Presented with

MySQL

MySQL Document Store를 활용한 NoSQL 개발

Sumi Ryu
MySQL Principal Solution Engineer
May 18, 2019

ORACLE

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.

류수미

LGCNS : Java 개발 및 AA, Laf/J@Fin

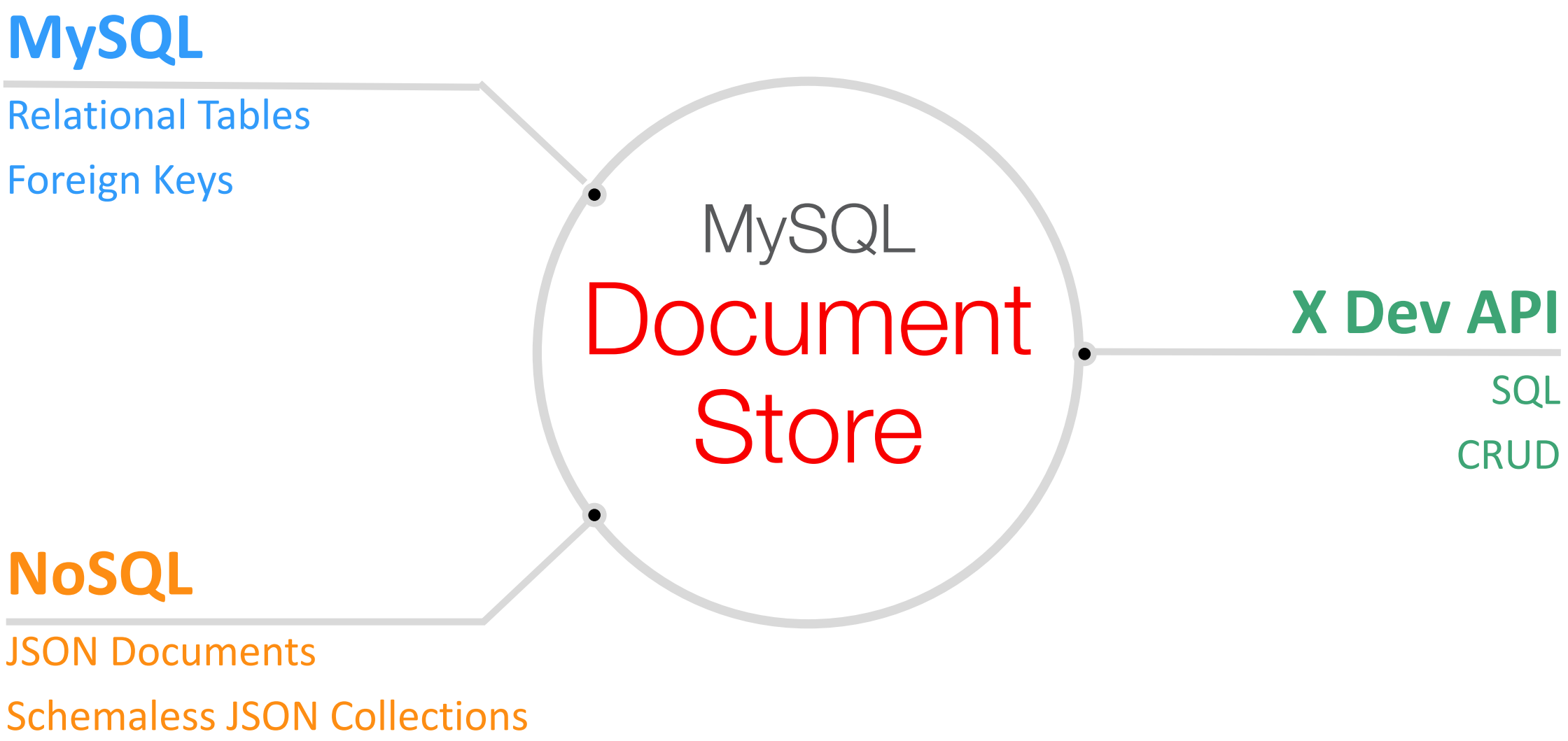
SUN : Pre sales for Web, WAS, Java EE, MySQL

Oracle : Pre sales for MySQL

sumi.ryu@oracle.com

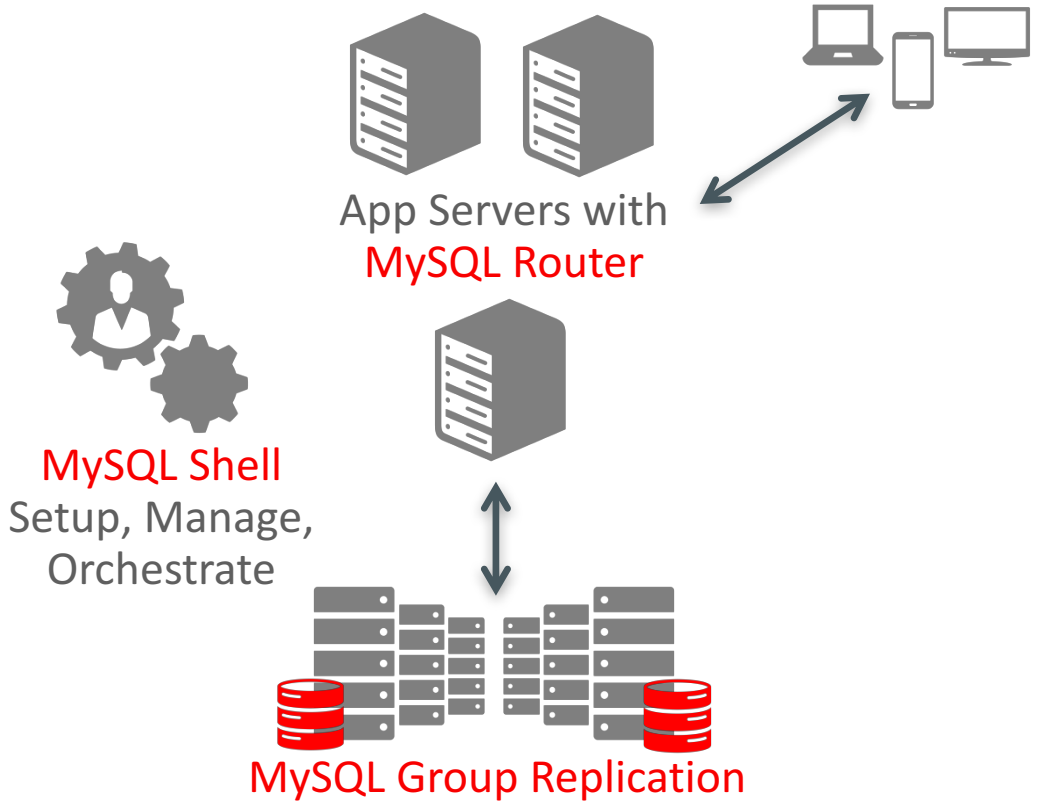
MySQL Presentations at OOW

ORACLE



MySQL InnoDB Cluster

"High Availability becomes a core first class feature of MySQL!"



