ORACLE

NoSQL + SQL = MySQL

MySQLドキュメントストア

Best of the both world

梶山 隆輔 / KAJIYAMA, Ryusuke

MySQL Solution Engineering Director, Asia Pacific & Japan

MySQL Global Business Unit

Safe harbor statement

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

宣伝：日本MySQLユーザ会 YouTubeチャンネル開設

MySQL Casualとあわせてチャンネル登録お願いいたします！！

https://www.youtube.com/channel/UCbNxgqu2iB8iT1X1iZnH3Ig

The world's most popular open source database

世界で最も普及しているオープンソース データベース

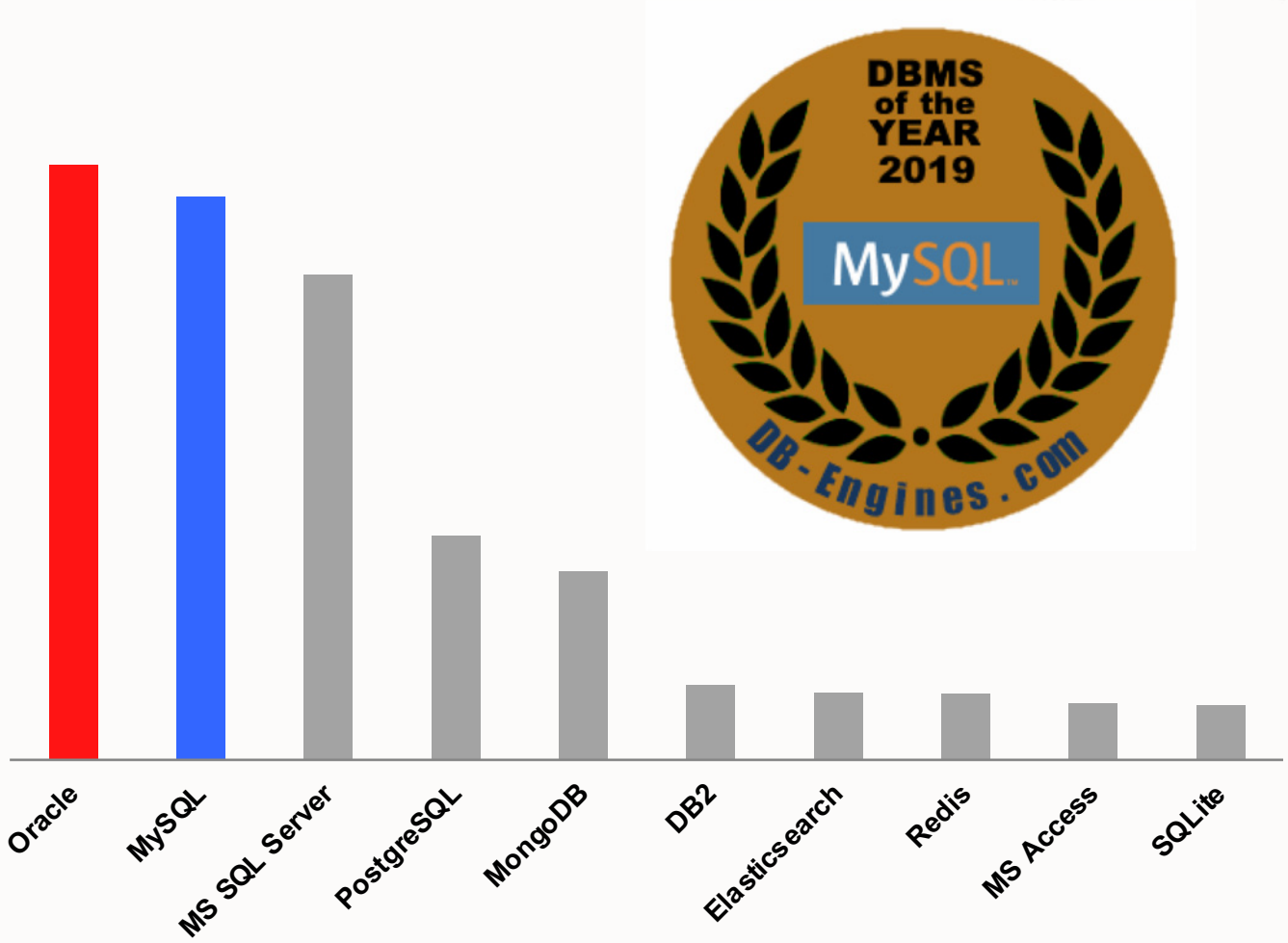
The world's most popular open source database

