Nie samym ORMem żyje człowiek

Słów kilka o jOOQu

Kraków, 13.06.2019

Łukasz Krauzowicz

O mnie

Łukasz Krauzowicz

- Software Developer @ Ailleron
- kraluk@gmail.com
- github.com/kraluk
- @lkrauzo
- 2+ lata z jOOQiem

Agenda

Agenda

- jOOQ to…
- Generator
- Active Record
- `SELECT` / `INSERT` / `UPDATE` / `DELETE`
- Funkcje agregujące
- Procedury składowane
- Lessons learned

Ale mamy ORMy

```
public interface CustomerRepository extends CrudRepository<Customer, Long> {

   List<Customer> findByLastName(String name);

}
```

```java
@RepositoryRestResource(collectionResourceRel = "customers", path = "customers")
public interface CustomerRepository extends PagingAndSortingRepository<Customer, Long>{

    List<Customer> findByLastName(@Param("name") String name);

}
```

Czyli po co ten jOOQ?

Kiedy go używać?

ORM? JPA? HIBERNATE?
JA TAM WOLĘ MIEĆ KONTROLĘ NAD SWOIMI SQLAMI

“To zależy.”

–Sławomir Sobótka

`jOOQ` **vs** `JDBC API`

- bezpieczeństwo **typowania**
- bezpieczeństwo **syntax**’u (błędy kompilacji!)
- brak mapowania parametrów po **indeksie** w zapytaniu*
- brak **konkatenacji** literałów tekstowych
- przykrycie “**ciężkiego**” API JDBC (wyjątki + “resource’y”)
- ukrycie “**stanowości**” i **braku obiektowości** API JDBC

