JPA

jako obiektowy dostęp do relacyjnych bazy danych

O MNIE

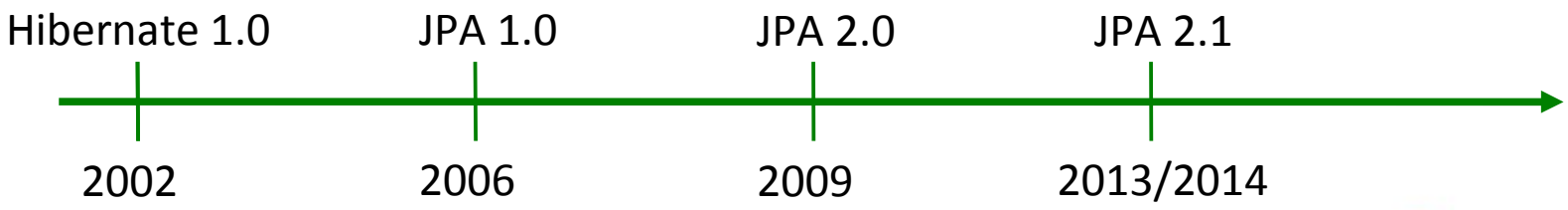
- Mirosław Szajowski
- Funkcja: Projektant/Architekt
- 8 lat stażu w IT
- Aplikacje webowe
- Aplikacje desktopowe
- Aplikacje mobilne

Agenda

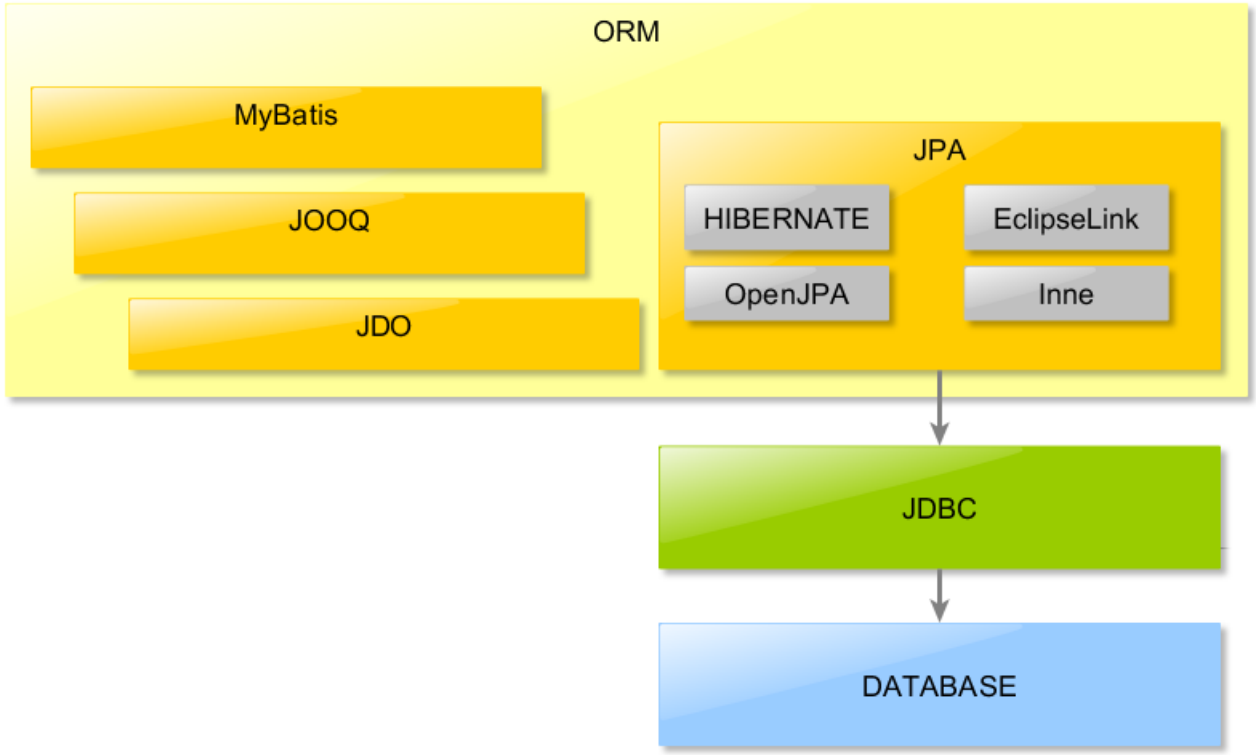
- Czym jest JPA
- Zalety JPA:
  - Produktywność/wygoda pracy
  - Wydajność
  - Bezpieczeństwo
  - Niezależność

