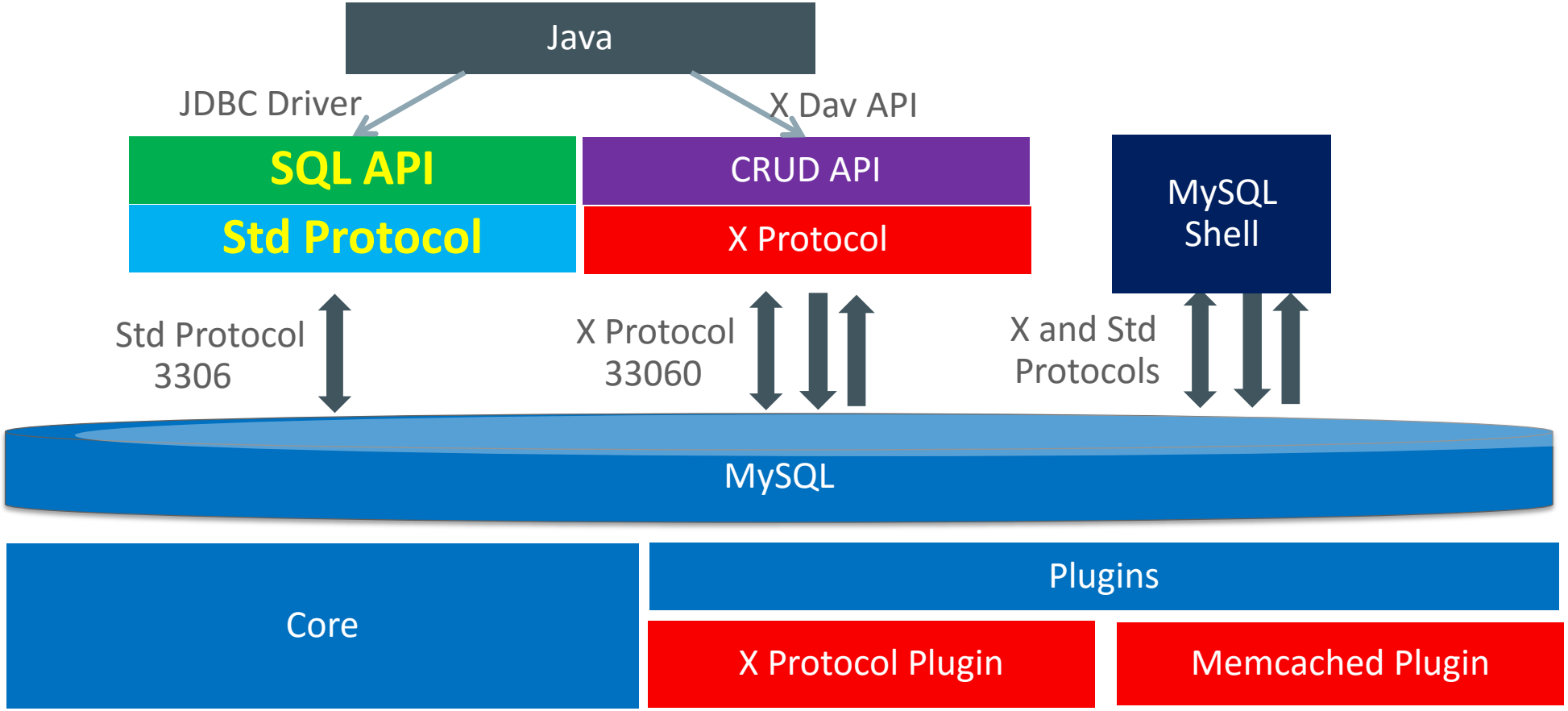
New JSON Data Type and Features on SQL Interface

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

Architecture– SQL Interface



Architecture– JSON Function and Data Type for SQL



Examples: CREATE and 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: example

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

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

New Functions to Handle JSON Data: Path

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

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

New functions to handle JSON data: Funcs

Note: Prior to 5.7.9 functions' prefix was JSN_

- 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()

New functions to handle JSON data: Funcs

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

Examples: 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]");
```

Indexing JSON data

- Use Functional Indexes, Luke ☺
- STORED and VIRTUAL types are supported