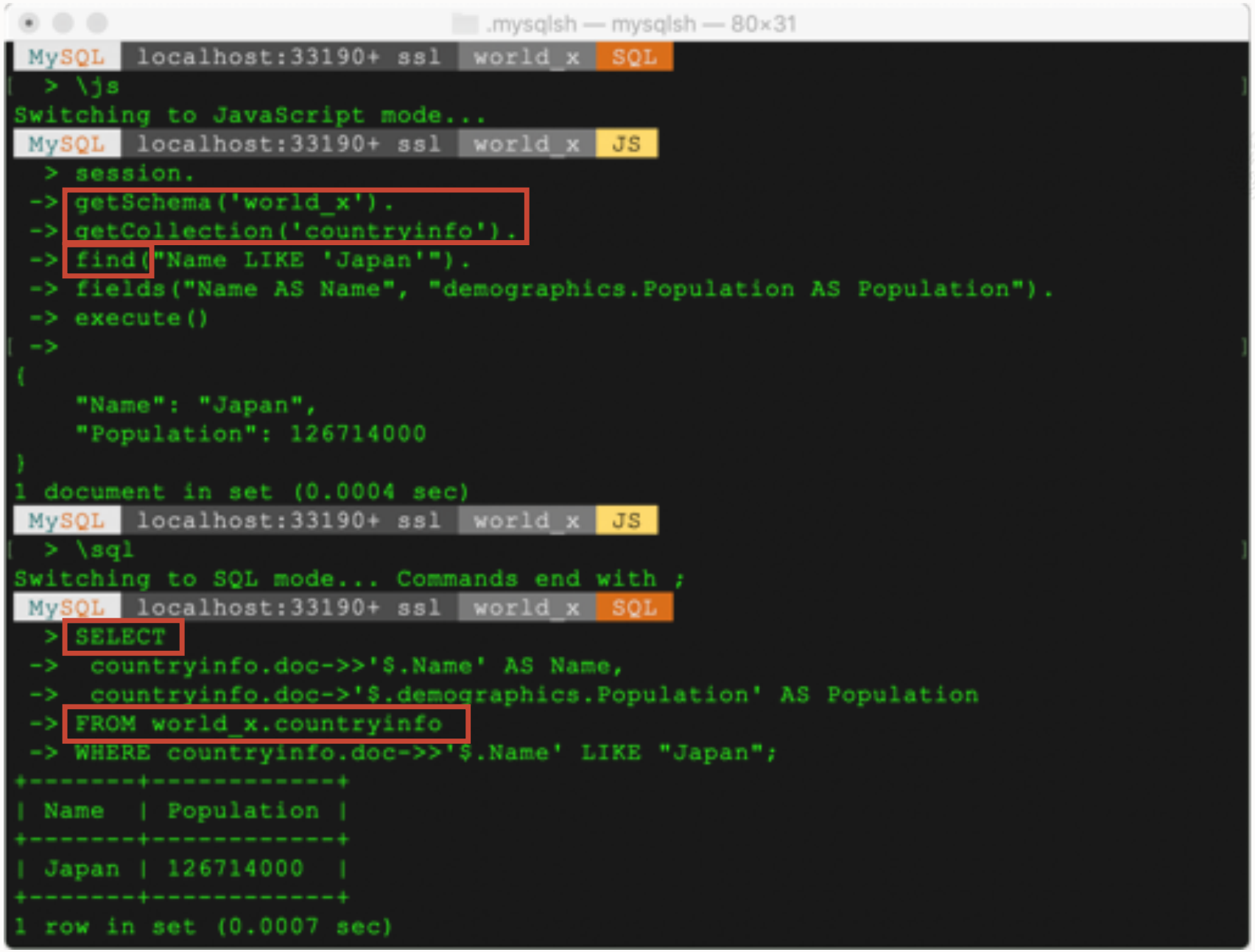
例) DB-Enginesによる調査結果

http://db-engines.com/en/

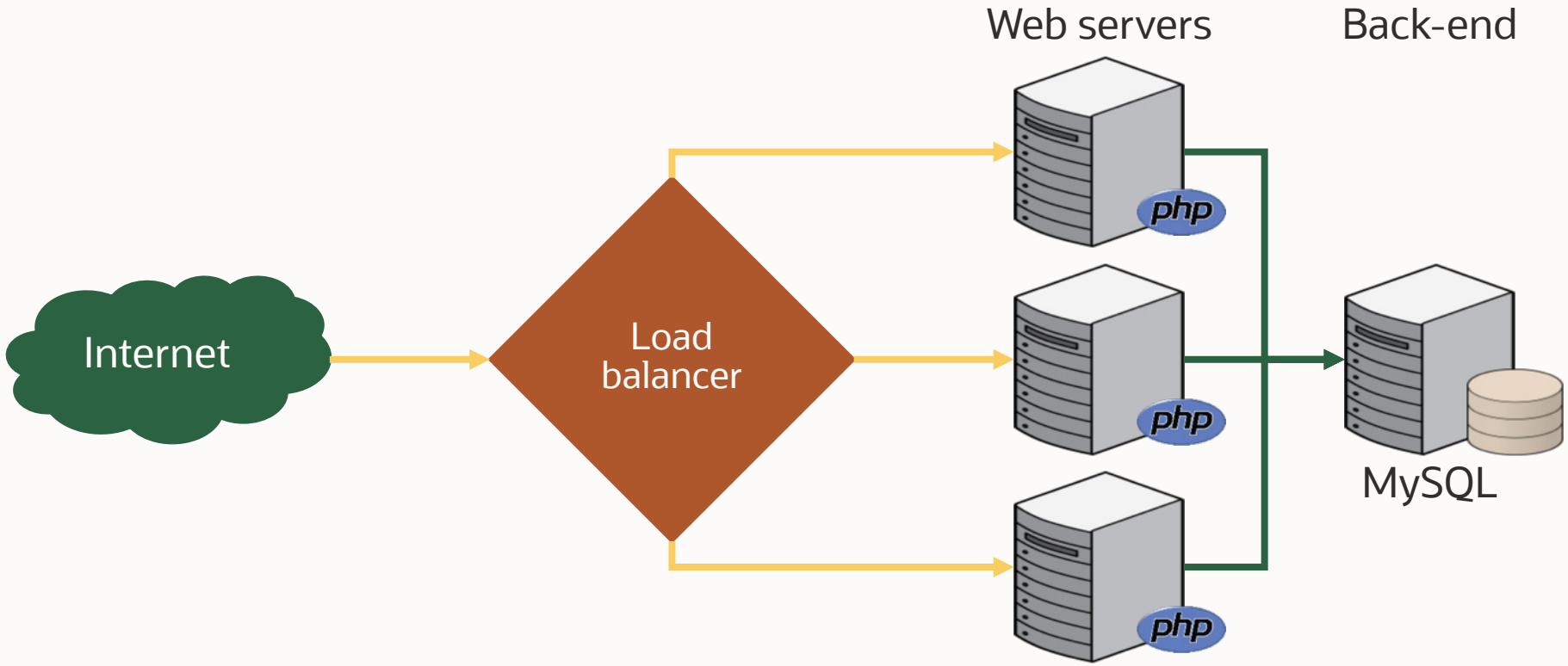
データベースソフトウェアの普及度や人気を、インターネット上の求人情報や職務経歴上での経験、および検索エンジンやSNSでの情報量を元に毎月作成し公開。