ORACLE

NoSQL의 유혹

- 대부분의 사용자 / 회사는 어느 시점에서 NoSQL을 사용해 보도록 강요 받고 있음
- 그들은 RDBMS가 확장되지 않거나 개발자가 SQL 또는 다른 것을 좋아하지 않기 때문이라고 믿음
- 나중에서야 모든 제한사항을 알게 됨 (보통 처음부터 완전히 새로운 도구를 배울 수있는 충분한 시간이 없었기 때문에)
- 결과는 대개 다음 중 하나:
  - JSON을 지원하는 일부 RDBMS 사용 (소규모 회사 / 사용자)
  - 신뢰할 수 있는 스토리지 엔진 (예 : 우버의 Schemaless 와 Facebook 의 MyRocks/RocksDB) 위에 NoSQL 인터페이스를 구현

What is a Document Store Database?

What is Document Store Database?

A document store database (also known as a document-oriented database, aggregate database, or simply document store or document database) is a database that uses ***a document-oriented model*** to store data.

Document store databases store each record and its associated data within a single document. Each document contains semi-structured data that can be queried against using various query and analytics tools of the DBMS.

출처: https://database.guide/what-is-a-document-store-database/

예제

< XML >

```
<artist>
  <artistname>Iron Maiden</<artistname>
  <albums>
    <album>
      <albumname>The Book of Souls</albumname>
      <datereleased>2015</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Killers</albumname>
      <datereleased>1981</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Powerslave</albumname>
      <datereleased>1984</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Somewhere in Time</albumname>
      <datereleased>1986</datereleased>
      <genre>Hard Rock</genre>
    </album>
  </albums>
</artist>
```

< JSON >

```
{
    '_id' : 1,
    'artistName' : { 'Iron Maiden' },
    'albums' : [
        {
            'albumname' : 'The Book of Souls',
            'datereleased' : 2015,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Killers',
            'datereleased' : 1981,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Powerslave',
            'datereleased' : 1984,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Somewhere in Time',
            'datereleased' : 1986,
            'genre' : 'Hard Rock'
        }
    ]
}
```

The MySQL Approach

MySQL Document-store

- “관계형 데이터베이스와 문서 데이터베이스 간의 크로스오버”
- 다양한 언어 및 환경을 커버하는 공통 API
  - **C++**, **Java**, **.Net**, **Node.js**, **Python** 및 **PHP**
  - 새로운 MySQL Shell
  - 관계형 테이블에 대한 쉬운 CRUD 작업 및 문서 데이터베이스 형태의 모음
- MySQL 재단에 기반
  - InnoDB
  - Replication
  - ....

The MySQL Approach

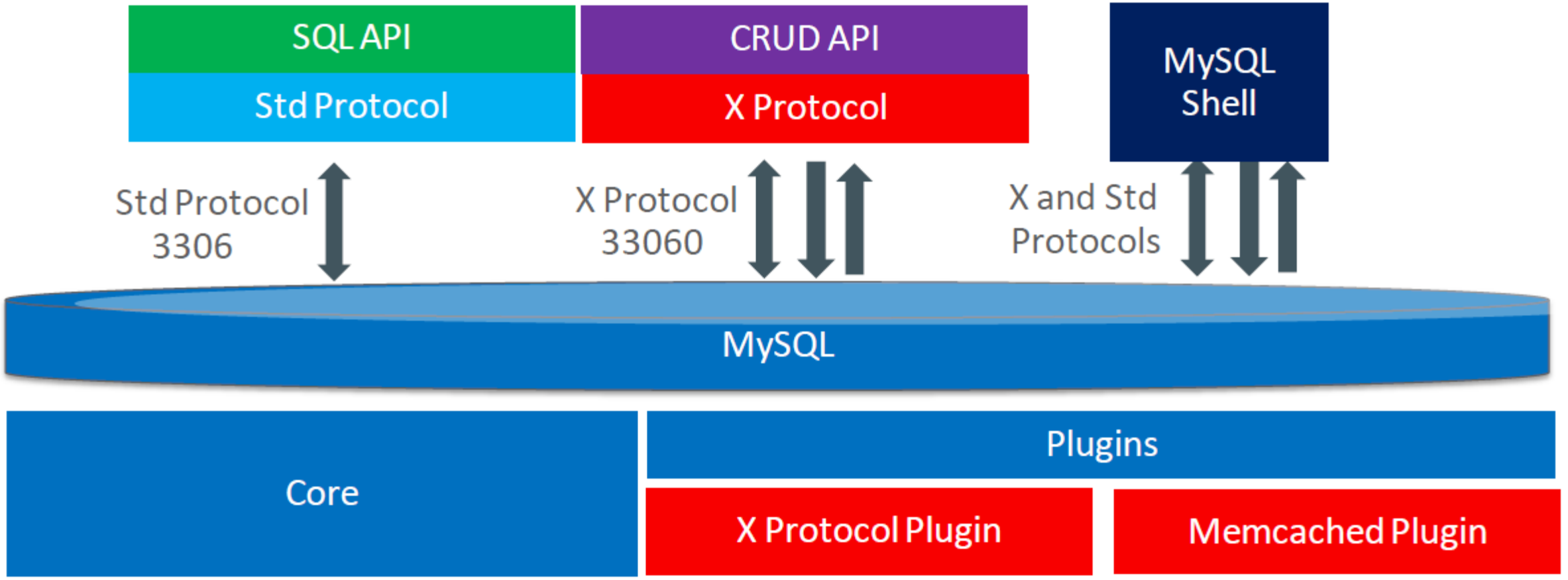
Documents and Collections

- Docstore의 문서는 JSON으로 표현됨
  - “JavaScript Object Notation”
  - 크로스 플랫폼 직렬화 포멧 (웹 서비스에서 공통)
  - ECMA-404 로 표준화됨 (http://json.org)
  - MySQL에서 잘 지원됨
- 컬렉션 내에 여러 문서가 저장됨
  - 기술적으로, 하나의 **InnoDB** 테이블임
  - JSON 유형의 일반적인 하나의 컬럼
  - 인덱스는 가상 컬럼으로 생성가능