Czym jest JPA

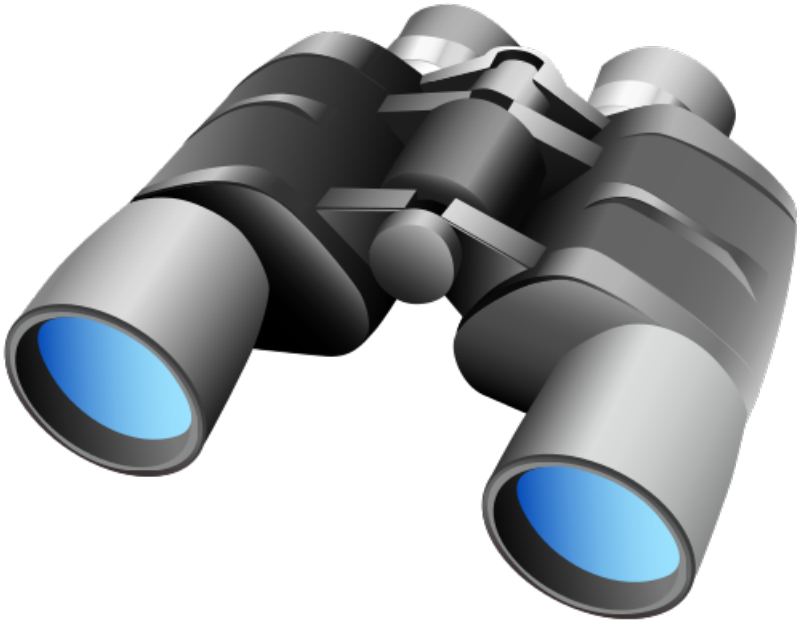
- Standard w świecie **J**AVA **E**nterprise **E**dition
- Umożliwia pracę z SQL bazą danych bez pisania SQL
- Większość projektów w JAVA korzysta z JPA*



JPA , JDBC, ORM, HIbernate...



JPA z bliska



Produktywność/wygoda pracy

JPA

INDEX_NR	FIRST_NAME	LAST_NAME	EMAIL	PASSWORD
1	JAN	NOWAK	jan.n@gmail.com	234324fdsfsd
2	MARCIN	KOWALSKI	marin.k@gmail.com	324324fdgdfg

ID	***INDEX_NR_STUDENT***	SUBJECT	VALUE
1	1	PROGRAMOWANIE	5
2	1	BAZY DANYCH	4,5
3	2	BAZY DANYCH	4

JPA

INDEX_NR	FIRST_NAME	LAST_NAME	EMAIL	PASSWORD
1	JAN	NOWAK	jan.n@gmail.com	234324fdsfsd
2	MARCIN	KOWALSKI	marin.k@gmail.com	324324fdgdfg

ID	*INDEX_NR_STUDENT*	SUBJECT	VALUE
1	1	PROGRAMOWANIE	5
2	1	BAZY DANYCH	4,5
3	2	BAZY DANYCH	4

```
//Wyszukaj studenta po indeksie
SELECT *
FROM STUDENT s
WHERE s.INDEX_NR = 1
```

```
//Pobierz oceny studenta
SELECT *
FROM MARK m
WHERE m.INDEX_NR_STUDENT = 1
```

JPA

JPA Umożliwia pracę z SQL

bazą danych bez pisania SQL

```
//Wyszukaj studenta po indeksie
Student student = JPA.find(Student.class, 1);
String email = student.getEmail();


//Pobierz oceny studenta
Collection<Mark> marks = student.getMarks();
```

JPA

JPA Umożliwia pracę z SQL

bazą danych bez pisania SQL

```
//Wyszukaj studenta po indeksie
Student student = JPA.find(Student.class, 1);
String email = student.getEmail();
```