≪MySQL≫
- グローバルで多くの利用者
- グローバルで多くの人材
- 多種・多用な管理ツール
- デュアルライセンス
  (コミュニティ・商用)
- Oracle社が企業として提供
  - バージョンリリース、パッチ
  - http://bugs.mysql.com/
  - フルタイムの専任開発者
  - 開発ロードマップ（ユーザベース）
  - 組み込み利用時のIP対応





Webのバックエンドシステム
旧来型の構成



Webのバックエンドシステム
よくあるシステム構成の例

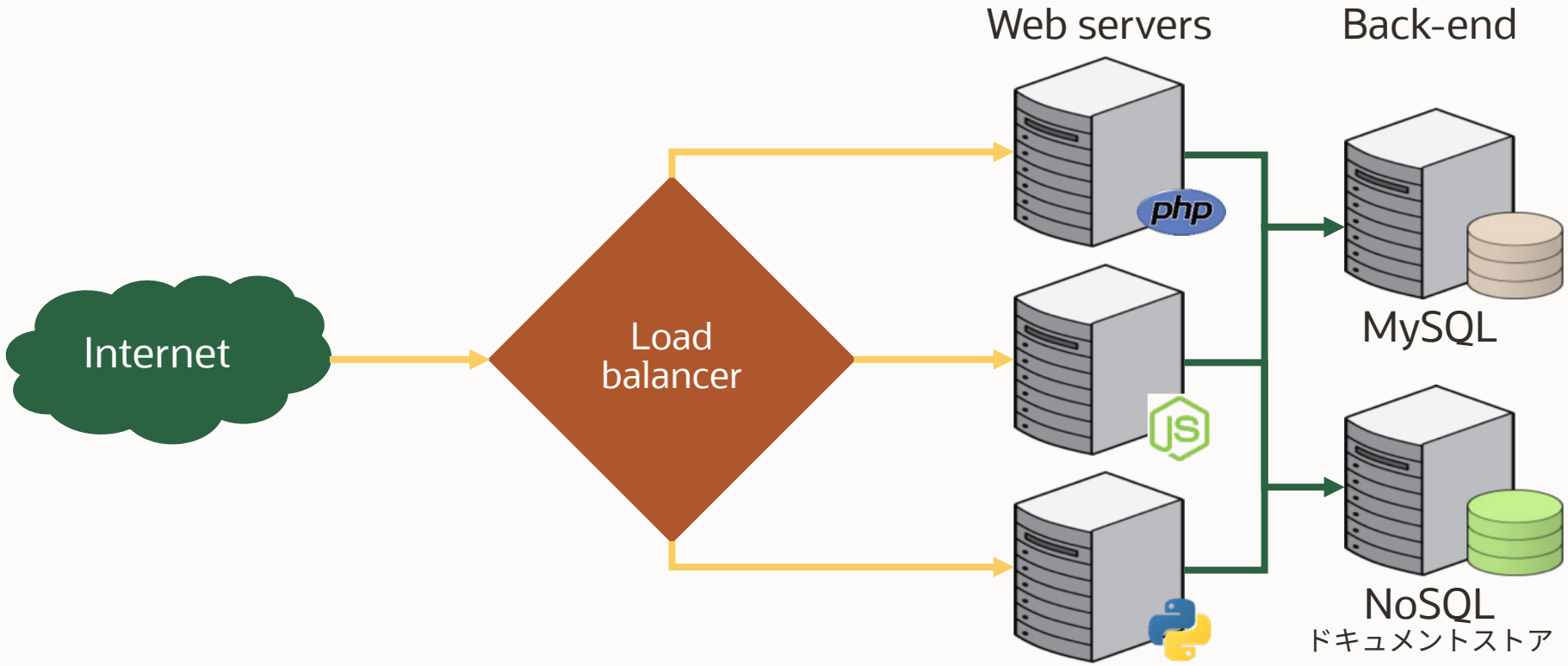
フロントエンドサーバー

- Apache
- HTML, CSS
- PHP/Ruby, Python, Java, AngularJS, Node.js

バックエンドサーバー

- リレーショナルデータベース: MySQL
- 非リレーショナルデータストア

Webのバックエンドシステム
最近よくみるタイプの構成



リレーショナルモデル vs ドキュメントストアモデル

リレーショナルモデル

customer