```
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);
```

Indexing JSON data: an example

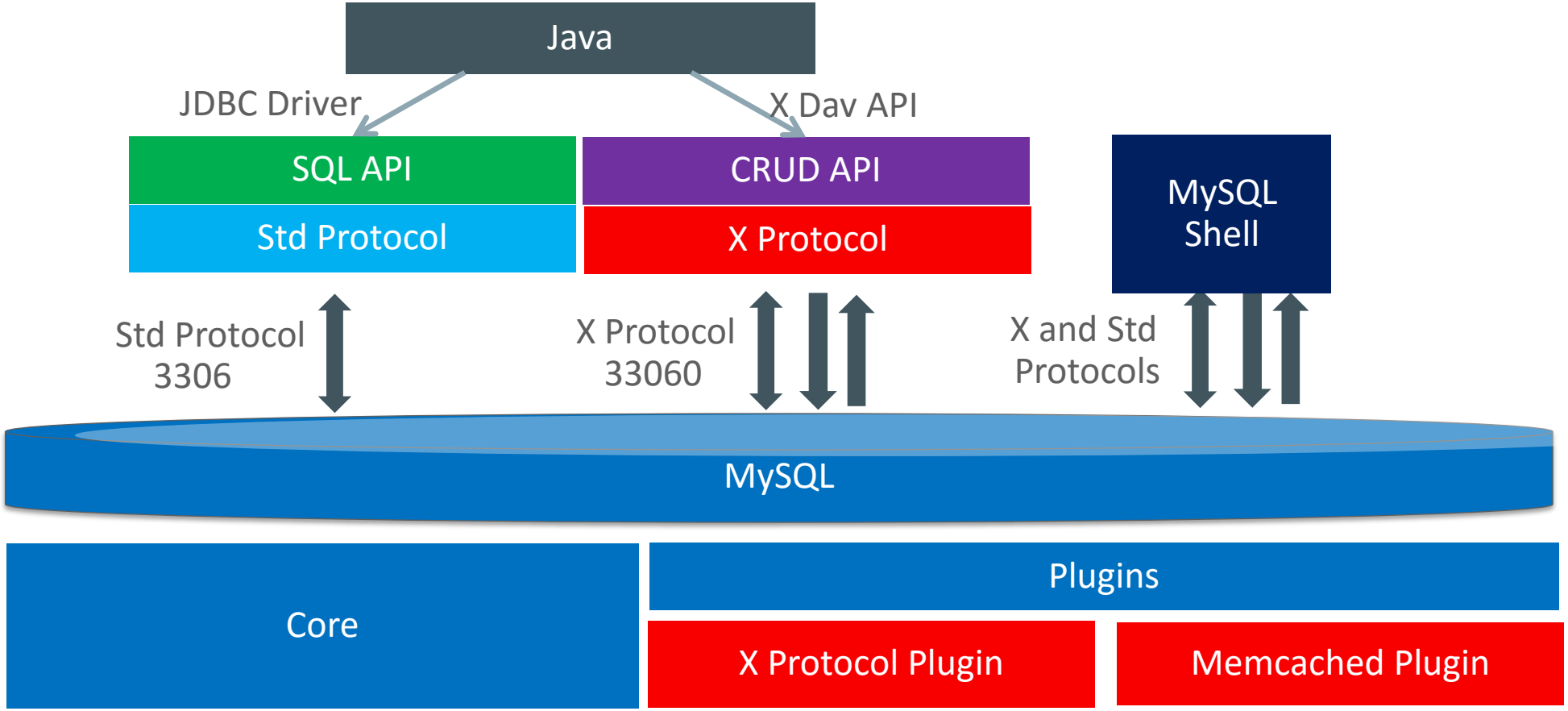
```
SELECT data, id FROM t1 WHERE