jOOQ

Disclaimer

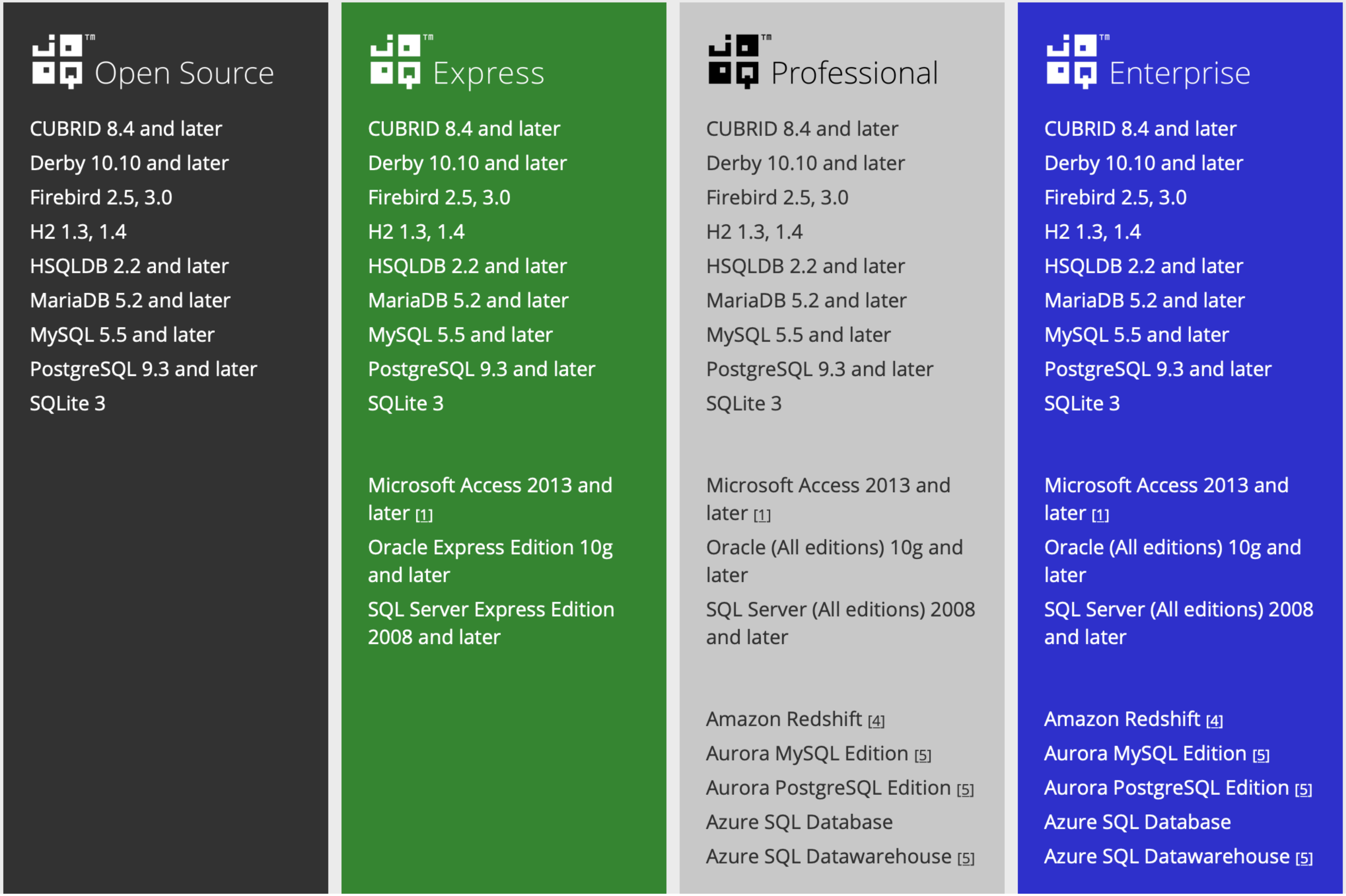
jOOQ

- nie jest ORMem
- nie jest rozwiązaniem wszystkich problemów
- nie jest idealny
- nie jest dla wszystkich :)
- jest inny

jOOQ

