MySQL 5.7 - NoSQL Възможности

NoSQL, JSON, PluginX

Петр Зайцев

CEO, Percona

7 July 2017

PERCONA

О презентации

Кратко о NoSQL

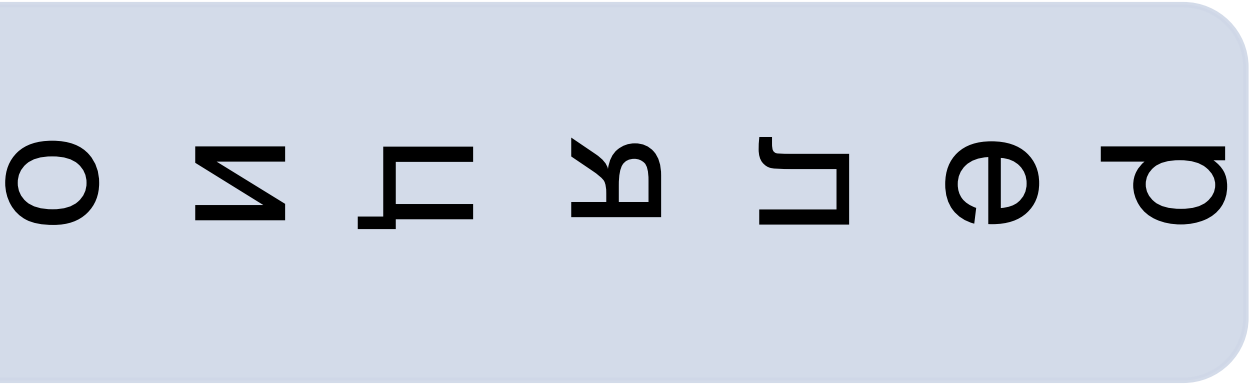
История NoSQL в MySQL

Новые возможности MySQL 5.7

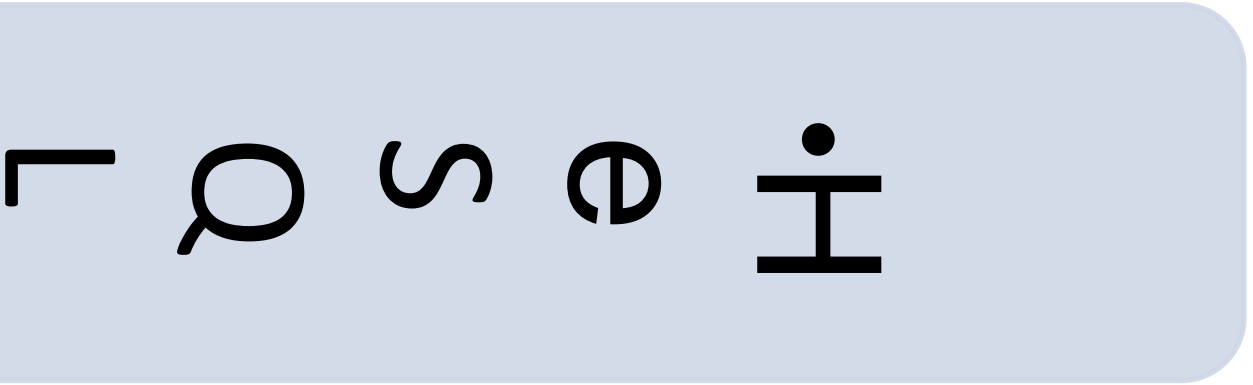
Кратко о NoSQL

Определение NoSQL обычно состоит из

Модель Данных



Язык Доступа



PERCONA

NoSQL Модели Данных

Key-Value

Document Stores

Wide-Column Stores

Graph

Multi-Model

Document Store – наиболее популярное из NoSQL

328 systems in ranking, July 2017

Rank			DBMS	Database Model	Score		
Jul 2017	Jun 2017	Jul 2016			Jul 2017	Jun 2017	Jul 2016
1.	1.	1.	Oracle	Relational DBMS	1374.88	+23.11	-66.65
2.	2.	2.	MySQL	Relational DBMS	1349.11	+3.80	-14.18
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1226.00	+27.03	+33.11
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	369.44	+0.89	+58.28
5.	5.	↓ 4.	MongoDB	Document store	332.77	-2.23	+17.77
6.	6.	6.	DB2	Relational DBMS	191.25	+3.74	+6.17
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	126.13	-0.42	+1.23
8.	8.	↓ 7.	Cassandra	Wide column store	124.12	-0.00	-6.58
9.	9.	↑ 10.	Redis	Key-value store	121.51	+2.63	+13.48
10.	↑ 11.	↑ 11.	Elasticsearch	Search engine	115.98	+4.42	+27.36

PERCONA

Другая расшифровка NoSQL

PERCONA

Преимущества NoSQL

- Скорость Разработки
- Простота
- Производительность
- Масштабируемость
- Более удобная модель для ряда приложений

NoSQL в MySQL

Фокус в MySQL

Гибкая схема

CRUD доступ

Пример JSON Документа

```
{
  ISBN: 9780992461225,
  title: "JavaScript: Novice to Ninja",
  author: "Darren Jones",
  format: "ebook",
  price: 29.00,
  publisher: {
    name: "SitePoint",
    country: "Australia",
    email: "feedback@sitepoint.com"
  }
}
```

CRUD

CREATE – Создавать документ

READ – Читать документ

UPDATE – Изменять документ

DELETE – Удалять документ

From MySQL to MongoDB

MySQL

```
mysql> select * from zips limit 1\G