   JSON_EXTRACT(data,"$.series") BETWEEN 3 AND 5;
   id BETWEEN 3 AND 5;
```

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

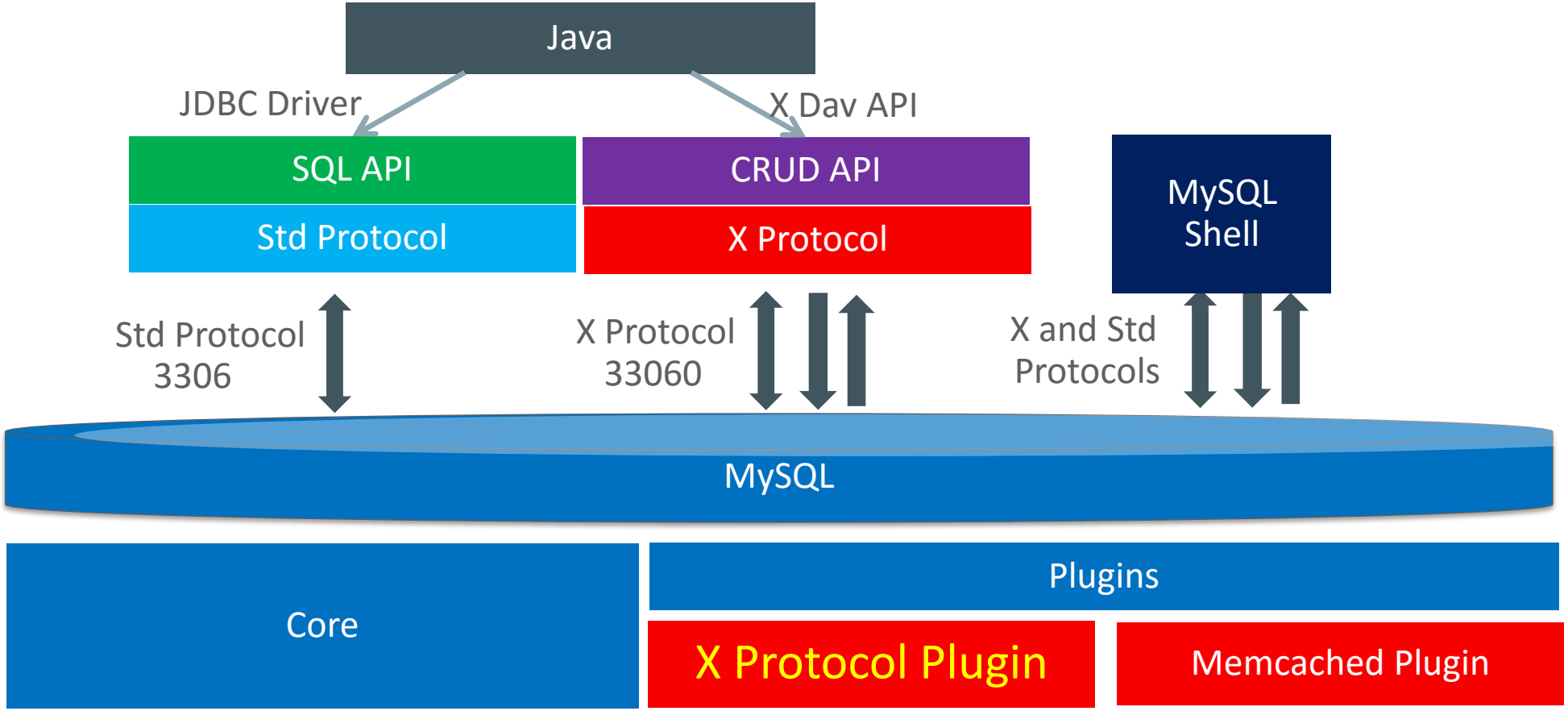
Indexing JSON data: an example