MySQL DocStore

- X Plugin
- X Protocol
- X DevAPI

MySQL DocStore Architecture



MySQL DocStore Components – X Plugin

MySQL Document Store는 MySQL 5.7.12+, MySQL 8 용 플러그인으로 존재합니다.이 플러그인은 기본적으로 활성화되어 있으며, 기존 MySQL 8 서버에 전체 문서 저장 레이어를 제공합니다.

- 일반 MySQL 3306 대신 다른 포트 세트 사용
- 프로토콜 계층에서 Google protobuf 메시지를 사용
- 클라이언트로부터 DevAPI 요청을 받아들이고 코어 MySQL 엔진을 사용하여 요청을 처리
- https://dev.mysql.com/doc/refman/8.0/en/document-store.html

The MySQL Approach

X Protocol

- X Plugin에 의해 구현 된 새로운 프로토콜
- Google 프로토 버퍼 기반 (향후 확장을 위한 유연성)
- 더 나은 콘텐츠 이해
  - CRUD 표현(expression) 트리
  - 파이프 라인 구문에 조건을 첨부할 수 있을 것에 대한 기대
  - 바운드 파라미터
- 보안
  - 기본적으로 SSL/TLS 제공
  - 인증되지 않은 사용자에게 정보가 누출되지 않음(예 : MySQL 서버 버전)

The MySQL Approach

X Protocol

```
Mysqlx.Crud.Find {
  collection { name: "collection_name", schema: "test" }
  data_model: DOCUMENT
  criteria {
    type: OPERATOR
    operator {
      name: "=="
      param {
        type: IDENT,
        identifier { name: "_id" }
      }
      param {
        type: LITERAL,
        literal {
          type: V_STRING,
          v_string: { value: "some_string" }
        }
      }
    }
  }
}
```

X DevAPI

문서 및 SQL을 위한 공통 API

MySQL DocStore Components - DevAPI

X DevAPI는 클러스터 형태의 MySQL 설정으로 구동되는 CRUD 기반 응용 프로그램을 쉽게 개발할 수있게 해주는 고급 데이터베이스 API입니다. NoSQL 문서(document) 작업의 CRUD 사용 편의성에 초점을 맞추면서 전문가 수준의 제어와 원시 SQL을 포함한 고급 기능에 대한 액세스를 제공합니다.

- 가장 대중적인 프로그램 언어로 개발이 가능
- 문서와 관계형 테이블을 쿼리하는 강력한 방법을 제공
- SQL을 직접 실행할 수 있음
- 라우터를 통해 InnoDB 클러스터를 작업 할 수 있음
- 개발자 안내서 및 X DevAPI 참조 설명서는 다음에서 찾을 수 있음 :
  - https://dev.mysql.com/doc/connector-python/en/
  - https://dev.mysql.com/doc/x-devapi-userguide/en/

특징

- 유연한 파라미터를 가진 능숙한 API
  - 공통 표현식(common expression) 언어를 기반으로 한 쿼리 빌더
  - 원시 SQL 인터페이스 또한 사용 가능
- 보안을 기본적으로 제공 (SSL/TLS 및 SHA256 인증)
- 트랜잭션, 세이브 포인트 및 행 잠금(row-locking)을 통한 데이터 일관성
- 속도 향상을위한 C / C ++, Python 확장

Fluent API

- 단일 진입점으로 부터 코드는 흘러감 – ***getSession()***
- 코드는 가독성이 높아지고, 유지 보수하기 좋음 (심지어 테스트도 가능)
- 특수화되고 시맨틱 방법으로 캡슐화된 작업들
- 리펙터링 작업을 위한 멋진 발판
- 텍스트 편집기 (또는 IDE) 힌트 및 자동 완성을 위한 최고수준의 지원
- 더 작아진 SQL 인젝션 대상 영역
- 다른 프로그래밍 환경 간의 공통 표준

Flexible Parameters

- 대부분의 공개 API 메소드는 다음과 함께 동작:
  - 다양한 개별 파라미터
  - 파라미터의 단일 배열
  - 이름이 붙여진 속성들을 가진 객체 (그것이 적용되는 곳)

```
mysqlx.get_session("root:@localhost")
mysqlx.get_session({user: "root"})

collection.add({name: "foo"}).add({name: "bar"})
collection.add([{name: "foo"}, {name: "bar"}])

collection.find("name = :name").bind("name", "foo")
collection.find("name = :name").bind({name: "foo"})
collection.find().fields("foo", "bar")
collection.find().fields(["foo", "bar"])
```

Expression Language

- SQL 하위 집합 (배우기 쉬우면서 여전히 강력 함)
- 표현적이고 인간이 읽을 수 있는
- 모든 공식 커넥터 구현 간의 공통 표준 (이식 가능하도록)

```python
# Python
collection \
  .find("name = 'foo' AND age > 42") \
  .fields("name", "age") \
  .group_by("name", "age") \
  .sort("name ASC", "age DESC") \
  .limit(4) \
  .offset(2) \
  .execute()
```