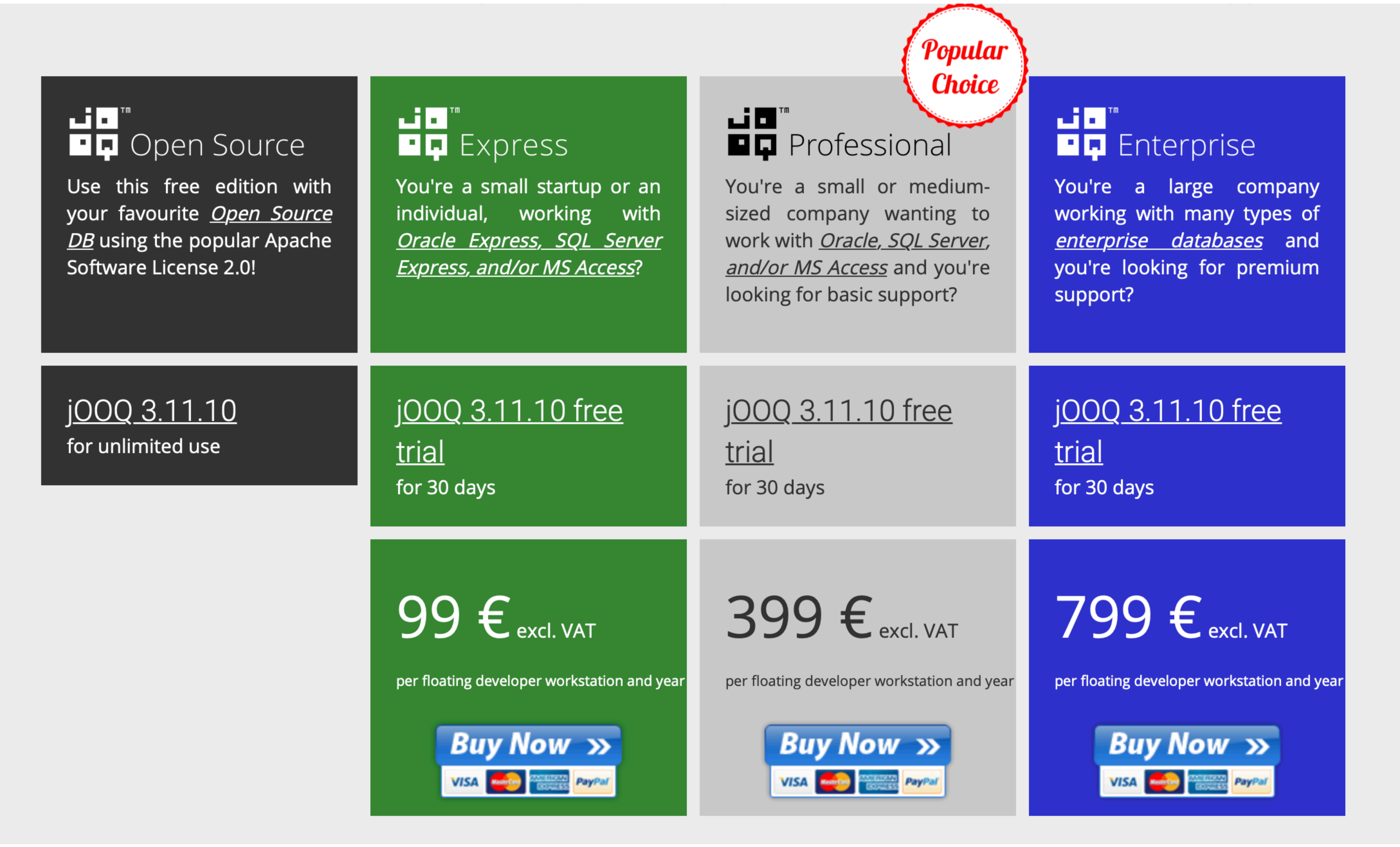
- jooq.org
- jooq.org/learn
- blog.jooq.org
- Java 6+ (commercial) / 8+ (open source)
- **`1.0.0`** - 14.08.2010
- **`3.11.11`** - 09.04.2019
- **Lukas Eder** - @lukaseder