```
> 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)
```

Architecture – mysqlx plugin



Architecture – mysqlx plugin



New! MySQL Server Plugin

Exposes Document APIS to the Connectors

- Designed to enable rapid innovation in APIs and Protocol
- Supports new protocol
  - With new CRUD and other added interfaces
- Includes instruments for monitoring in Performance Schema
- Works concurrently with Standard SQL APIs
- Runs a new port - 33060

Install mysqlx Plugin

- You may find plugin library with MySQL 5.7.12 or later release
  - 在lib/plugin/mysqlx.so(或mysqlx.dll)

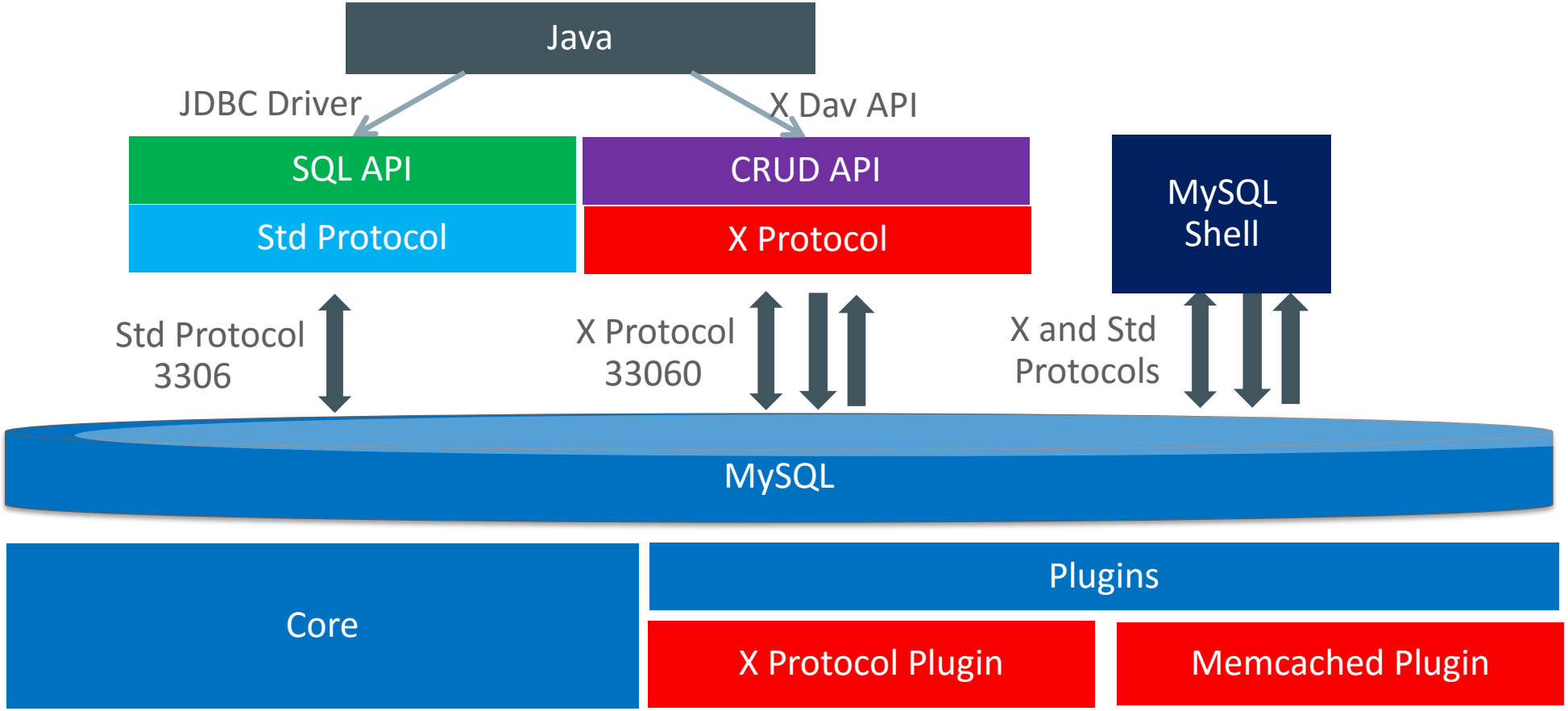
```