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

SSL Modes

보안은 기본으로 제공

- 세션 생성시, TCP 연결에 대해 SSL / TLS는 기본적으로 활성화되어 있음
- 로컬 UNIX 소켓 연결을 사용하지 않도록 설정 (필요 없음)하면 더 빠름
- 추가적으로 서버 인증서 유효성 검사
- 옵션들을 on-demand 로 재정의 할 수 있음

```
mysqlx.get_session(user="root", password="", ssl_mode=mysqlx.SSLMode.DISABLED)
mysqlx.get_session("mysqlx://root:@localhost?ssl-mode=DISABLED")
mysqlx.get_session(user="root", password="", ssl_mode=mysqlx.SSLMode.VERIFY_CA,
                   ssl_ca="/path/to/ca.pem")
mysqlx.get_session("mysqlx://root:@localhost?ssl-mode=VERIFY_CA&ssl-ca=/path/to/ca.pem")
mysqlx.get_session(user="root", password="", ssl_ca="/path/to/ca.pem",
                   ssl_key="/path/to/key.pem", ssl_cert="/path/to/cert.pem")
mysqlx.get_session("mysqlx://root:@localhost?ssl-ca=/path/to/ca.pem&"
                   "ssl-key=/path/to/key.pem&ssl-cert=/path/to/cert.pem")
```

Authentication Mechanisms

Secure by default (as well)

- SHA-1 및 SHA-2 암호 해시
- 다음 서버 플러그인에 대한 인증 메커니즘 지원 :
  - **mysql_native_password**
  - **sha256_password**
  - **caching_sha2_password**

```
mysqlx.get_session(user="root", password="", auth=mysqlx.Auth.MYSQL41)
mysqlx.get_session("mysqlx://root:@localhost?auth=MYSQL41")
mysqlx.get_session(user="root", password="", auth=mysqlx.Auth.PLAIN)
mysqlx.get_session("mysqlx://root:@localhost?auth=PLAIN")
mysqlx.get_session(user="root", password="", auth=mysqlx.Auth.SHA256_MEMORY)
mysqlx.get_session("mysqlx://root:@localhost?auth=SHA256_MEMORY")
```

Transactions and Savepoints

Session-level atomic operations

- 세션 범위에서 트랜잭션 생성, 커밋 또는 롤백
- 이들 트랜잭션의 중간 세이브포인트로 생성, 릴리스 또는 롤백

```
session.start_transaction()

collection.add({"name": "Wilma", "age": 33}).execute()
assert(2 == collection.count())

session.set_savepoint("sp")

collection.add({"name": "Barney", "age": 42}).execute()
assert(3 == collection.count())

session.rollback_to("sp")
assert(2 == collection.count())

session.commit()
```

Row Locking

Document-level isolation

- 문서를 읽는 쿼리는 동시 트랜잭션이있을 때 격리 수준을 결정할 수 있음
- 두 종류의 잠금(lock):
  - Exclusive locks
  - Shared locks
- 세가지 다른 동작 모드:
  - ***DEFAULT***
  - ***NOWAIT***
  - ***SKIP_LOCKED***

Raw SQL

만물박사 (Jack of all trades)

- 구조화 된 분석 및 / 또는 보고의 경우, SQL은 항상 옵션임
- 다음과 같이 X DevAPI의 일부가 아닌 기능을 이용할 수 있음:
  - 관계형 테이블에 대한 DDL 조작
  - 키 제약 조건 관리
  - JOIN 사용

```
# create a table
session.sql('CREATE TABLE foo (bar VARCHAR(3))').execute()
# add a unique constraint
session.sql('ALTER TABLE foo ADD COLUMN bar VARCHAR(3) GENERATED ALWAYS AS doc->>"$.bar" '
            'VIRTUAL UNIQUE KEY NOT NULL').execute()
# execute a JOIN query
session.sql('SELECT DISTINCT t1.bar FROM foo t1 JOIN baz t2 ON t1.bar = t2.qux WHERE t1.qux = t2.quux').execute()
```

ORACLE

MySQL Connector/Python

Python으로 MySQL Document Store 활용해서 NoSQL 개발

Python

- 다양한 목적 (Web, GUI, 스크립팅, 데이터 과학, etc.)
- 객체 지향
- 해석되고(Interpreted) 동적으로 입력되는 언어
- 가독성 및 생산성에 중점을 둠
- batteries included
- 강한 내성
- 크로스 플랫폼
- C / C ++을 사용하여 확장 가능하도록 설계된 Extensible 은 많은 외부 라이브러리에 대한 액세스 허용
- GitHub에서 가장 인기있는 두 번째 언어
- 구현 : CPython, Jython, IronPython, PyPy

MySQL DocStore Components - Connector/Python

MySQL Connector / Python은 X DevAPI 프로토콜 용 클라이언트 라이브러리를 제공하며 Python 프로그램이 Python Database API Specification v2.0 (PEP 249)을 준수하는 API를 사용하여 MySQL 데이터베이스에 액세스 할 수 있도록함

- 동일한 패키지에서 MySQL classic 및 X DevAPI 프로토콜 지원
- 동일한 코드 기반을 사용하는 Python 2.7, 3.4, 3.5, 3.6 지원
- 속도 향상을위한 선택적인 C / C ++ Python 확장
- 프로토콜 압축 (MySQL 클래식 프로토콜)
- TCP / IP 소켓을 사용한 연결 및 Unix에서는 Unix 소켓을 사용
- SSL을 사용한 안전한 TCP / IP 연결
- Linux, Mac 및 Windows (또는 Python이 설치된 모든 플랫폼)에서 이용가능

ORACLE

MySQL DocStore Components - Connector/Python

설치

- pip 를 이용해서 Connector / Python 설치 (권장 방법)

```
shell> pip install mysql-connector-python
```

- 사용가능한 플랫폼 패키지
  - Linux: RPM, DEB
  - MacOS: DMG
  - Windows: MSI Package
  - Platform Independent: tar.gz

MySQL DocStore Components - Connector/Python

소스로 설치

- 사전 준비사항
  - gcc와 같은 C / C ++ 컴파일러
  - Protobuf C++ (버전 3.0.0 이상)
  - Python 개발 파일
  - MySQL C API 라이브러리와 인터페이스하는 선택적인 C 확장팩을 컴파일하기위한 개발 파일을 포함해서 MySQL 서버가 설치됨

MySQL DocStore Components - Connector/Python

소스로 설치 (Linux)

- tar 아카이브로 Connector / Python을 설치하기 위해 최신 버전 (여기에서는 <version>으로 표시)을 다운로드

```
shell> tar xzf mysql-connector-python-<version>.tar.gz
shell> cd mysql-connector-python-<version>.tar.gz
shell> python setup.py --with-protobuf-include-dir=<protobuf-include-dir> \
       --with-protobuf-lib-dir=<protobuf-lib-dir> --with-protoc=<protoc-binary>
```