Licencja



Open Source	Express	Professional	Enterprise
CUBRID 8.4 and later	CUBRID 8.4 and later	CUBRID 8.4 and later	CUBRID 8.4 and later
Derby 10.10 and later	Derby 10.10 and later	Derby 10.10 and later	Derby 10.10 and later
Firebird 2.5, 3.0	Firebird 2.5, 3.0	Firebird 2.5, 3.0	Firebird 2.5, 3.0
H2 1.3, 1.4	H2 1.3, 1.4	H2 1.3, 1.4	H2 1.3, 1.4
HSQLDB 2.2 and later	HSQLDB 2.2 and later	HSQLDB 2.2 and later	HSQLDB 2.2 and later
MariaDB 5.2 and later	MariaDB 5.2 and later	MariaDB 5.2 and later	MariaDB 5.2 and later
MySQL 5.5 and later	MySQL 5.5 and later	MySQL 5.5 and later	MySQL 5.5 and later
PostgreSQL 9.3 and later	PostgreSQL 9.3 and later	PostgreSQL 9.3 and later	PostgreSQL 9.3 and later
SQLite 3	SQLite 3	SQLite 3	SQLite 3
	Microsoft Access 2013 and later [1]	Microsoft Access 2013 and later [1]	Microsoft Access 2013 and later [1]
	Oracle Express Edition 10g and later	Oracle (All editions) 10g and later	Oracle (All editions) 10g and later
	SQL Server Express Edition 2008 and later	SQL Server (All editions) 2008 and later	SQL Server (All editions) 2008 and later
		Amazon Redshift [4]	Amazon Redshift [4]
		Aurora MySQL Edition [5]	Aurora MySQL Edition [5]
		Aurora PostgreSQL Edition [5]	Aurora PostgreSQL Edition [5]
		Azure SQL Database	Azure SQL Database
		Azure SQL Datawarehouse [5]	Azure SQL Datawarehouse [5]

http://www.jooq.org/legal/licensing#databases (11.06.2019)