id	name	email	city_id
3412	John Smith	foobar@example.com	45

city

id	city	country_id
45	San Francisco	US

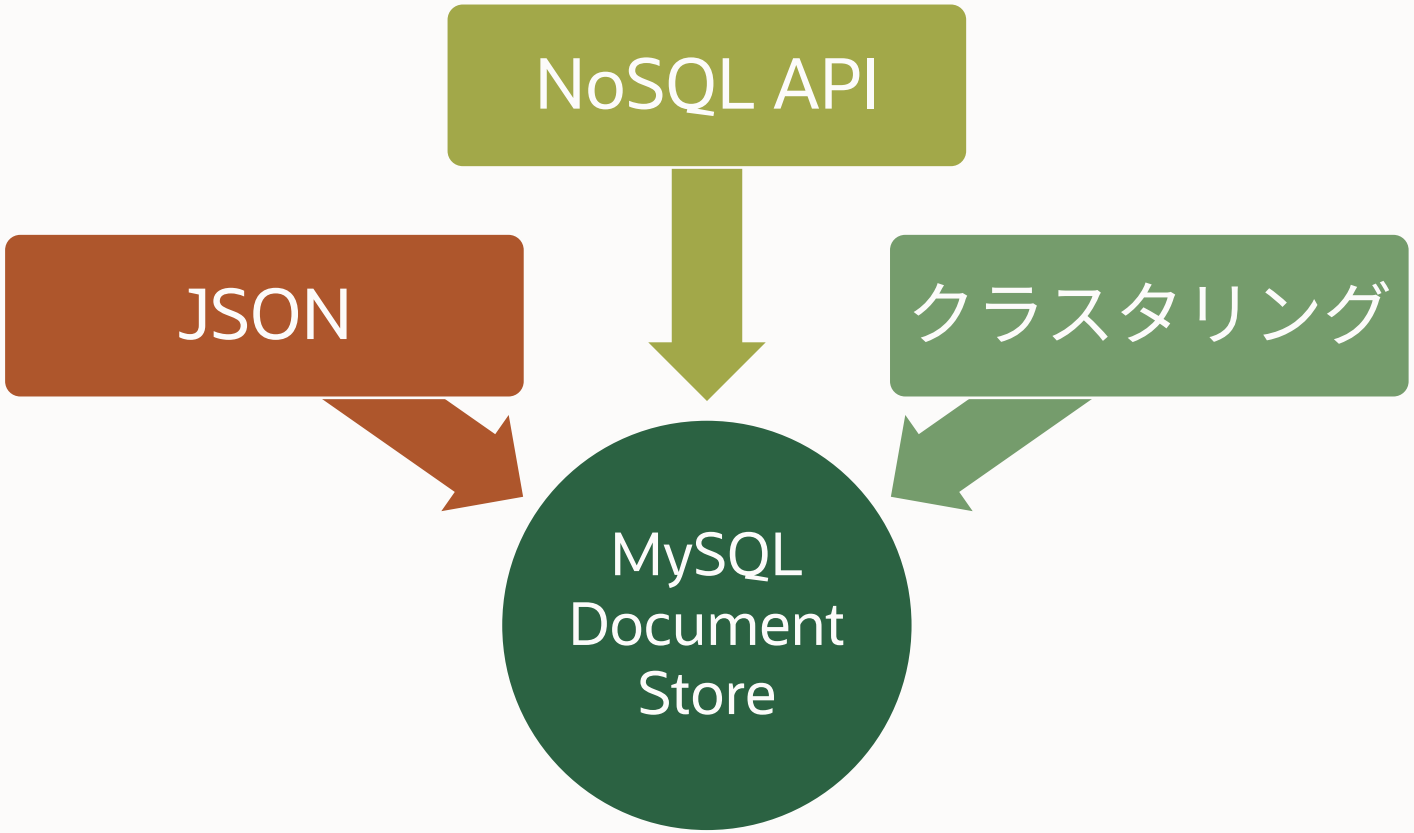
shop_order

id	id_customer	date	total
381	3412	2017-08-24	312.20
412	3412	2017-10-02	24.95

ドキュメントストアモデル

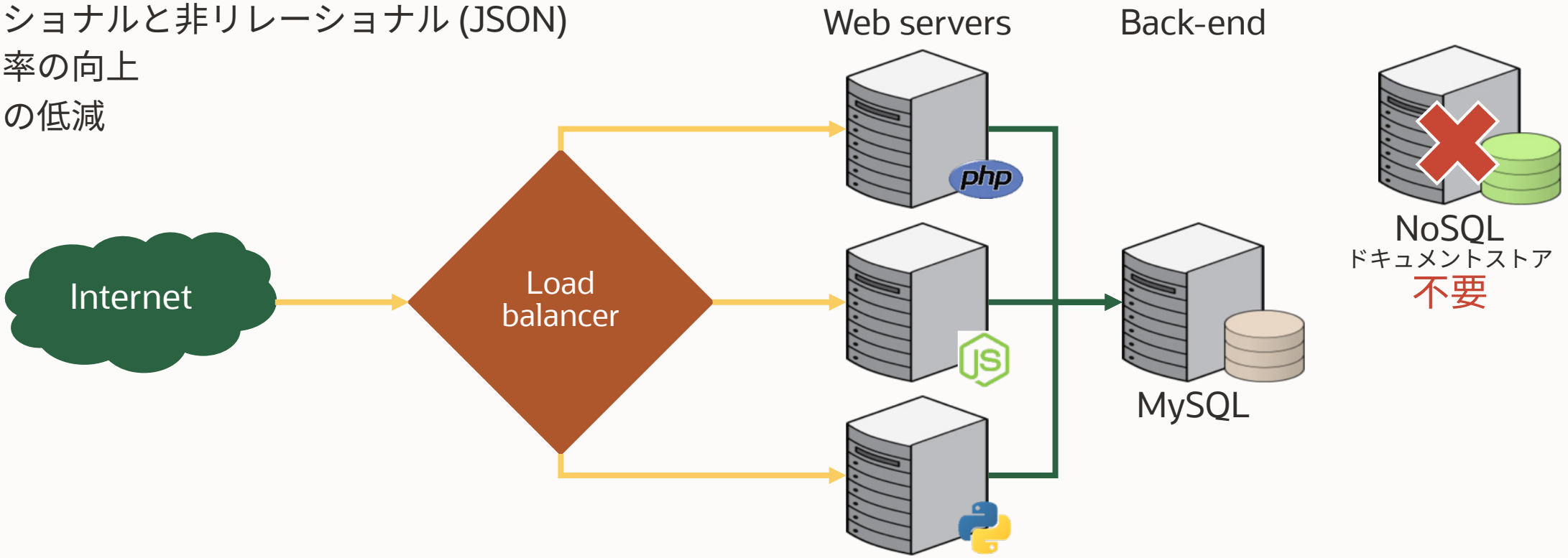
```
{
  _id: 3412,
  name: "John Smith",
  email: john@oracle.com,
  city: "San Francisco",
  country: "US",
  orders: [
    {date: "2017-08-24", total: 312.20},
    {date: "2017-10-02", total: 24.95}
  ]
}
```