- MySQL C 클라이언트 라이브러리와 인터페이스하는 C 확장팩을 포함 시키기위해 --with-mysql-capi 옵션을 추가

```
shell> python setup.py --with-protobuf-include-dir=<protobuf-include-dir> \
       --with-protobuf-lib-dir=<protobuf-lib-dir> --with-protoc=<protoc-binary> --with-mysql-capi=<mysql-capi>
```

MySQL DocStore Components - Connector/Python

Getting Started – 서버로 연결

```
import mysqlx

# Using a connection string
session = mysqlx.get_session("root:myql@localhost:33060")

# Using kwargs
session = mysqlx.get_session(host="localhost", port=33060, user="root", password="myql")

# Using a Python dictionary
session = mysqlx.get_session({"host": "localhost", "port": 33060, "user": "root", "password": "myql"})
```

MySQL DocStore Components - Connector/Python

Getting Started – 스키마 및 콜렉션 생성

```
import mysqlx

# Connect to server
session = mysqlx.get_session(host="localhost", port=33060, user="root", password="myql")

# Create a schema
schema = session.create_schema("test")

# Create a collection
collection = schema.create_collection("my_collection")
```

MySQL DocStore Components - Connector/Python

Getting Started – 데이타 삽입

```
# Use the collection "my_collection"
collection = schema.get_collection("my_collection")

# Add a document to "my_collection" using a Python dictionary
collection.add({"name": "Sakila", "age": 15}).execute()

# Add a document to "my_collection" using a JSON string
collection.add('{"name": "Sakila", "age": 15}').execute()

# Add a document to "my_collection" using a DbDoc object
doc = mysqlx.DbDoc({"name": "Sakila", "age": 15})
collection.add(doc).execute()

# Add a document to "my_collection" specifying an _id
collection.add({"_id": "1", "name": "Sakila", "age": 15}).execute()

# You can also add multiple documents at once
collection.add({"name": "Sakila", "age": 15},
               {"name": "Jack", "age": 32},
               {"name": "Clare", "age": 37}).execute()
```

MySQL DocStore Components - Connector/Python

Getting Started – 문서 찾기

```
# Use the collection "my_collection"
collection = schema.get_collection("my_collection")

# Specify which document to find with Collection.find()
result = collection.find("name like :param").bind("param", "S%").limit(1).execute()

doc = result.fetch_one() # Fetch one document

print(doc, type(doc))
# {"_id": "00005b48c75b0000000000000025", "age": 15, "name": "Sakila"} <class 'mysqlx.dbdoc.DbDoc'>

# Get all documents where age > 18
result = collection.find("age > 18").execute()

docs = result.fetch_all() # Fetch all documents

print(docs)
# [{'_id': '00005b48c75b000000000000002f', 'age': 32, 'name': 'Jack'},
#   '_id': '00005b48c75b0000000000000030', 'age': 37, 'name': 'Clare'}]
```

MySQL DocStore Components - Connector/Python

Getting Started – 문서 수정

```
# Use the collection "my_collection"
collection = schema.get_collection("my_collection")

# Get document with name = 'Jack'
result = collection.find("name = 'Jack'").execute()
doc = result.fetch_one()
print(doc)
# {"_id": "00005b48c75b000000000000002f", "age": 32, "name": "Jack"}

# Set age to 21
result = collection.modify("name = 'Jack'").set("age", 21).execute()

result = collection.find("name = 'Jack'").execute()
doc = result.fetch_one()
print(doc)
# {"_id": "00005b48c75b000000000000002f", "age": 21, "name": "Jack"}
```

MySQL DocStore Components - Connector/Python

Getting Started – 문서 삭제

```
# Add documents
collection.add({"_id": 1, "name": "Sakila", "age": 15},
               {"_id": 2, "name": "Jack", "age": 32},
               {"_id": 3, "name": "Clare", "age": 37}).execute()
print("Total documents in collection:", collection.count())
# Total documents in collection: 3

collection.remove("_id = 1").execute()
print("Total documents in collection:", collection.count())
# Total documents in collection: 2

result = collection.find().execute()
docs = result.fetch_all()
print(docs)
# [{'_id': 2, 'age': 32, 'name': 'Jack'},
#  {'_id': 3, 'age': 37, 'name': 'Clare'}]
```