Popular Choice

jOOQ Open Source	jOOQ Express	jOOQ Professional	jOOQ Enterprise
Use this free edition with your favourite *Open Source DB* using the popular Apache Software License 2.0!	You're a small startup or an individual, working with *Oracle Express, SQL Server Express, and/or MS Access*?	You're a small or medium-sized company wanting to work with *Oracle, SQL Server, and/or MS Access* and you're looking for basic support?	You're a large company working with many types of *enterprise databases* and you're looking for premium support?
jOOQ 3.11.10 for unlimited use	jOOQ 3.11.10 free trial for 30 days	jOOQ 3.11.10 free trial for 30 days	jOOQ 3.11.10 free trial for 30 days
	99 € excl. VAT per floating developer workstation and year	399 € excl. VAT per floating developer workstation and year	799 € excl. VAT per floating developer workstation and year
	Buy Now » VISA MasterCard AMERICAN EXPRESS PayPal	Buy Now » VISA MasterCard AMERICAN EXPRESS PayPal	Buy Now » VISA MasterCard AMERICAN EXPRESS PayPal

http://www.jooq.org/download/ (11.04.2019)

`jOOQ` Generator

`jOOQ` Generator

- jooq.org/doc/3.11/manual/code-generation
- **Standalone** (oficjalny)
- **Maven** (oficjalny)
- **Gradle** (github.com/etiennestuder/gradle-jooq-plugin)
- jooq.org/xsd/jooq-codegen-3.11.0.xsd

Gradle plugin

```
plugins {
    id "nu.studer.jooq" version "3.0.3"
}
```

Gradle plugin

```
jooq {
    version = "$jooqVersion"
    edition = "OSS"

    sakila(sourceSets.main) {
        String initSchema = "$projectDir/src/main/sql/sakila-ddl.sql".replaceAll("\\\\", "/")

        jdbc {
            driver = "org.h2.Driver"
            url = "jdbc:h2:mem:jooqplayground;INIT=RUNSCRIPT FROM '$initSchema'"
        }
        generator {
            name = "org.jooq.codegen.DefaultGenerator"
            database {
                name = "org.jooq.meta.h2.H2Database"
                includes = ".*"
                excludes = "information_schema"
            }
            generate {
                relations = true
                deprecated = false
                records = true
                immutablePojos = true
                fluentSetters = true
                daos = false
            }
            target {
                packageName = "io.kraluk.playground.jooq.db"
            }
        }}}
```

Maven plugin

```xml
<dependency>
    <groupId>org.jooq</groupId>
    <artifactId>jooq-codegen-maven</artifactId>
    <version>3.11.11</version>
</dependency>
```

`org.jooq.DSLContext`

“`org.jooq.DSLContext` - the one to rule them all.”

–Losowy użytkownik jOOQa

DSLContext

```
DSLContext dslContext(
    final javax.sql.DataSource dataSource) {

    return DSL.using(
        dataSource,
        SQLDialect.H2);
}
```

CRUD

Active (Updatable) Record

Dygresja

Active Record Pattern

```
part = new Part()

part.name = "Sample part"

part.price = 123.45

part.save()
```

```
INSERT INTO
parts (name, price)
VALUES
('Sample part', 123.45);
```

Rekord <—> wiersz w tabeli bazy danych

Obiekt **Active Record** posiada zestaw metod CRUD **—>** dostęp do bazy danych

```java
public interface UpdatableRecord<R extends UpdatableRecord<R>>
extends TableRecord<R> {

   int store() throws DataAccessException, DataChangedException;

   int insert() throws DataAccessException;

   int update() throws DataAccessException, DataChangedException;

   int delete() throws DataAccessException, DataChangedException;

   void refresh() throws DataAccessException;

}
```