*************************** 1. row
***************************
country_code: US
 postal_code: 34050
  place_name: FPO
 admin_name1:
 admin_code1: AA
 admin_name2: Erie
 admin_code2: 029
 admin_name3:
 admin_code3:
    latitude: 41.03750000
   longitude: -111.67890000
    accuracy:
1 row in set (0.00 sec)
```

MongoDB

```
MongoDB shell version: 3.0.8
connecting to: zips
> db.zips.find().limit(1).pretty()
{
        "_id" : "01001",
        "city" : "AGAWAM",
        "loc" : [
                -72.622739,
                42.070206
        ],
        "pop" : 15338,
        "state" : "MA"
}
```

Where is my *SQL*?

SQL to MongoDB Mapping Chart

https://docs.mongodb.org/manual/reference/sql-comparison/

MySQL

```
CREATE TABLE users (
    id MEDIUMINT NOT NULL
        AUTO_INCREMENT,
    user_id Varchar(30),
    age Number,
    status char(1),
    PRIMARY KEY (id)
)
```

MongoDB

```
db.users.insert( {
    user_id: "abc123",
    age: 55,
    status: "A"
 } )
```

(no schema)

Where is my *SQL*?

SQL to MongoDB Mapping Chart

https://docs.mongodb.org/manual/reference/sql-comparison/

MySQL

```
SELECT *
FROM users
WHERE status = "A"
AND age = 50
```

MongoDB

```
db.users.find(
    { status: "A",
      age: 50 }
)
```

NoSQL протокол доступа в MySQL

HANDLER Команда	Прямой доступ к данным
MySQL Cluster (NDB) 2004	Собственный протокол
MyCacheD 2009	Memcache протокол
HandlerSocket 2010	Собственный протокол
Официальная поддержка MemcacheD протокола 2011	Демон memcached
Protocol X 2016	Протокол на основе Google Protocol Buffers

PERCONA

История поддержки NoSQL Модели Данных

- Храним Сериализованные объекты в BLOB/TEXT
- Поддержка Xpath в MySQL (2005)
- Динамические колонки в MariaDB (2012)
- JSON UDF функции Светы Смирновой (2013)
- Поддержка типа данных JSON MySQL 5.7 (2015)

Тип JSON в MySQL 5.7

Архитектурные основы поддержки JSON

- Нативный JSON тип данных
- Поддержка индексов не основанных на колонке
- Поддержка доступа к полям в синтаксисе SQL