MySQL Document Store



MySQL Document Store
ハイブリッド型バックエンド

- 単一のデータベースシステム
- リレーショナルと非リレーショナル (JSON)
- 運用効率の向上
- コストの低減



アプリケーション開発者に柔軟性を

データ型

JSON データ型

リレーショナルなテーブルと非構造データとシームレスに統合。さらに MySQL 8.0 では更新性能の最適化

SQL 関数

JSON 関数

JSON データの参照更新のための各種 SQL 関数を実装。MySQL 8.0 では JSON データを SQL で分析するための変換関数も追加

ハイブリッドAPI

MySQL X DevAPI

SQL と CRUD な NoSQL のハイブリッドAPIによる開発柔軟性

ドキュメントストアとSQL
JSONデータ型とJSON関数

Insert時のJSON構文バリデーション機能

ドキュメントにインデックス設定し高速アクセス

JSONドキュメントを操作する約30の関数群

- JSONデータを取得
  (JSON_EXTRACT, JSON_KEYS など)
- JSONデータや構造の検証
  (JSON_SEARCH, JSON_CONTAINS など)
- JSONデータの変更
  (JSON_SET, JSON_INSERT, JSON_REMOVE など)
- JSON配列やオブジェクトの生成
  (JSON_ARRAY, JSON_OBJECT)

```
CREATE TABLE employees (data JSON);

INSERT INTO employees VALUES
 ('{"id": 1, "name": "Jane"}'),
 ('{"id": 2, "name": "Joe"}');

SELECT * FROM employees;
+------------------------------------+
| data                               |
+------------------------------------+
| {"id": 1, "name": "Jane"}          |
| {"id": 2, "name": "Joe"}           |
+------------------------------------+

SET @document = '[10, 20, [30, 40]]';

SELECT JSON_EXTRACT(@document, '$[1]');
+-----------------------------------+
| JSON_EXTRACT(@document, '$[1]') |
+-----------------------------------+
| 20                                |
+-----------------------------------+
```

JSON_EXTRACTの省略演算子

Accepts a ***JSON Path***, which is similar to a selector:

$("#type")

JSON_EXTRACT
(column_name, "$.type")

JSON_EXTRACTでは、2つの省略形をサポート：

```
column_name->"$.type"    (= JSON_EXTRACT)
column_name->>"$.type"   (= JSON_EXTRACT + JSON_UNQUOTE)
```

MySQL Document Store
テーブルとJSONのJOIN

Collection people

```
{
  _id: 101,
  FirstName: "John",
  LastName: "Smith",
  Street: "123 Elm Street",
  State_ID: 5
}
```

Table states

id	name
1	Alabama
…	…
5	California
…	…
50	Wyoming

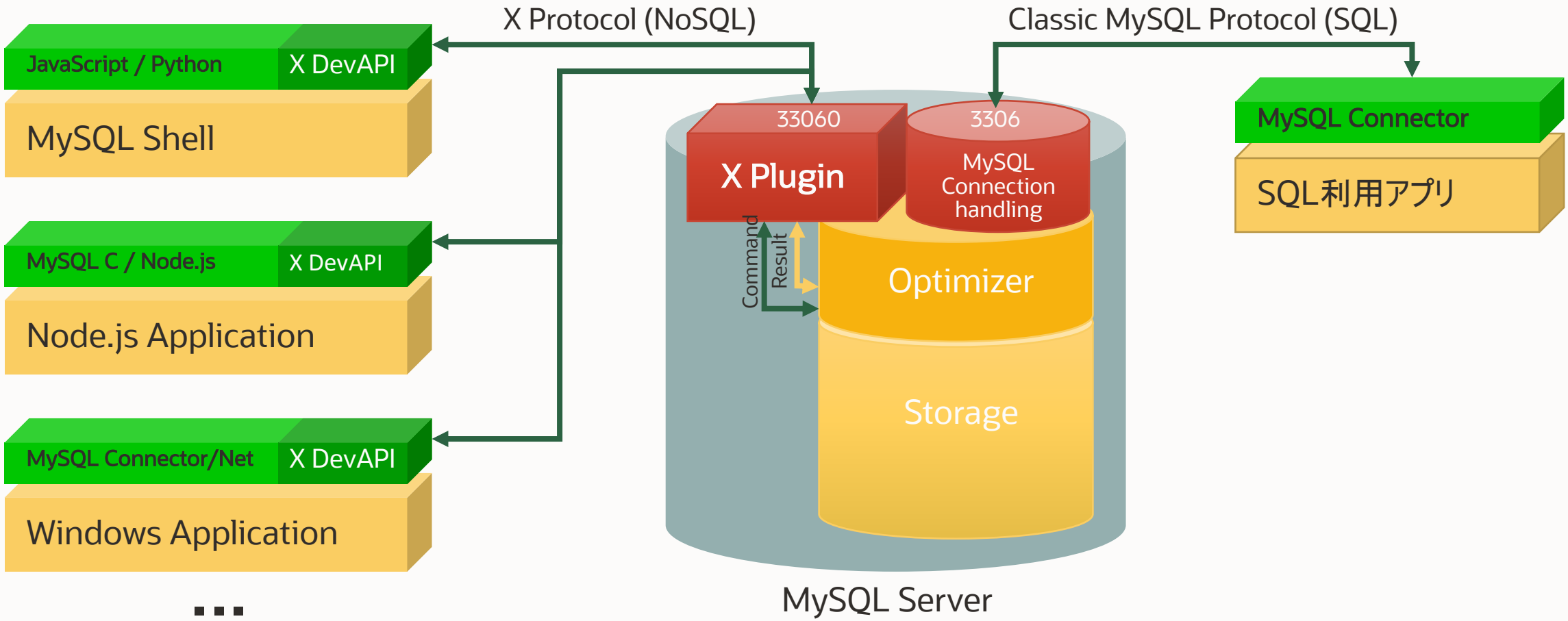
```
mysql> SELECT
    ->    `people`.`doc`->>'$.FirstName',
    ->    `people`.`doc`->>'$.LastName'
    -> FROM
    ->    `people`, `states`
    -> WHERE
    ->    `people`.`doc`->'$.State_ID' = `states`.`id` AND
    ->    `states`.`name` = 'California'
```