INSERT

```java
public void save(final Film film) {
    context
        .insertInto(FILM)
        .set(FILM.TITLE, film.getTitle())
        .set(FILM.DESCRIPTION, film.getDescription())
        .set(FILM.RELEASE_YEAR, film.getReleaseYear())
        .set(FILM.LANGUAGE_ID, film.getLanguageId())
        .set(FILM.ORIGINAL_LANGUAGE_ID, film.getOriginalLanguageId())
        .set(FILM.RENTAL_DURATION, film.getRentalDuration())
        .set(FILM.RENTAL_RATE, film.getRentalRate())
        .set(FILM.LENGTH, film.getLength())
        .set(FILM.REPLACEMENT_COST, film.getReplacementCost())
        .set(FILM.RATING, film.getRating())
        .set(FILM.SPECIAL_FEATURES, film.getSpecialFeatures())
        .execute();
}
```

```java
public void save(final FilmRecord record) {
    record.store();
    record.refresh();
}
```

UPDATE

```java
public void update(final Film film) {
    context
        .update(FILM)
        .set(FILM.TITLE, film.getTitle())
        .set(FILM.DESCRIPTION, film.getDescription())
        .set(FILM.RELEASE_YEAR, film.getReleaseYear())
        .set(FILM.LANGUAGE_ID, film.getLanguageId())
        .set(FILM.ORIGINAL_LANGUAGE_ID, film.getOriginalLanguageId())
        .set(FILM.RENTAL_DURATION, film.getRentalDuration())
        .set(FILM.RENTAL_RATE, film.getRentalRate())
        .set(FILM.LENGTH, film.getLength())
        .set(FILM.REPLACEMENT_COST, film.getReplacementCost())
        .set(FILM.RATING, film.getRating())
        .set(FILM.SPECIAL_FEATURES, film.getSpecialFeatures())
        .where(
            FILM.FILM_ID.eq(film.getFilmId())
        )
        .execute();
}
```

```java
public void update(final FilmRecord record) {
    record.store();
    record.refresh();
}
```

DELETE

```
public int delete(final Integer id) {
    context
        .delete(FILM)
        .where(
            FILM.FILM_ID.eq(id)
        )
        .execute();
}
```

```java
public void delete(final FilmRecord record) {
    record.delete();
}
```

SELECT

```java
public Optional<Film> find(final Integer id) {
    return context
        .select()
        .from(FILM)
        .where(
            FILM.FILM_ID.eq(id)
        )
        .fetchOptionalInto(Film.class);
}
```

```java
public Optional<FilmRecord> find(final Integer id)
{
    return context
        .selectFrom(FILM)
        .where(
            FILM.FILM_ID.eq(id)
        ).fetchOptional();
}
```

FUNKCJE AGREGUJĄCE

`org.jooq.impl.DSL`

i jego rozszerzenia

- org.jooq.util.cubrid.**CUBRID**DSL
- org.jooq.util.derby.**Derby**DSL
- org.jooq.util.cubrid.firebird.**Firebird**DSL
- org.jooq.util.h2.**H2**DSL*****
- org.jooq.util.hsqldb.**HSQLDB**DSL*****
- org.jooq.util.mariadb.**MariaDB**DSL
- org.jooq.util.mysql.**MySQL**DSL
- org.jooq.util.postgres.**Postres**DSL
- org.jooq.util.sqllte.**SQLite**DSL

PROCEDURY SKŁADOWANE

org.jooq.impl.**AbstractRoutine**<**T**>

class **Routines**

FETCHING

jooq.org/doc/3.11/manual/sql-execution/fetching/

LESSONS LEARNED

czyli tego nie próbuj*

***)** moim zdaniem

Lessons Learned

- **Formatowanie kodu!**
- Klątwa `if-else`
- Fluent API jest dobre
- Duża pokusa mieszania odpowiedzialności
- Transakcje same się nie obsłużą*
- **Formatowanie kodu!**