INSTALL PLUGIN mysqlx  SONAME 'mysqlx.so';
```

  - Create an account for MySQL X protocol and grant privileges(optional)

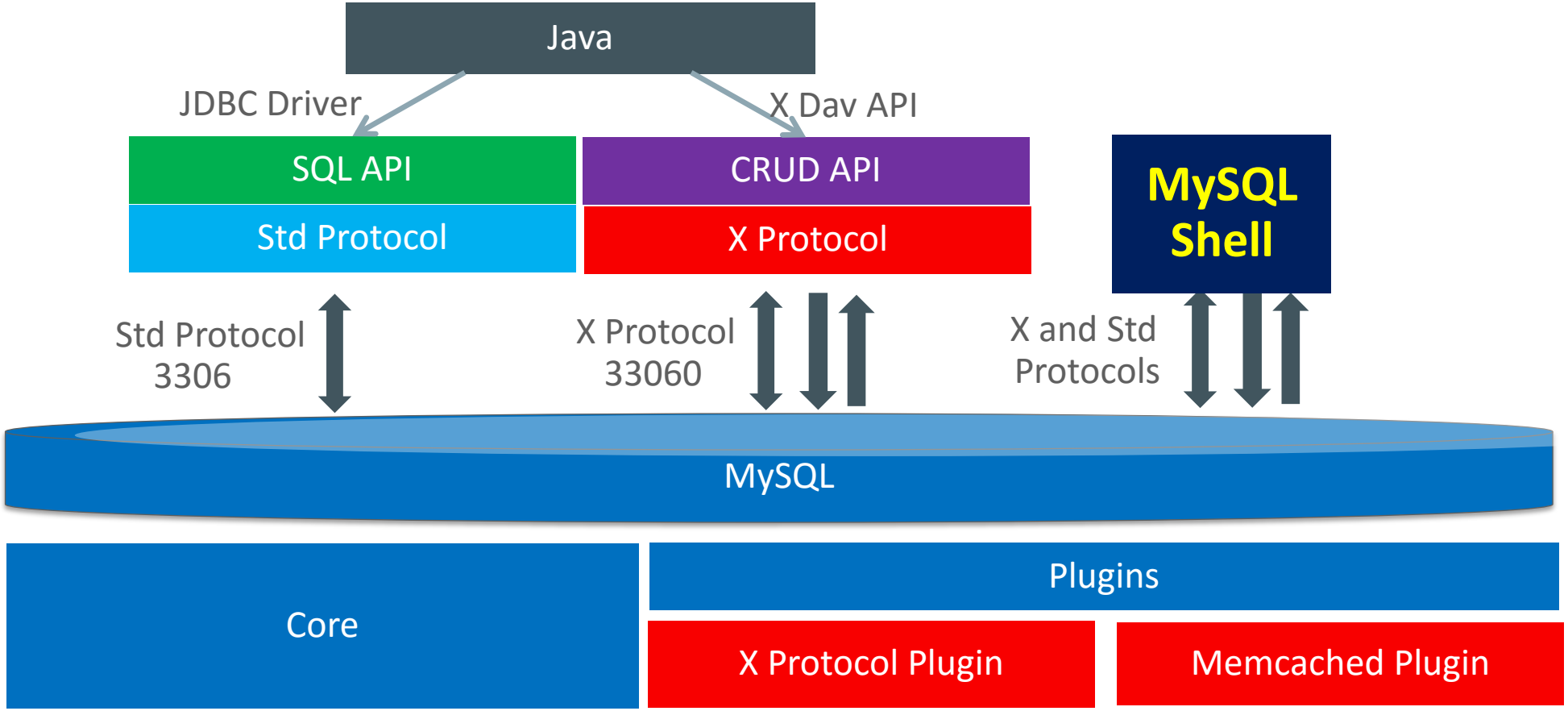
```
> CREATE USER IF NOT EXISTS mysqlxsys@localhost IDENTIFIED WITH mysql_native_password AS
  'password' ACCOUNT LOCK;
> GRANT SELECT ON mysql.user TO mysqlxsys@localhost; GRANT SUPER ON *.* TO
  mysqlxsys@localhost;
```

- Install MySQL Shell
  - Download from http://dev.mysql.com/downloads/shell/
  - From MySQL Repository or install MySQL Shell maually
  - Execute MySQL Shell – ex: mysqlsh --uri mysqlxsys@localhost:33060/world_x

A rchitecture– MySQL Shell