Нативный JSON тип

Бинарный формат хранения

Парсинг и валидация только при вставке

Индекс для быстрого доступа к полям и элементам массивов

Поддержка резервации места и обновления на месте (В будущем)

Поддержка Типов

Все стандартные JSON типы

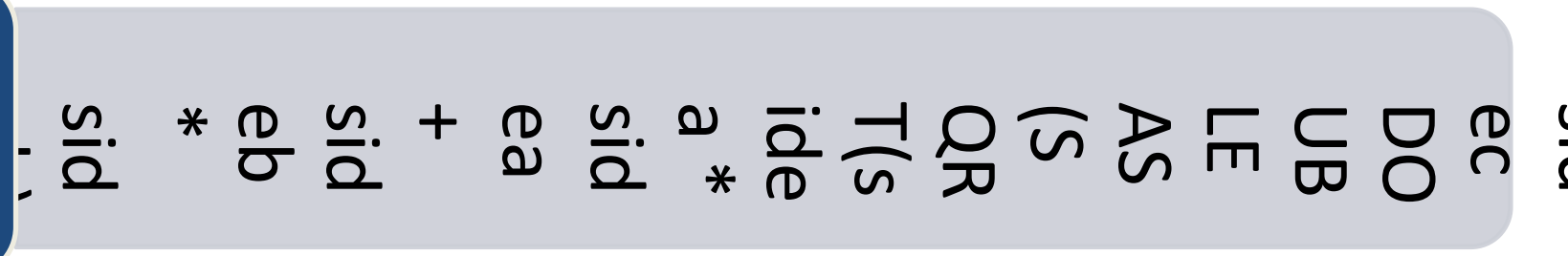
- Номера
- Строки
- Булевский Тип
- Вложенные документы
- Массивы

Расширенная поддержка типов

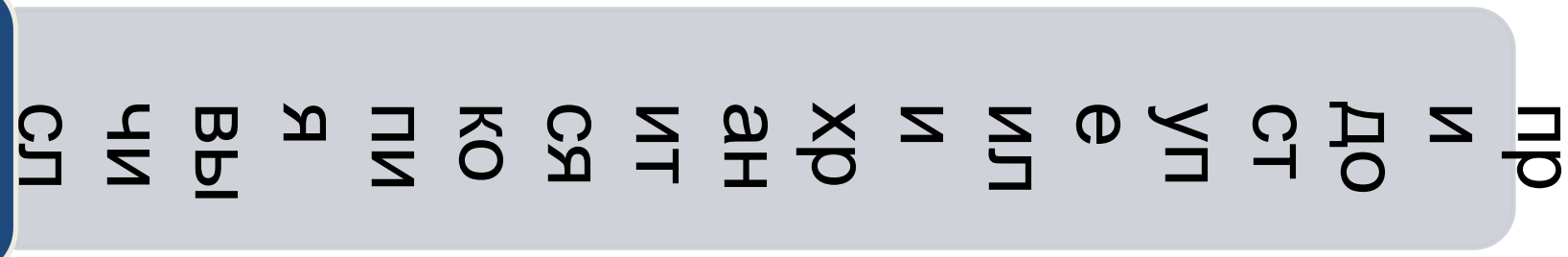
- Дата
- Время
- DateTime

Поддержка вычислимых колонок

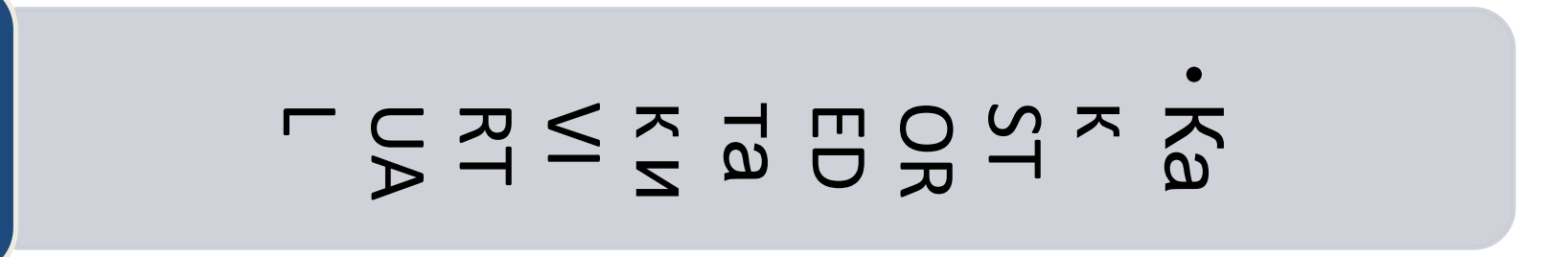
Значение колонки вычисляется на базе других колонок