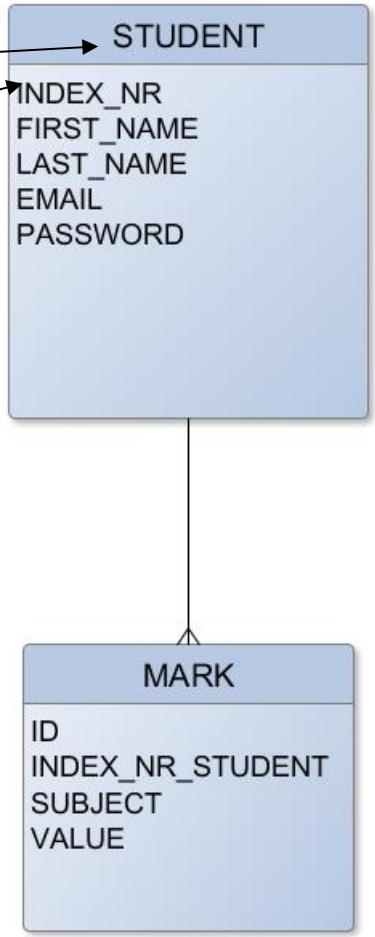
```
SELECT *
FROM STUDENT s
WHERE s.INDEX_NR = 1
```


```
//Pobierz oceny studenta
Collection<Mark> marks = student.getMarks();
```

```
SELECT *
FROM MARK
WHERE INDEX_NR_STUDENT = 1
```

JPA SQL bez SQL jak to możliwe

```java
@Entity
@Table(name="STUDENT")
public class Student {

    @Id
    @Column(name="INDEX_NR")
    Long index;

    @Column(name="EMAIL")
    String email;

    @Column(name="FIRST_NAME")
    String firstName;

    @Column(name="LAST_NAME")
    String lastName;

    @Column(name="PASSWORD")
    String password;

    @OneToMany(fetch=FetchType.LAZY)
    @JoinColumn(name="INDEX_NR_STUDENT")
    Collection<Mark> marks;

}
```



JPA

```
//Wyszukaj studenta po indeksie
Student student = JPA.find(Student.class, 1);
String email = student.getEmail();
```

```
SELECT *
FROM STUDENT s
WHERE s.INDEX_NR = 1
```

```
//Pobierz oceny studenta
Collection<Mark> marks = student.getMarks();
```

```
SELECT *
FROM MARK
WHERE INDEX_NR_STUDENT = 1
```

JPA – dodaj/popraw/usuń studenta

```
//Odpowiwdnik INSERT INTO

Student student = new Student();
student.setEmail("nowak@gmail.com");
student.setFirstName("Jan");
student.setLastName("Nowak");
student.setPassword("PassowrdHash");

JPA.persist(student);
```

```
//Odpowiwdnik UPDATE

Student student = JPA.find(Student.class, 1);
student.setPassword("NewPasswordHash");

JPA.persist(student);
```

```
//Odpowiwdnik DELETE

Student student = JPA.find(Student.class, 1);

JPA.delete(student);
```

JPA – zapytania

```
//JPA Criteria
CriteriaBuilder cb = JPA.getCriteriaBuilder();
CriteriaQuery<Student> cq = cb.createQuery(Student.class);

Root<Student> student = cq.from(Student.class);
cq.select(student);
cq.where(student.get(Student_.index).eq(1));

TypeQuery<Student> q = JPA.createQuery(cq);
Student student = q.getUniqueResult();
```

```
//Hibernate Criteria
Student student = Hibernate
.createCriteria(Student.class)
.add(Restrictions.eq("index", 1))
.uniqueResult()
```

```
//JPQL/HQL
FROM Student WHERE index = 1
```

```
SELECT *
FROM STUDENT s
WHERE s.INDEX_NR = 1
```