Architecture – MySQL Shell



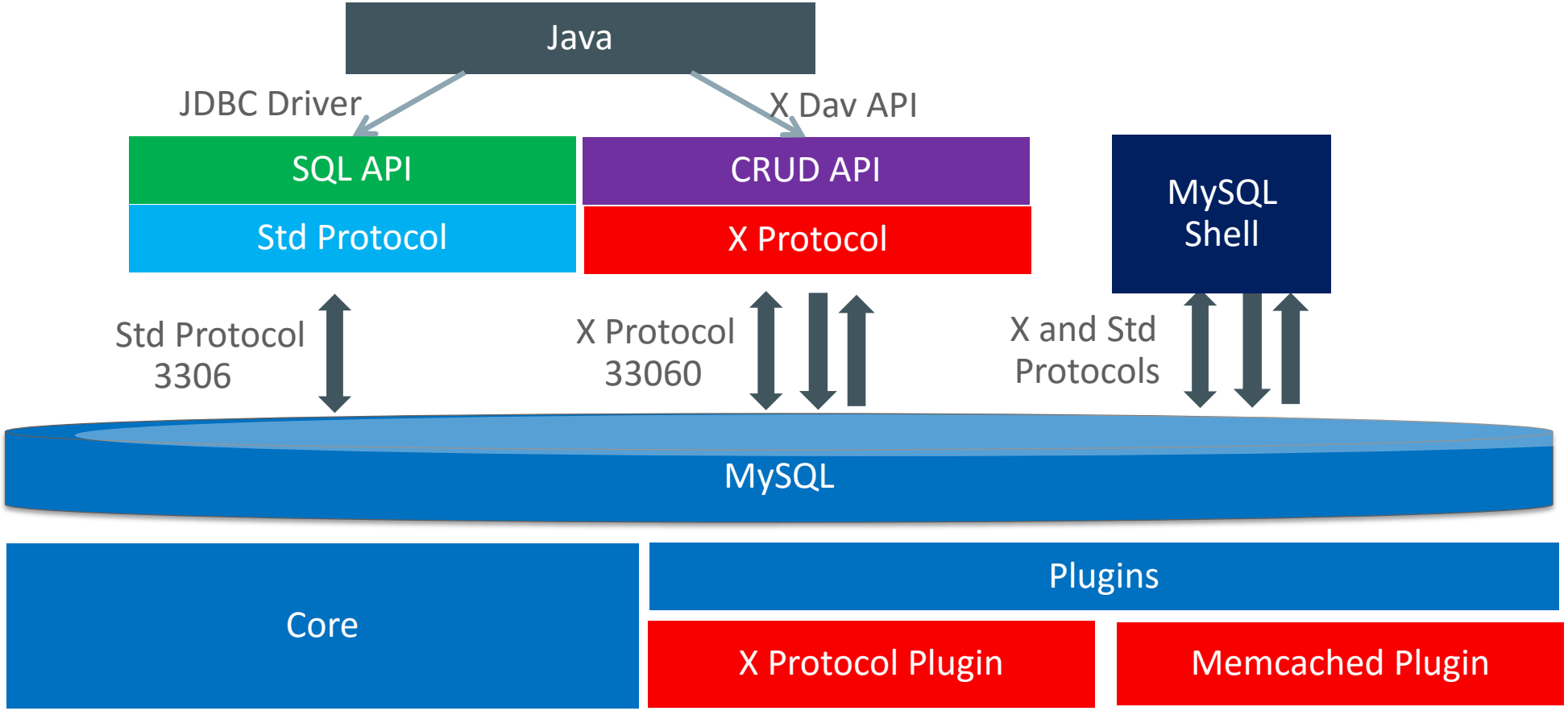
New! MySQL Shell – Key Features

Integrated Development and Administration Shell

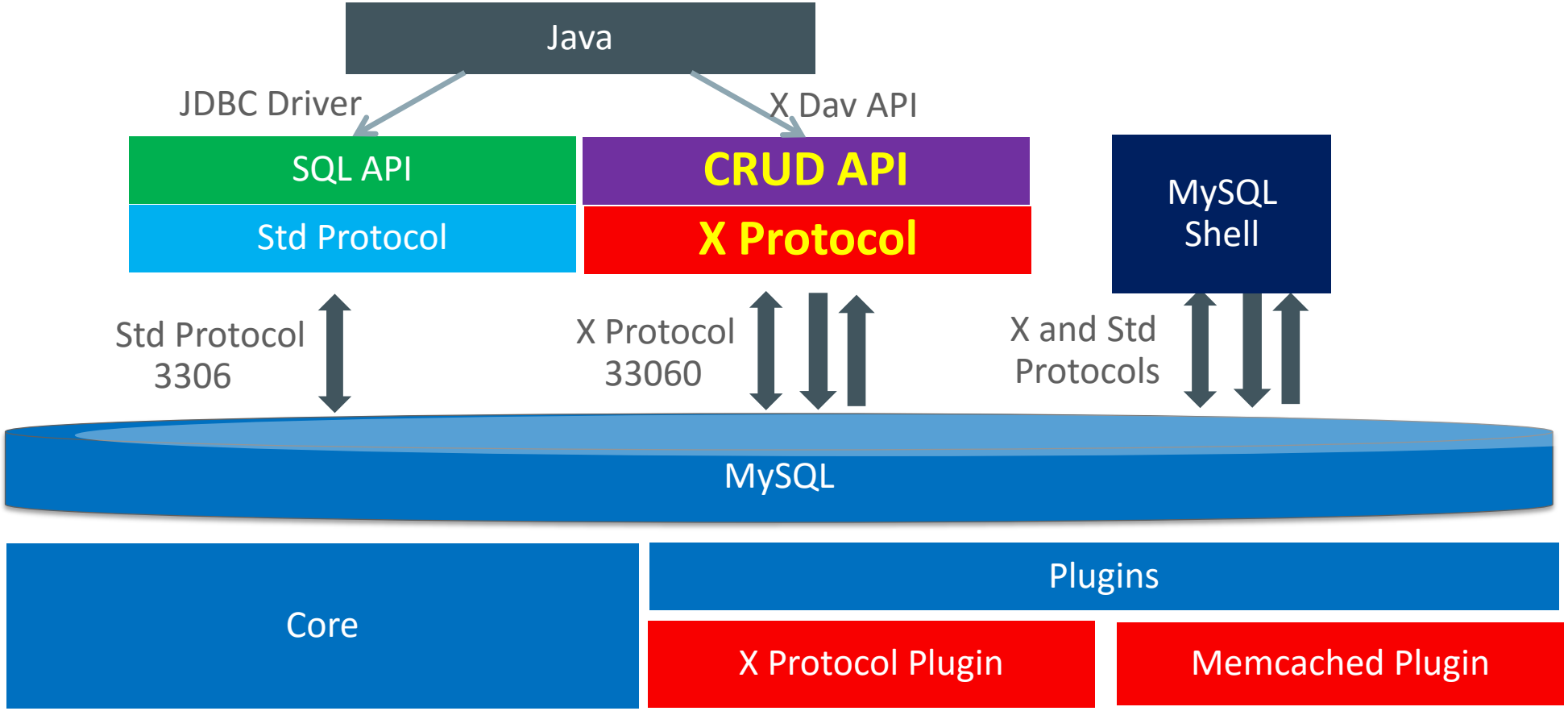
- Multi-Language - JavaScript, Python, and SQL languages
- Supports both Document and Relational Models
- Interactive and Batch processing modes
- Three results formats – Traditional Table, JSON, Tab Separated,
- Exposes Full Development API
  - Sessions: 3 session type
  - Schemas
  - Collections
  - Tables
  - CRUD on Tables and Collections
  - SQL execution (--classic session, \sql mode)
  - Result Processing
  - Parameter Binding