MySQL DocStore Components - Connector/Python

Getting Started – 인덱스 생성

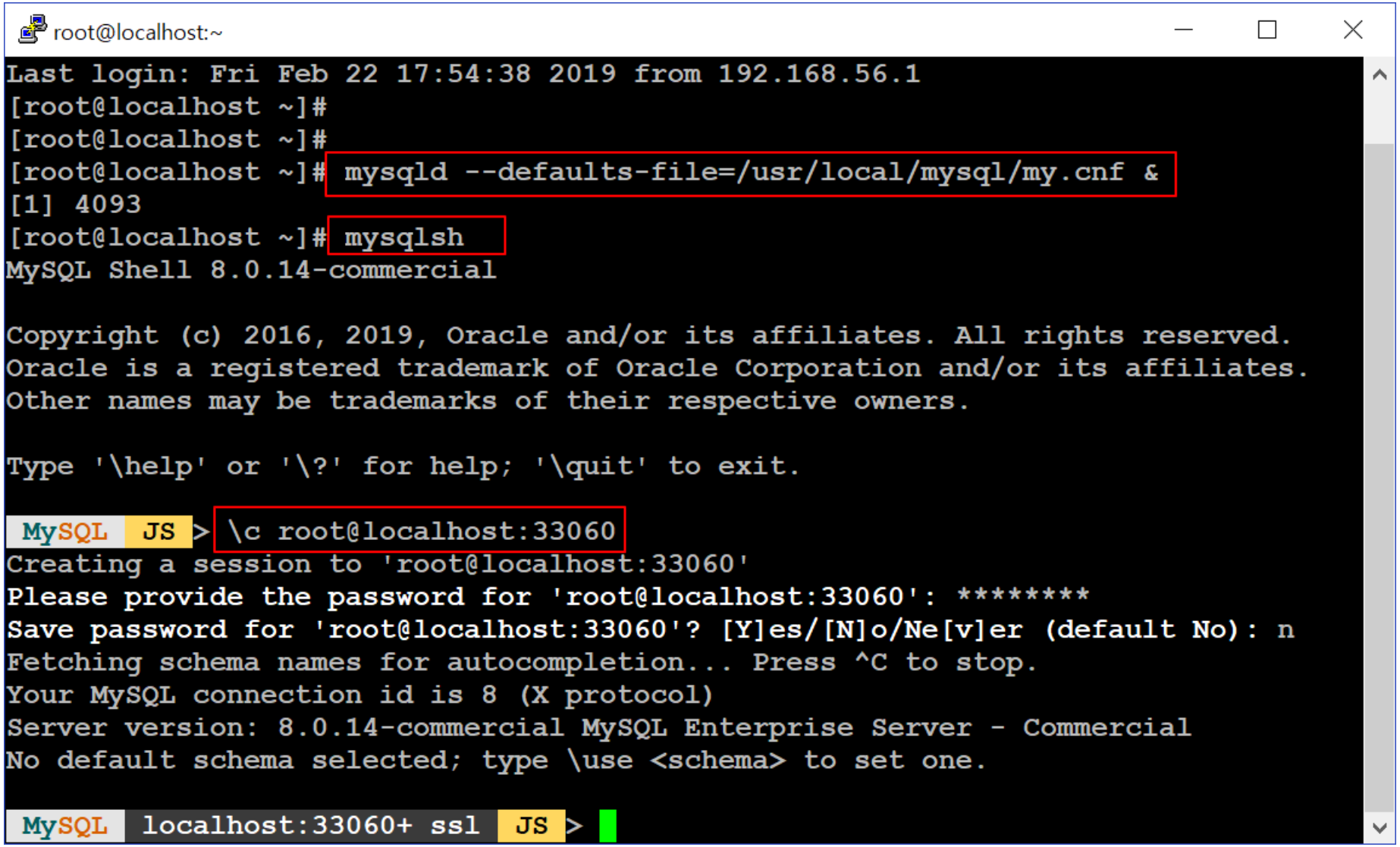
```
# Create index on 'age'
collection.create_index("index_age", {"fields": [{"field": "age",
                                                  "type": "INT"}],
                                       "type":"INDEX"})

# {"fields": [{"field": member_path,
#              "type": member_type,
#              "required": member_required,
#              "collation": collation,
#              "options": options,
#              "srid": srid},
#           # {... more members,
#           # repeated as many times
#           # as needed}
#           ],
# "type": type}
```

데모 환경

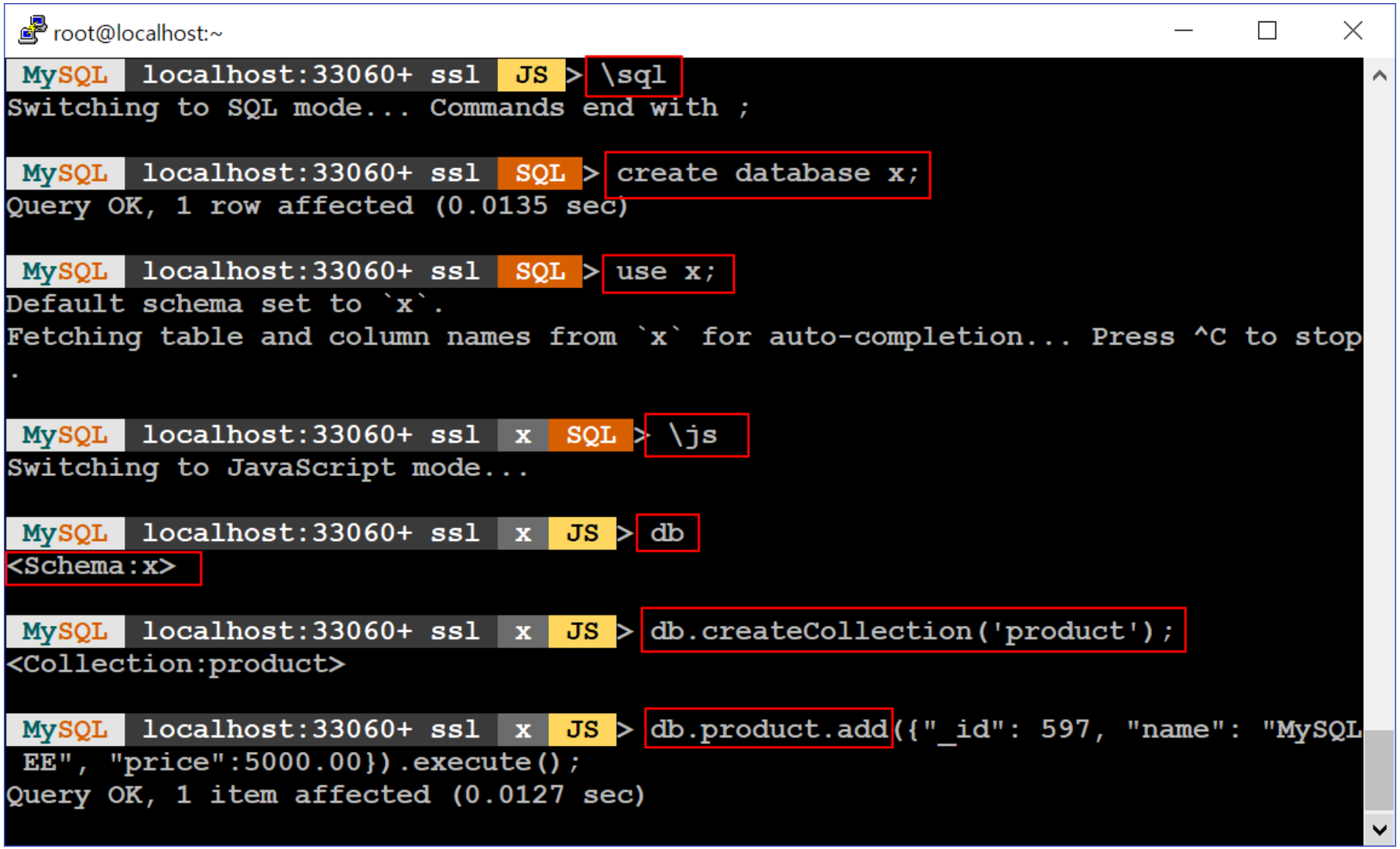
- VM : 64-bit Oracle Linux Server – 7.6(minimal installation)
- MySQL 8.0.16
- MySQL Shell 8.0.16