MySQL Document Store

JSONデータ型とWindow関数の組み合わせ

```
mysql> SELECT
    ->   `sales`.`doc`->>'$.employee' AS `Employee`,
    ->   `sales`.`doc`->>'$.date' AS `Date`,
    ->   `sales`.`doc`->'$.sales' AS `Sales`,
    ->   SUM(`sales`.`doc`->'$.sales')
    ->   OVER (PARTITION BY `sales`.`doc`->'$.employee') AS `Total`
    -> FROM `sales`;
+----------+------------+-------+-------+
| Employee | Date       | Sales | Total |
+----------+------------+-------+-------+
| Odin     | 2019-03-01 |   300 |  1200 |
| Odin     | 2019-03-02 |   400 |  1200 |
| Odin     | 2019-03-03 |   500 |  1200 |
| Thor     | 2019-03-01 |   200 |  1400 |
| Thor     | 2019-03-02 |   400 |  1400 |
| Thor     | 2019-03-03 |   800 |  1400 |
+----------+------------+-------+-------+
6 rows in set (0.0011 sec)
```

MySQL Native NoSQL API



X DevAPI

X Plugin (MySQL) ⇔ X Protocol ⇔ X DevAPI (Driver)

X Pluginを有効にする事で、X Protocol経由で通信可能
ドキュメントとテーブルのコレクションに対してのCRUD処理
NoSQLライクな構文でドキュメントに対しCRUD処理可能
Fluent API

Tables or Collections?

MySQL Document Storeにおけるコレクションは2つの列を持つ特別なテーブル:

- 主キー: `_id`
- JSONドキュメント: `doc`
  - JSONドキュメントの`_id`フィールドは任意の値かサーバーがUUIDを自動生成
  - このフィールドの値が主キーとして展開される

必要に応じて列やインデックスの追加は可能

SQL, NoSQL, テーブル, コレクションはそれぞれシームレスに利用可能

すべての操作はレプリケーションに反映される

SHOW CREATE TABLE `myCollection`¥G

```
Table: myCollection
Create Table: CREATE TABLE `myCollection`
(
  `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=utf8mb4
```

MySQL Connectors include X Dev API

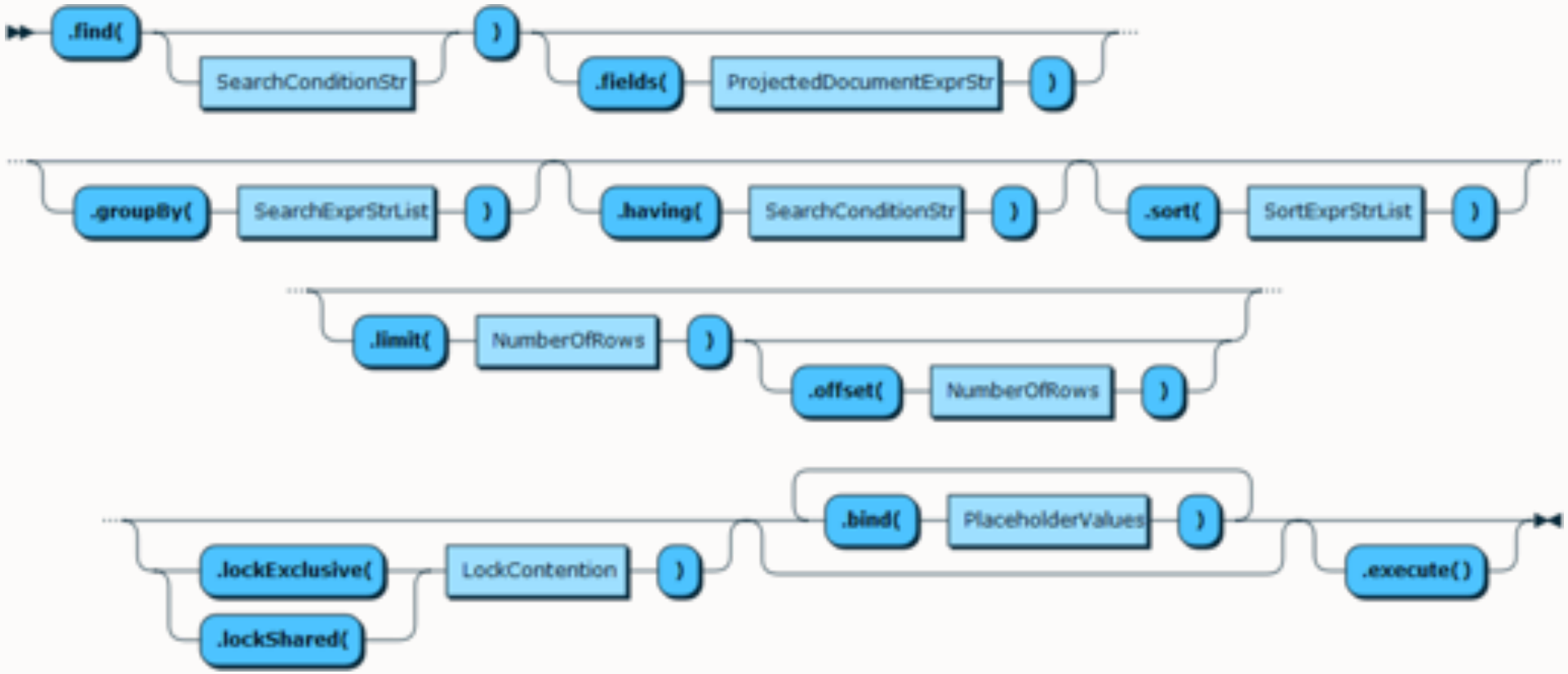
Use SQL, CRUD APIs
スキーマレスドキュメントおよびリレーショナルテーブルに対応