```
C:\Program Files\MySQL\MySQL Shell 1.0\bin>mysqlsh -uroot -p -h127.0.0.1
Creating an X Session to root@127.0.0.1:33060
Enter password:********
No default schema selected.

Welcome to MySQL Shell 1.0.4 Development Preview

Copyright (c) 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type '\help', '\h' or '\?' for help.

Currently in JavaScript mode. Use \sql to switch to SQL mode and execute queries.
mysql-js> \u my_test
Schema `my_test` accessible through db.
mysql-js> db.getCollections()
[
    <Collection:mycollection>,
    <Collection:mycollection2>,
    <Collection:mycollection3>,
    <Collection:ttt>
]
mysql-js> db.mycollection.find()
[
    {
        "_id": "13a90b0a4af7451c9affb89dae36d5a8",
        "age": 15,
        "name": "Sakila"
    },
    {
```

Architecture



Architecture – X Dav API and Connector



New! MySQL Connectors

- Flexible
  - Enabling rapid but stable evolution
  - Use SQL, CRUD APIs – Document and Relational, or “All of the Above”
    - All of this is in addition to the Classic APIs
- Supporting
  - Concurrency, Asynchronous, Pipeline, Batch, ...
- Supporting new modern language capabilities
  - Method Chaining, Promises, Asynchronous constructs, JSON results, ...

Sessions and Connections

- A new high-level concept enables you to write code that can transparently scale from single MySQL Server to a multiple server environment
- A session can have one or more connections
- Initial version supports executing on a single node at at time
- Connection Types
  - Node
  - App
  - SQL – traditional mode

MySQL Documents and Collections

- Collections are containers for documents
  - These documents share a purpose
  - Possibly share one or more indexes
  - Each collection has a unique name
  - Exists within a single schema
- Within a Collection you can
  - Add(), Find(), Modify(), and Remove() - JSON documents
- Collections can be
  - Create(), List(), Drop()

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

- Supports many operators to specify searches
  - ||, &&, XOR, IS, NOT, BETWEEN, IN, LIKE, !=, <>, >, >=, <, <=, &, |, <<, >>, +, -, *, /, ~, %.
- Searching
  - db.CountryInfo.find("GNP > 500000 and demographics.Population < 100000000")
  - db.CountryInfo.find("GNP*1000000/demographics.Population > 30000")
- Binding - bind()
  - db.CountryInfo.find("Name = :country").bind("country", "Italy")
- Project Results – fields() – returns specific fields
  - db.CountryInfo.find("Name = :country").bind("country", "Italy")

CRUD Operations – NoSQL/Document and SQL/Relational

Operation	Document	Relational
Create	Collection.add()	Table.insert()
Read	Collection.find()	Table.select()
Update	Collection.modify()	Table.update()
Delete	Collection.remove()	Table.delete()

Access Document based Database from Java - preparing

- Download and assign classpath to Connector/J and protocol buffer library
  - Connector/J 6.0.x from http://dev.mysql.com/downloads/connector/j/
  - Google Protocol buffer library (protobuf-java-2.6.1.jar) from https://repo1.maven.org/maven2/com/google/protobuf/protobuf-java/2.6.1/

Access Document based Database from Java - Sample