Можем быть VIRTUAL или STORED



Можно индексировать



PERCONA

Вычислимые колонки

```
CREATE TABLE `ontime` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `YearD` year(4) NOT NULL,
  `FlightDate` datetime DEFAULT NULL,
  `Carrier` char(2) DEFAULT NULL,
  `OriginAirportID` int(11) DEFAULT NULL,
  `OriginCityName` varchar(100) DEFAULT NULL,
  `OriginState` char(2) DEFAULT NULL,
  `DestAirportID` int(11) DEFAULT NULL,
  `DestCityName` varchar(100) DEFAULT NULL,
  `DestState` char(2) DEFAULT NULL,
...
`Flight_dayofweek` tinyint(4)
GENERATED ALWAYS AS (dayofweek(FlightDate)) VIRTUAL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB;
alter table ontime add key (Flight_dayofweek);
```

```
SELECT Flight_dayofweek, count(*)
FROM ontime_sm_virtual
GROUP BY Flight_dayofweek
```

Не храним колонку но индексируем

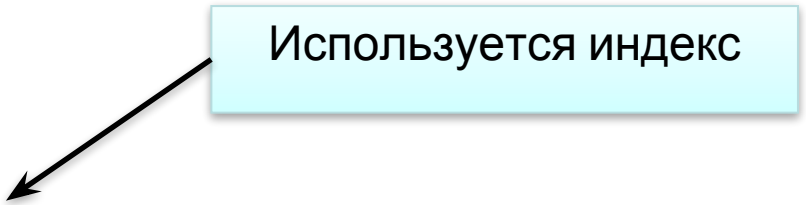
https://www.percona.com/blog/2015/04/29/generated-virtual-columns-in-mysql-5-7-labs/

https://dev.mysql.com/worklog/task/?id=8114

PERCONA

Вычислимые колонки

```
mysql> EXPLAIN SELECT carrier, count(*)
       FROM ontime_sm_virtual
       WHERE Flight_dayofweek = 7 group by carrier\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: ontime_sm_virtual
   partitions: NULL
         type: ref
possible_keys: Flight_dayofweek
          key: Flight_dayofweek
      key_len: 2
          ref: const
         rows: 165409
     filtered: 100.00
        Extra: Using where; Using temporary; Using filesort
1 row in set, 1 warning (0.00 sec)
```



PERCONA

Нужен удобный SQL синтаксис

- Использование JSON функций для доступа к полу очень неудобно

```
mysql> SELECT c, c->"$.id", g
     > FROM jemp
     > WHERE c->"$.id" > 1
     > ORDER BY c->"$.name";
+---------------------------------+-----------+------+
| c                               | c->"$.id" | g    |
+---------------------------------+-----------+------+
| {"id": "3", "name": "Barney"} | "3"       |    3 |
| {"id": "4", "name": "Betty"}  | "4"       |    4 |
| {"id": "2", "name": "Wilma"}  | "2"       |    2 |
+---------------------------------+-----------+------+
3 rows in set (0.00 sec)
```

Пример использования типа JSON

```
mysql> create table json_test (
id int primary key auto_increment,
data json
) engine=InnoDB;
Query OK, 0 rows affected (0.02 sec)
```