***)** Ale jest: `group: 'org.springframework.boot', name: 'spring-boot-starter-jooq'`

Klątwa `if-else`

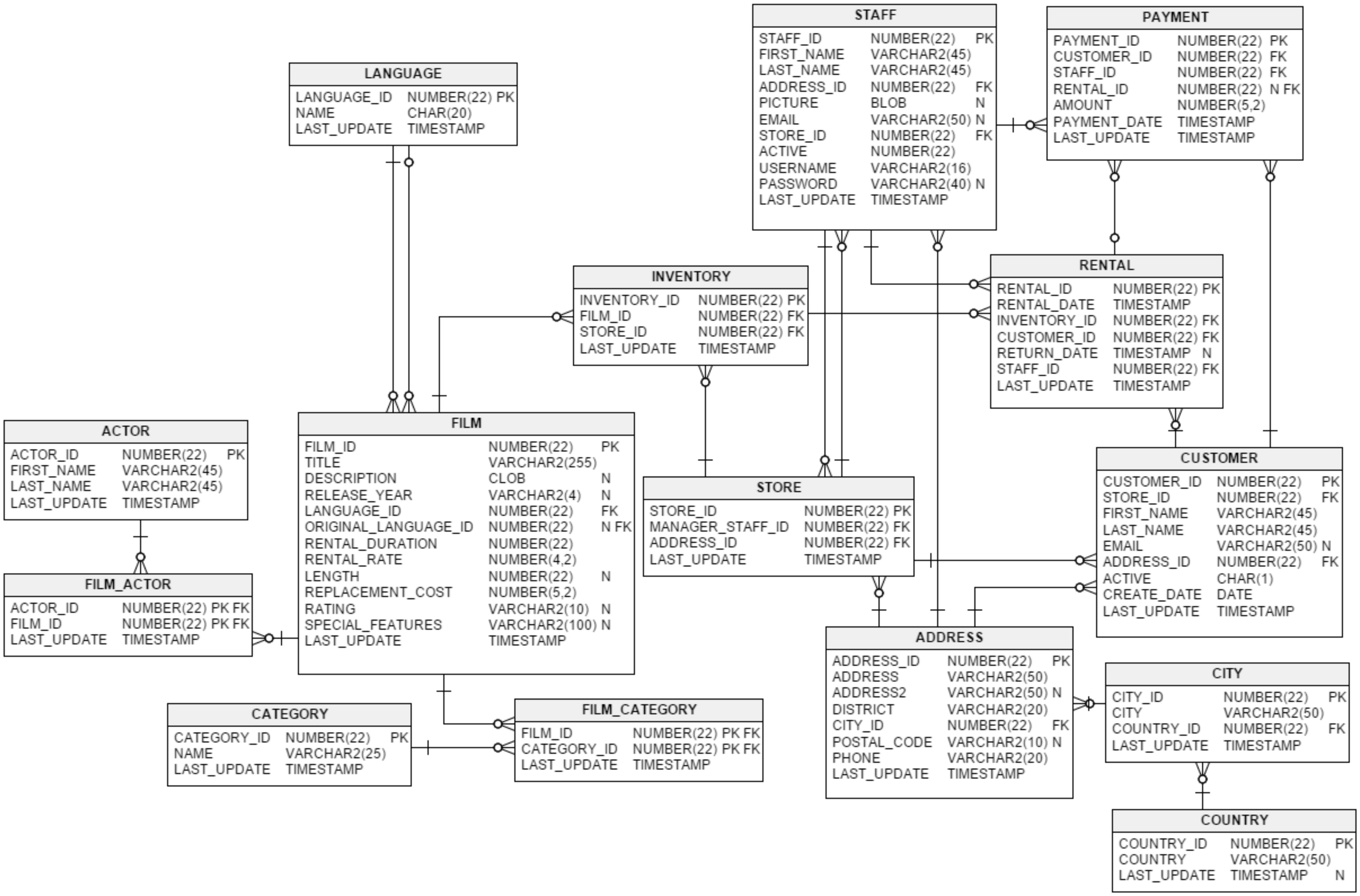
```
List<Something> findSomethingBySomething(SearchCriteria criteria,
          List<RestrictionDefinition> definitions) {

     List<Type> types = SearchUtils.getTypes(
               criteria, Restriction.TYPES, definitions);

     DSLContext request = JOOQUtil.getDSLContext();
     SelectQuery<?> query = request.selectQuery();

     query.addSelect(SOMETHING.ID,
           SOMETHING.DATE_1,
           SOMETHING.DATE_2,
           SOMETHING.CREATED_AT,
           SOMETHING.UPDATED_AT,
           SOMETHING.NUMBER,
           SOMETHING.MAIN_ID,
           SOMETHING.KIND,
           SOMETHING.STATUS,
           SOMETHING.CODE,
           SOMETHING.CREATED_BY);
     query.addFrom(SOMETHING);

     query.addConditions(SOMETHING.IMPORTANT_ID.equal(IdCtx.getId()));
query.addConditions(Operator.AND, getRights(types));

if (criteria.hasRestriction(Restriction.STATUS)) {
    query.addConditions(SOMETHING.STATUS
    .in(criteria.getRestriction(Restriction.STATUS).getValue(List.class)));
}

if (criteria.hasRestriction(Restriction.START_DATE)) {
    LocalDateTime date =
          convert(criteria.getRestriction(Restriction.DATE_1).getValue());

   query.addConditions(SOMETHING.DATE_1.greaterOrEqual(date));
}

if (criteria.hasRestriction(Restriction.END_DATE)) {
    LocalDateTime date =
          convert(criteria.getRestriction(Restriction.DATE_2).getValue());

   query.addConditions(SOMETHING.DATE_2.lessOrEqual(date));
}

if (criteria.hasRestriction(Restriction.CREATED_AT_START_DATE)) {
    LocalDateTime date =
           convert(
               criteria.getRestriction(
                      Restriction.CREATED_AT_START_DATE).getValue());

    query.addConditions(SOMETHING.CREATED_AT.greaterOrEqual(date));
}
```

```
if (criteria.hasRestriction(Restriction.CREATED_AT_END_DATE)) {
    LocalDateTime date =
           convert(criteria.getRestriction(
                 Restriction.CREATED_AT_END_DATE).getValue());

    query.addConditions(SOMETHING.CREATED_AT.lessOrEqual(date));
}

if (criteria.hasRestriction(Restriction.VERSION_START_DATE)) {
    LocalDateTime date =
           convert(criteria.getRestriction(
                 Restriction.VERSION_START_DATE).getValue());

    query.addConditions(SOMETHING.UPDATED_AT.greaterOrEqual(date));
}

if (criteria.hasRestriction(Restriction.VERSION_END_DATE)) {
    LocalDateTime date =
           convert(criteria.getRestriction(
                 Restriction.VERSION_END_DATE).getValue());
    query.addConditions(SOMETHING.UPDATED_AT.lessOrEqual(startDate));
}

if (criteria.hasRestriction(Restriction.PARTNER_ID)) {
    Integer id = criteria.getRestriction(
                 Restriction.ARTNER_ID).getIntegerValue();
    query.addConditions(SOMETHING.MAIN_ID.equal(id));
}
query.addConditions(createSomeOtherImportantConditions());
```

```
    PreviousIdPagination pagination = (PreviousIdPagination)
            criteria.getPagination().get();

    **handlePagination(query, pagination, criteria.getOrdering());**

    String sql = query.getSQL(ParamType.INLINED);
    return QueryExecutor.execute(sql)
            .stream()
            .map(this::convertToSomething)
            .collect(toList());
 }

 private void handlePagination(SelectQuery query, PreviousIdPagination
         pagination, List<Order> ordering) {

    List<String> orderings = new ArrayList<>();
    orderings.add(Restriction.ORDER_DATE_1);
    orderings.add(Restriction.ORDER_ID);
    orderings.add(Restriction.ORDER_CREATED_ATDATE);
    orderings.add(Restriction.ORDER_VERSION);


    // some other magic constructions happens here…
}
```

Materiały

Sakila DB



jooq.org/sakila

github.com/kraluk/jooq-introduction

Pytania?

Dzięki za uwagę!