```
package mysqldocstore;
import com.mysql.cj.api.x.*;
import com.mysql.cj.x.MysqlxSessionFactory;
import com.mysql.cj.x.json.*; //以上package都是MySQL Connetor/J 6.0.x 所新增的
public class MySQLdocStore {
    public static void main(String[] args) {
        String url = "mysqlx://localhost:33060/my_test?user=itu&password=welcome";
        XSession mySession = new MysqlxSessionFactory().getSession(url);
        mySession.startTransaction();
        Schema myDb = mySession.getSchema("my_test");
        Collection myColl = myDb.getCollection("mycollection");
        myColl.add("{\"name\":\"Sakila\", \"age\":15}").execute();
        myColl.add("{\"name\":\"Susanne\", \"age\":24}").execute();
        myColl.add("{\"name\":\"Mike\", \"age\":39}").execute();
        mySession.commit();
        DocResult docs = myColl.find("name like :name AND age < :age").bind("name",
  "Sakila").bind("age", 20).execute();
        DbDoc doc = docs.fetchOne();
        System.out.println("Age below 20 is "+doc);
    }
}
```

Access Document based Database from Java – Sample P1

```
package mysqldocstore;
import com.mysql.cj.api.x.*;
import com.mysql.cj.x.MysqlxSessionFactory;
import com.mysql.cj.x.json.*;
//以上package都是MySQL Connetor/J 6.0.x 所新增的
```

Access Document based Database from Java – Sample P2

```
String url = "mysqlx://localhost:33060/my_test?user=itu&password=welcome";
XSession mySession = new MysqlxSessionFactory().getSession(url);
mySession.startTransaction();
Schema myDb = mySession.getSchema("my_test");
Collection myColl = myDb.getCollection("mycollection");
myColl.add("{\"name\":\"Sakila\", \"age\":15}").execute();
myColl.add("{\"name\":\"Susanne\", \"age\":24}").execute();
myColl.add("{\"name\":\"Mike\", \"age\":39}").execute();
mySession.commit();
DocResult docs =
  myColl.find("name like :name AND age < :age").bind("name",
  "Sakila").bind("age", 20).execute();
DbDoc doc = docs.fetchOne();
System.out.println("Age below 20 is "+doc);
```

Execute The Program

```
$ java –jar MySQLdocStore.jar
Age below 20 is {
"_id" : "bb8c3bf0242a44a0a49767e3dcee7cc9",
"age" : 15,
"name" : "Sakila"
}
```

Execution Results

- On MySQL Shell

```
mysql-js> db.mycollection.find();
[
    {
        "_id": "2f5a69049e0744f4b1f3abd9bbbbd2dd",
        "age": 39,
        "name": "Mike"
    },
    {
        "_id": "bb8c3bf0242a44a0a49767e3dcee7cc9",
        "age": 15,
        "name": "Sakila"
    },
    {
        "_id": "d9d3e1c86a3643be93e65ac1981c659f",
        "age": 24,
        "name": "Susanne"
    }
]
3 documents in set (0.00 sec)
```

Execution Results (Cont.)

- From MySQL SQL interface

```
mysql> select * from mycollection\G
*************************** 1. row ***************************
doc: {"_id": "2f5a69049e0744f4b1f3abd9bbbbd2dd", "age": 39, "name": "Mike"}
_id: 2f5a69049e0744f4b1f3abd9bbbbd2dd
*************************** 2. row ***************************
doc: {"_id": "bb8c3bf0242a44a0a49767e3dcee7cc9", "age": 15, "name": "Sakila"}
_id: bb8c3bf0242a44a0a49767e3dcee7cc9
*************************** 3. row ***************************
doc: {"_id": "d9d3e1c86a3643be93e65ac1981c659f", "age": 24, "name": "Susanne"}
_id: d9d3e1c86a3643be93e65ac1981c659f
3 rows in set (0.00 sec)
```

Features From MySQL

- Support Transaction
- MVCC and ACID
- HA solutions
- Management and Monitor
  - MySQL Enterprise Monitor
  - MySQL Enterprise Backup
  - MySQL Enterprise Audit
  - MySQL Firewall
- Data Shard

We Need You

- Try it and feedback us(bugs.mysql.com)
- Support by your framework
  - Java
  - Javascript + Node.js
  - 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

More Resources

- Chinese version 5.7 New Features White Paper https://go.oracle.com/LP=42563?elqCampaignID=73758
- 1月13日Group Replication Webinar

http://event.on24.com/r.htm?eventid=1322985&sessionid=1&key=3937BEF05794C23D47B53E03E36DFD4D&partnerref=On24:EX:PEV::Acmug


ORACLE®