- X Protocol(Port:33060)이용하여 서버접속
  - Collection 생성
  - Document 추가
  - CRUD/SQL 실행
  - 트랜젝션 실행

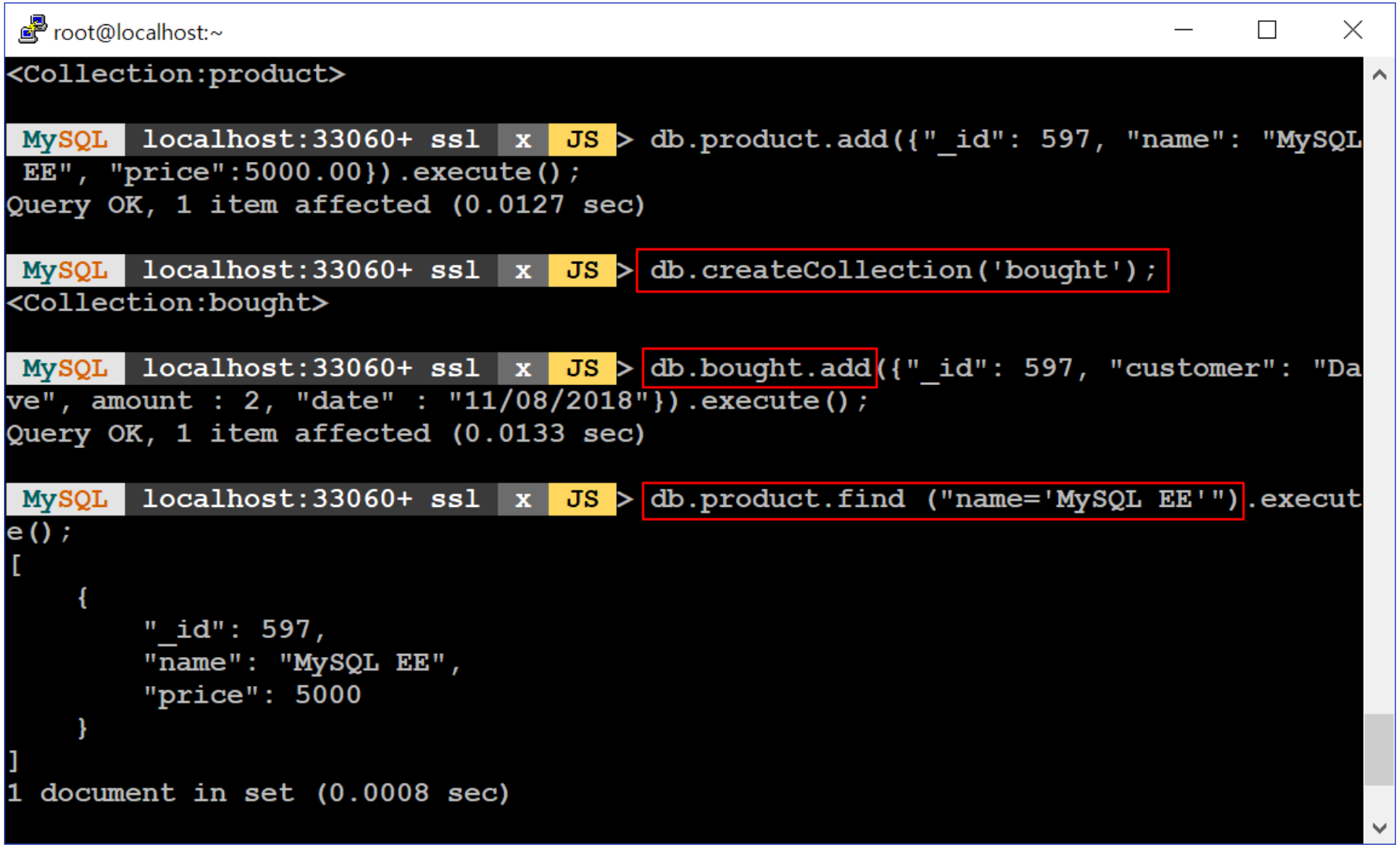
33060포트로 서버 접속



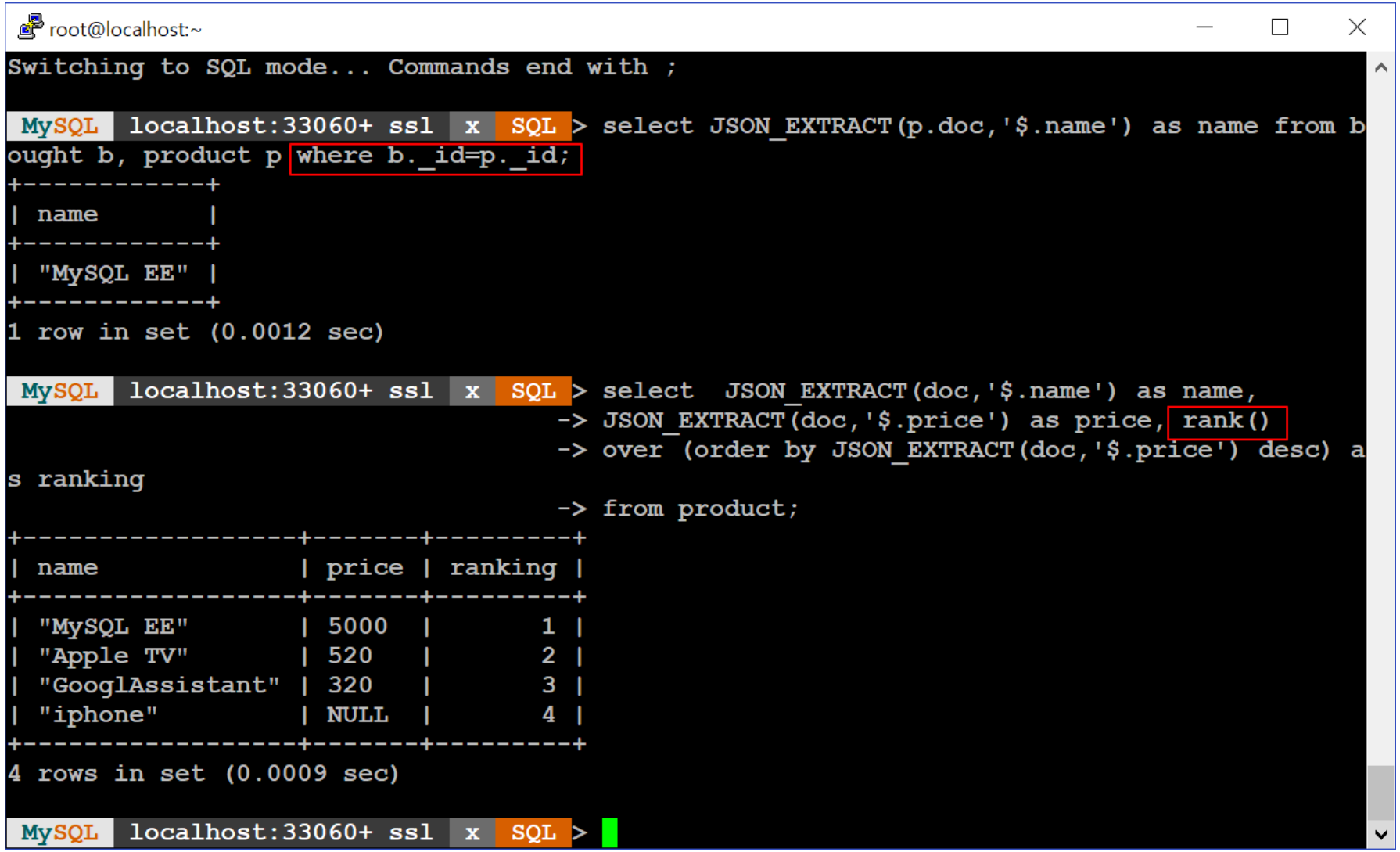