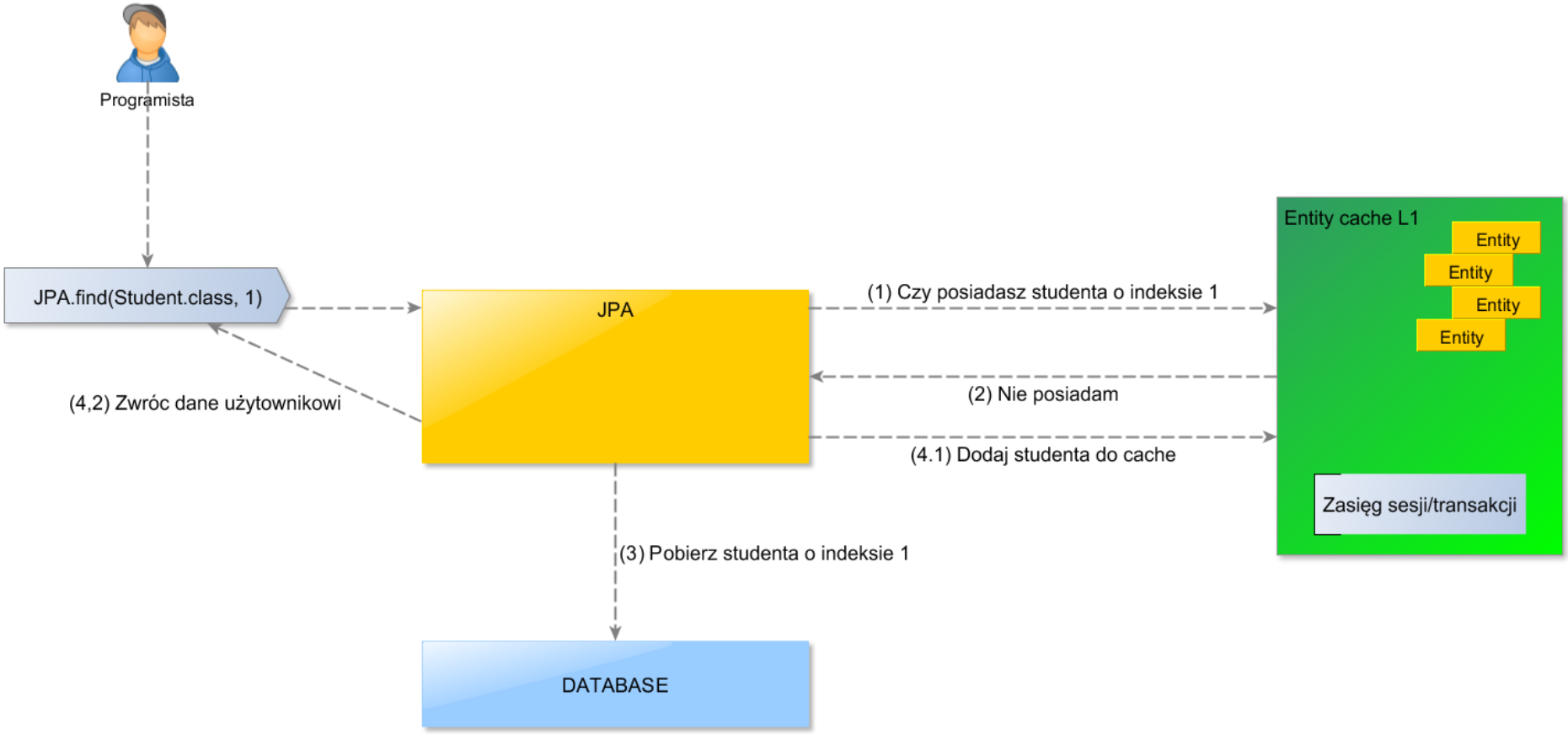
JPA - wydajność

JPA - wydajność

```
Student student = JPA.find(Student.class, 1);
Student student = JPA.find(Student.class, 1);
Student student = JPA.find(Student.class, 1);
```

Ile zostanie wygenerowanych SQL?