То же самое как JSON_EXTRACT(data,"$.type")

```
mysql> select * from json_test where data->'$.type' = 'Point' limit 1;
+----+-------------------------------------------------------+
| id | data                                                  |
+----+-------------------------------------------------------+
|  1 | {"type": "Point", "coordinates": [-87.9101245, 41.7585879]} |
+----+-------------------------------------------------------+
```


PERCONA

Поддержка Индексов для JSON

```
mysql> explain select * from json_test where data->'$.type' = 'Point' limit 1\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: json_test
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 996823
     filtered: 100.00
        Extra: Using where
```

```
mysql> alter table json_test
add data_type varchar(255) GENERATED ALWAYS AS (data->'$.type') VIRTUAL;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> alter table json_test add key (data_type);
Query OK, 0 rows affected (2.51 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Поддержка индексов для JSON

```
mysql> explain select * from json_test where data->'$.type' = 'Point' limit 1\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: json_test
   partitions: NULL
         type: ref
possible_keys: data_type
          key: data_type
      key_len: 258
          ref: const
         rows: 1
     filtered: 100.00
        Extra: NULL
```

Новый протокол доступа (Protocol X)

А хотелось бы не использовать SQL Вообще

Protocol X добавлен в MySQL 5.7

Множество улучшений помимо NoSQL

Поддержка SQL и CRUD одновременно

Новый CLI клиент – MySQL Shell

Инсталлируем MySQL Shell

…

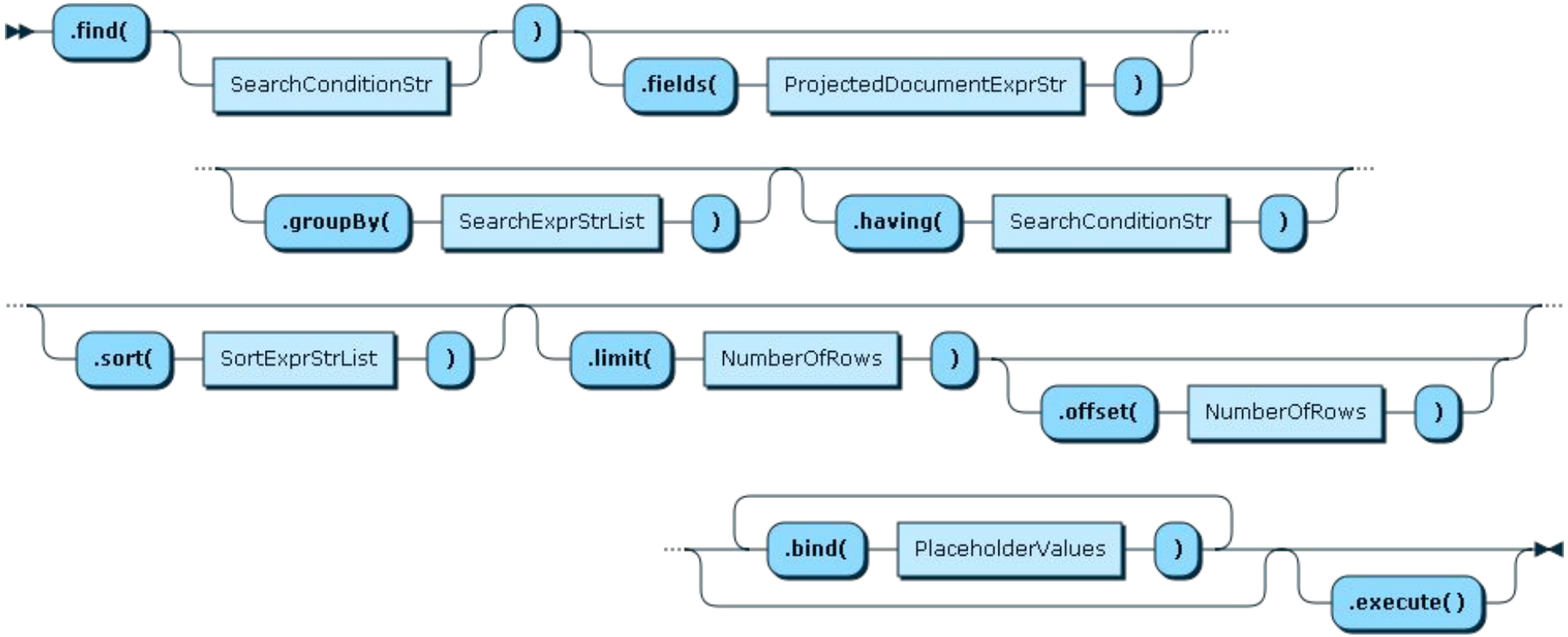
- `apt-get install mysql-apt-config`
- `apt-get install mysql-shell`
- `mysqlsh -u root -h localhost -p --classic --dba enableXProtocol`
- `mysqlsh -u root --sql --recreate-schema world_x < /tmp/world_x-db/world_x.sql`

https://dev.mysql.com/doc/refman/5.7/en/document-store-setting-up.html

Используем MySQL Shell

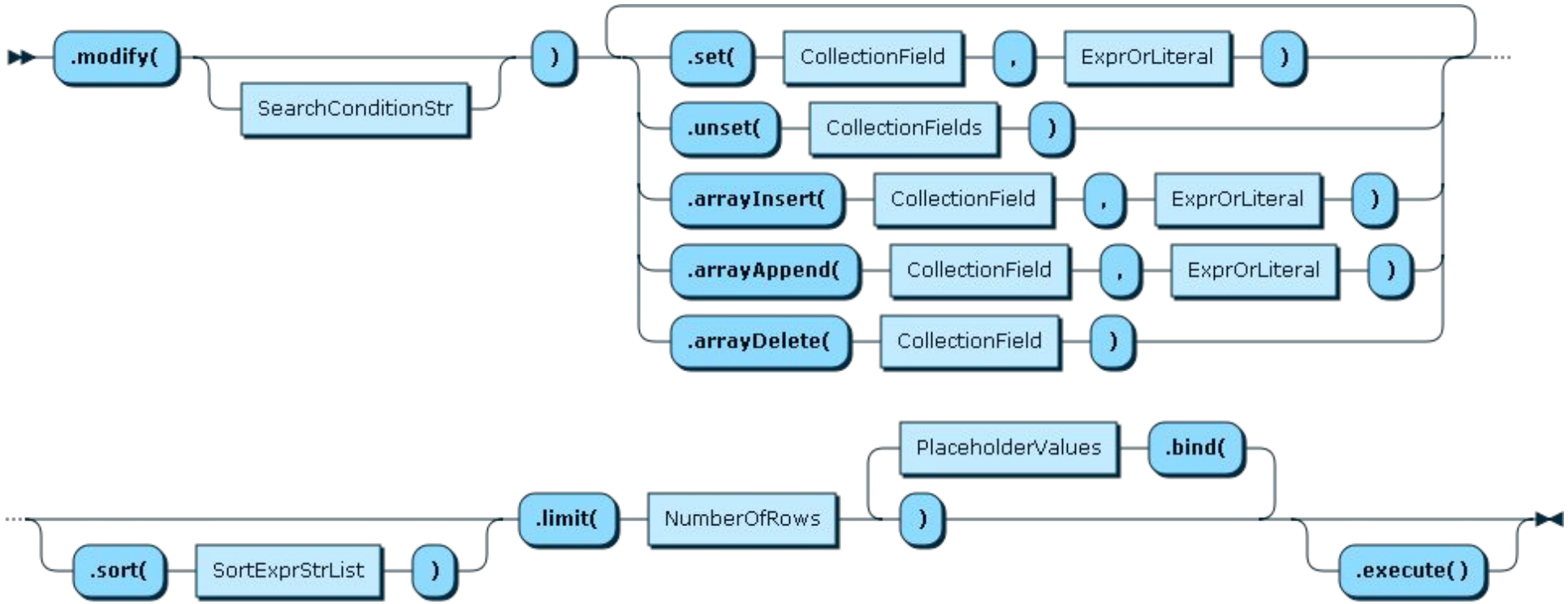
```
mysql-js> db = session.getSchema('world_x')
<Schema:world_x>

mysql-js> db.getCollections()
{
    "CountryInfo": <Collection:CountryInfo>
}
```

.Find()



PERCONA

.Modify()



PERCONA

А как же это все работает внутри ?

- Каждая коллекция представляет собой таблицу
- Бонус: Работает с разными Storage Engines