- Classic APIsに加えて、これらの全てが追加されます

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

参照) http://dev.mysql.com/doc/x-devapi-userguide/en/crud-operations-overview.html

ドキュメントの検索




```
products.find("color = 'yellow'").sort(["name"]).execute();
```

言語仕様

SQLのサブセット (利用は簡単、かつパワフル)
読みやすく構造が理解しやすい
すべてのConnectorで可能な限り共通の構文 (なるべく移植しやすい構文)

```
// JavaScript
collection
  .find("name = 'foo' AND age > 42")
  .fields("name", "age")
  .groupBy("name", "age")
  .sort("name ASC", "age DESC")
  .limit(4)
  .offset(2)
  .execute()
```

```
// Java
collection
  .find("name = 'foo' AND age > 42")
  .fields("name", "age")
  .groupBy("name", "age")
  .sort("name ASC", "age DESC")
  .limit(4)
  .offset(2)
  .execute()
```

MySQL Native NoSQL
Node.JSの例

```
var schema = session.getSchema("ProductsDB");
var products = schema.getCollection("products_collection");
var query = "$.name == :name";
products.find(query)
        .bind("name", "Vacuum cleaner")
        .execute(function (doc) {
  console.log(doc);
}).catch(function (err) {
  console.log(err.message);
  console.log(err.stack);
});
```