Collection생성, JSON도큐먼트 추가



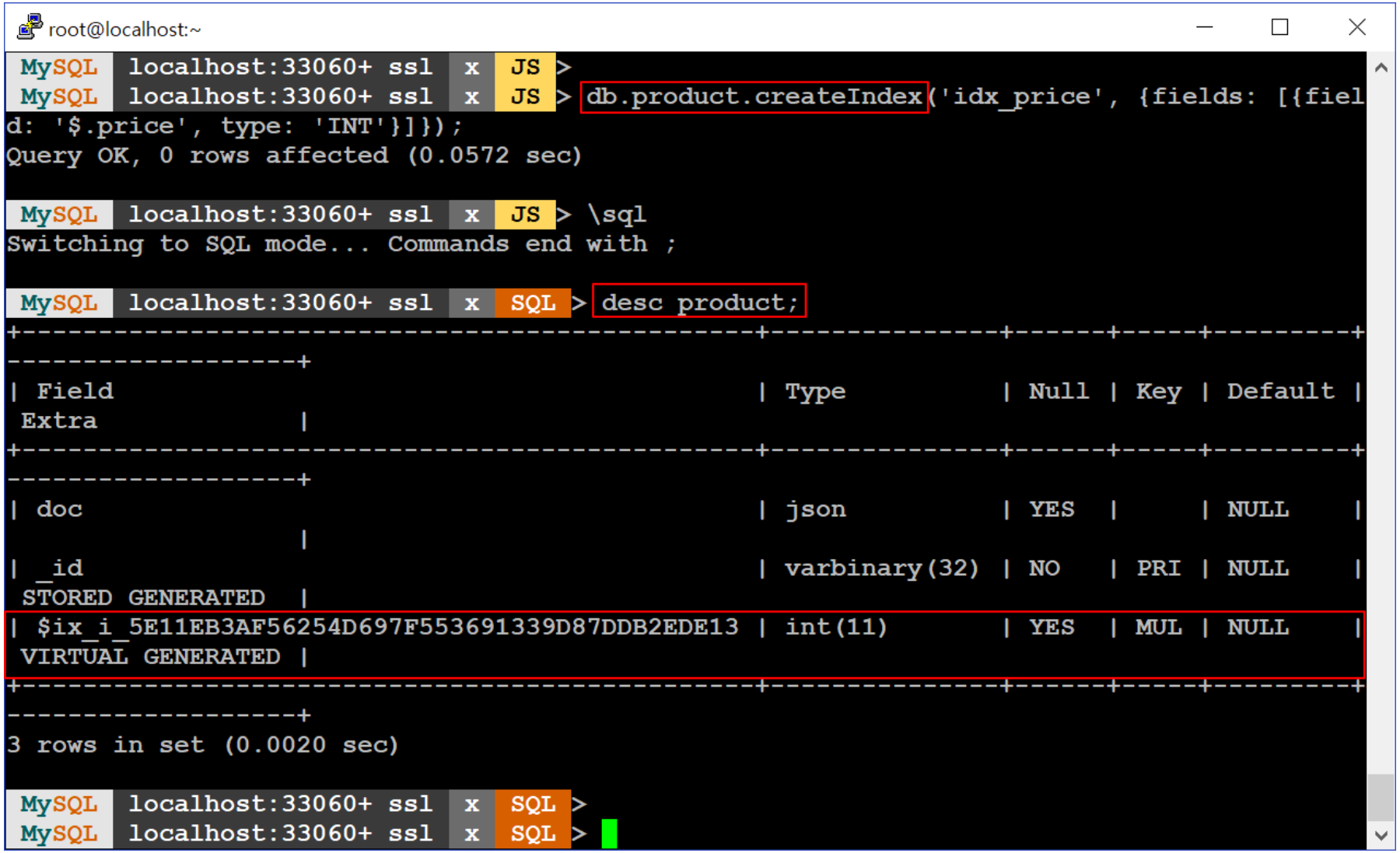
CRUD함수: add(),find()



Collection에 대해 JOIN, Window함수 이용



Collection에 대해 인덱스 추가



Collection에 대한 트랜젝션 지원