```
mysql> show create table CountryInfo
*************************** 1. row ***************************
       Table: CountryInfo
Create Table: CREATE TABLE `CountryInfo` (
  `doc` json DEFAULT NULL,
  `_id` varchar(32) GENERATED ALWAYS AS (json_unquote(json_extract(`doc`,'$._id'))) STORED NOT NULL,
  PRIMARY KEY (`_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Выполняем запрос

```
mysql-js> db.getCollection("CountryInfo").find('Name= "United States"').limit(1)
[
    {
        "GNP": 8510700,
        "IndepYear": 1776,
        "Name": "United States",
        "_id": "USA",
        "demographics": {
            "LifeExpectancy": 77.0999984741211,
            "Population": 278357000
        },
        "geography": {
            "Continent": "North America",
            "Region": "North America",
            "SurfaceArea": 9363520
        },
        "government": {
            "GovernmentForm": "Federal Republic",
            "HeadOfState": "George W. Bush",
            "HeadOfState_title": "President"
        }
    }
]
1 document in set (0.02 sec)
```

Если посмотреть в MySQL General Log

- Видим:
  - `2016-05-17T21:02:21.213899Z  186 Query  SELECT doc FROM `world_x`.`CountryInfo` WHERE (JSON_EXTRACT(doc,'$.Name') = 'United States') LIMIT 1`
- MySQL конвертирует CRUD в SQL на низком уровне

Создаем Индекс

```
mysql-js> db.CountryInfo.find("demographics.Population < 100")
...[output removed]
8 documents in set (0.00 sec)
```

```
mysql-js> db.CountryInfo.createIndex("pop").
field("demographics.Population", "INTEGER", false).execute()
Query OK (0.04 sec)
```

```
mysql-js> db.CountryInfo.createIndex("name", mysqlx.IndexType.Unique).
field("Name", "TEXT(40)", true).execute()
Query OK (0.04 sec)
```

Достоинства и недостатки такого дизайна

Плюсы

- Новый интерфейс для разработчиков но администраторам баз данных не нужно учить новый язык
- Все существующие инструменты работают

Минусы

- Сконвертированные запросы тяжело понимать
- Сложно понять из какого места в приложении такие запросы приходят

Так же стоит иметь в виду

- Поддержка возможностей обеспечения консистентности данных
  - Транзакции
  - Режимы изоляции
  - Точки сохранения (Savepoints)
- Можно использовать CRUD и SQL в одной транзакции
- Можно использовать разные Storage Engines
- Инструментация с Performance Schema
- Поддержка MySQL Replication и Percona XtraDB Cluster

Поддержка в языках программирования

Только некоторые языки поддерживаются на данный момент

- C/C++
- Java
- Node.JS
- Python
- .NET

https://dev.mysql.com/doc/index-connectors.html

Подведем итоги

MySQL позволяет использовать CRUD и SQL интерфейс доступа к данным

Можно хранить реляционные и JSON данные в одном транзакционном хранилище

Новое в MySQL 5.7 – ведется активная разработка

Не решает проблем “Scale Out” как ряд других NoSQL решений

Percona Live Europe Call for Papers & Registration are Open!

- Championing Open Source Databases
  - MySQL, MongoDB, Open Source Databases
  - Time Series Databases, PostgreSQL, RocksDB
  - Developers, Business/Case Studies, Operations
  - September 25-27th, 2017
  - Radisson Blu Royal Hotel, Dublin, Ireland

Submit Your Proposal by July 17th!

www.percona.com/live/e17

PERCONA


BECOME A
PERCONA SUPERHERO
WE'RE HIRING
CONTACT
careers@percona.com
PERCONA

Thank You!