MySQL Shellの特徴

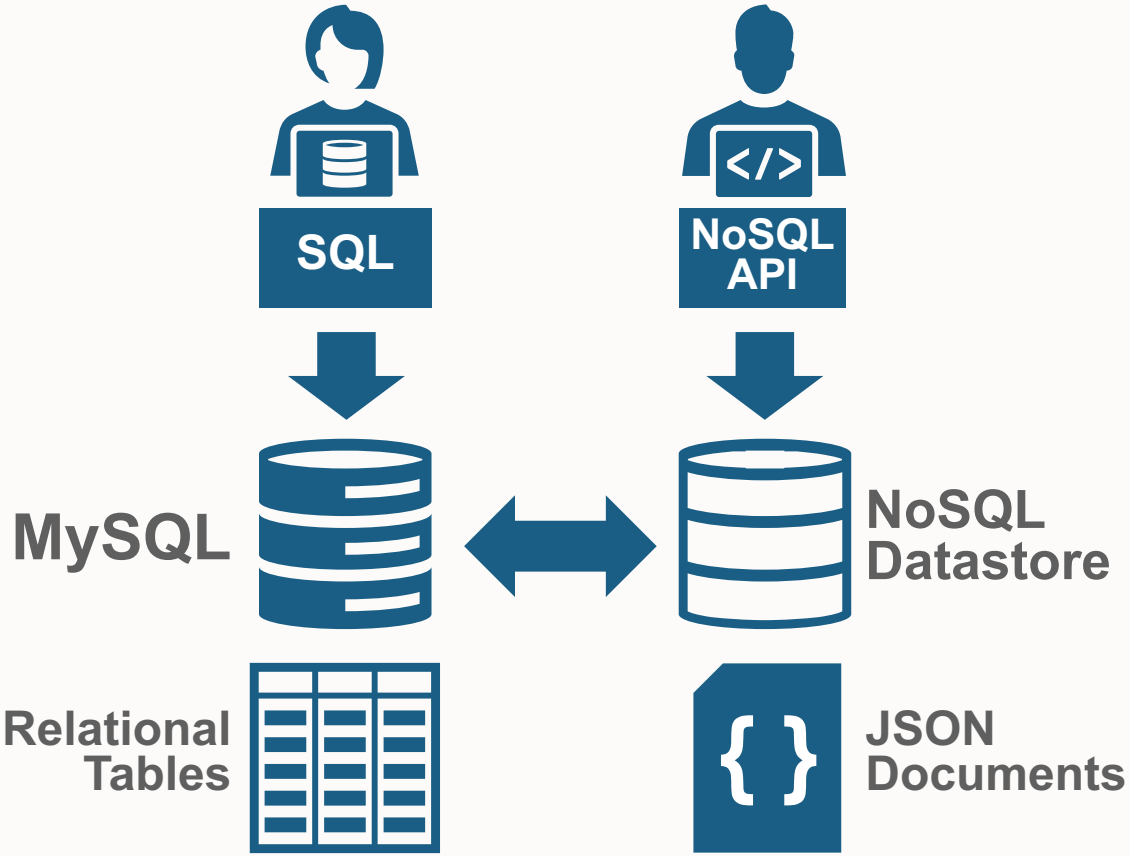
① 多言語をサポート
- JavaScript
- Python
- SQL

② 実行形式を選択可能
- バッチ
- インタラクティブ

③ 各種ユーティリティ＋拡張機能
- アップグレードチェッカー
- JSONインポート
- 高速データロード
- レポート機能
- プラグイン機能

④ 統合されたAPI
- ドキュメントストア操作
- InnoDBクラスタ管理

RDBMSとNoSQLデータストアを併用する際の懸念事項



開発者

複数のAPIを学習する必要がある

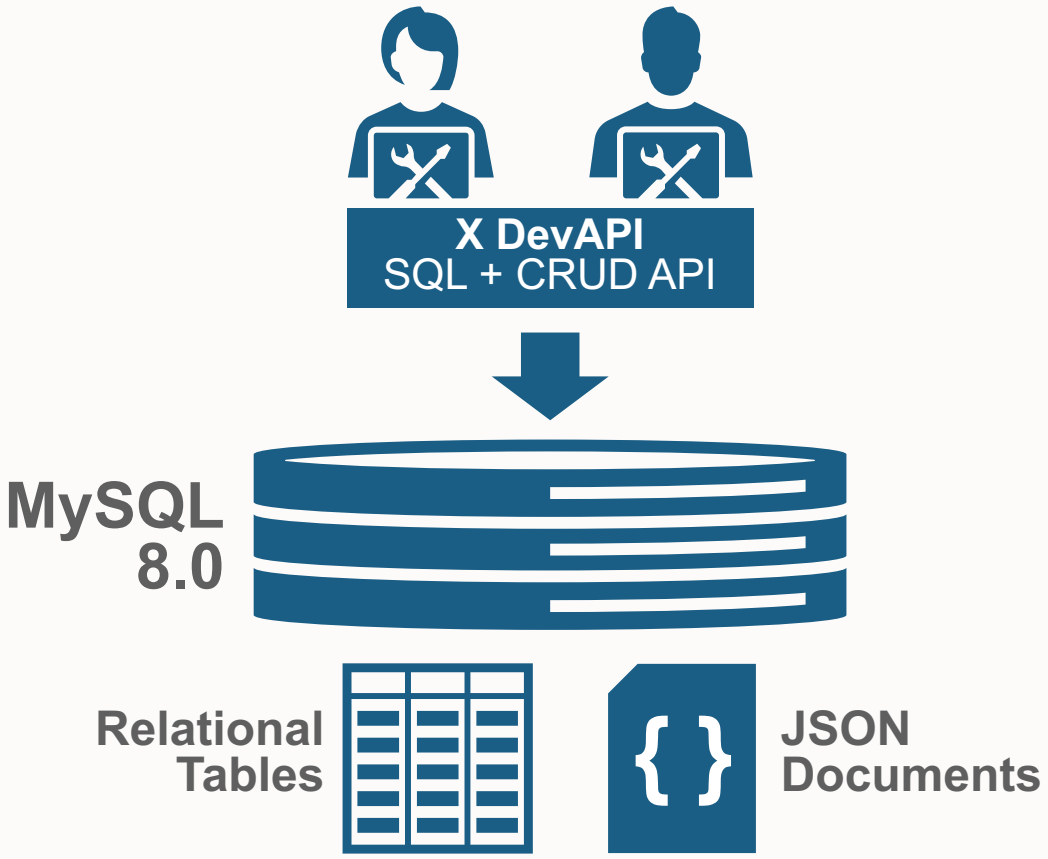
データ管理

テーブルとJSONドキュメントの
確実なデータ同期が困難

運用

個別に運用ツールを導入して
別々の運用管理が求められる

MySQL Document Store: NoSQL + SQL = MySQL 8.0



開発者にとっての柔軟性

統合されたAPIによる柔軟性

データ管理の信頼性と柔軟性

単一のデータストアなのでデータ同期不要 テーブルとJSONドキュメントのJOINも可能

運用効率の向上

単一データベースのみの運用で済むので管理負荷低減

NoSQL + SQL = MySQL

- Flexible APIs for Developers
- Hybrid Data Models
- Proven Transaction Management
- Reliable Data Consistency
- Simplified Operations

宣伝：シェルスクリプトマガジン Vol.61から66

世界で(たぶん)唯一のMySQL ShellやX DevAPIに関する連載

第1回: MySQL Shell概要

第2回: MySQL Shellの運用ユーティリティ

第3回: X Dev API JavaScript編

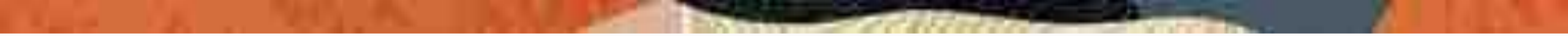
第4回: X Dev API Python編

第5回: InnoDB Cluster

第6回: InnoDB ReplicaSet

バックナンバー販売中　https://shell-mag.com/category/back-number/

Thank you



ORACLE