JPA - Cache

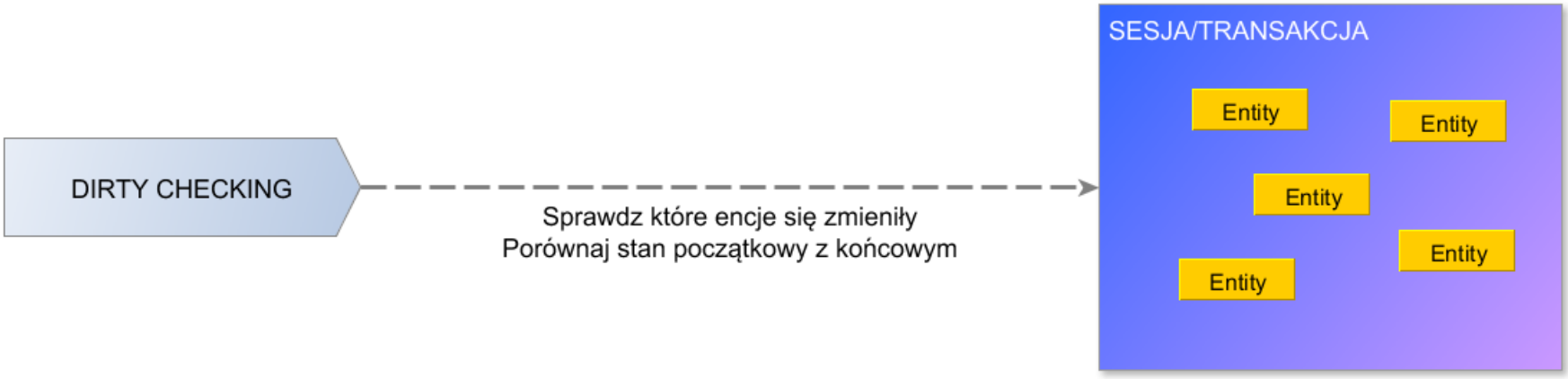

Programista
JPA.find(Student.class, 1)
JPA
(1) Czy posiadasz studenta o indeksie 1
(2) Nie posiadam
(4.1) Dodaj studenta do cache
(4,2) Zwróc dane użytownikowi
(3) Pobierz studenta o indeksie 1
DATABASE
Entity cache L1
Entity
Entity
Entity
Entity
Zasięg sesji/transakcji

JPA - wydajność

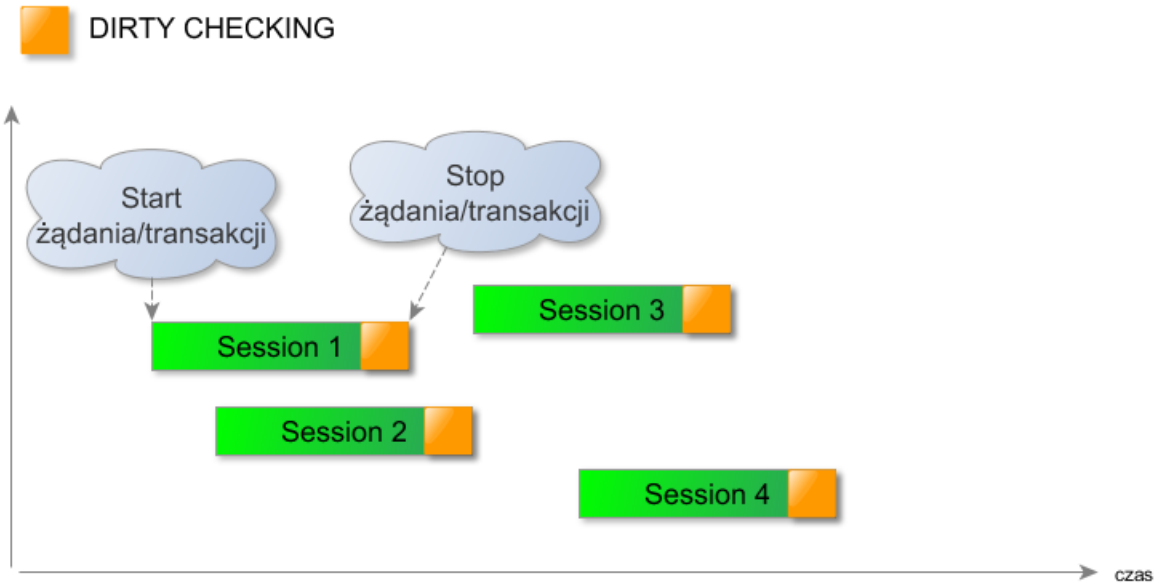
```
//Odpowiwdnik UPDATE

Student student = JPA.find(Student.class, 1);
student.setPassword("NewPasswordHash");

JPA.persist(student);

student.setPassword("NewPasswordHash2");
JPA.persist(student);

student.setPassword("NewPasswordHash3");
JPA.persist(student);
```

Ile zostanie wygenerowanych SQL?

DIRTY CHECKING



DIRTY CHECKING



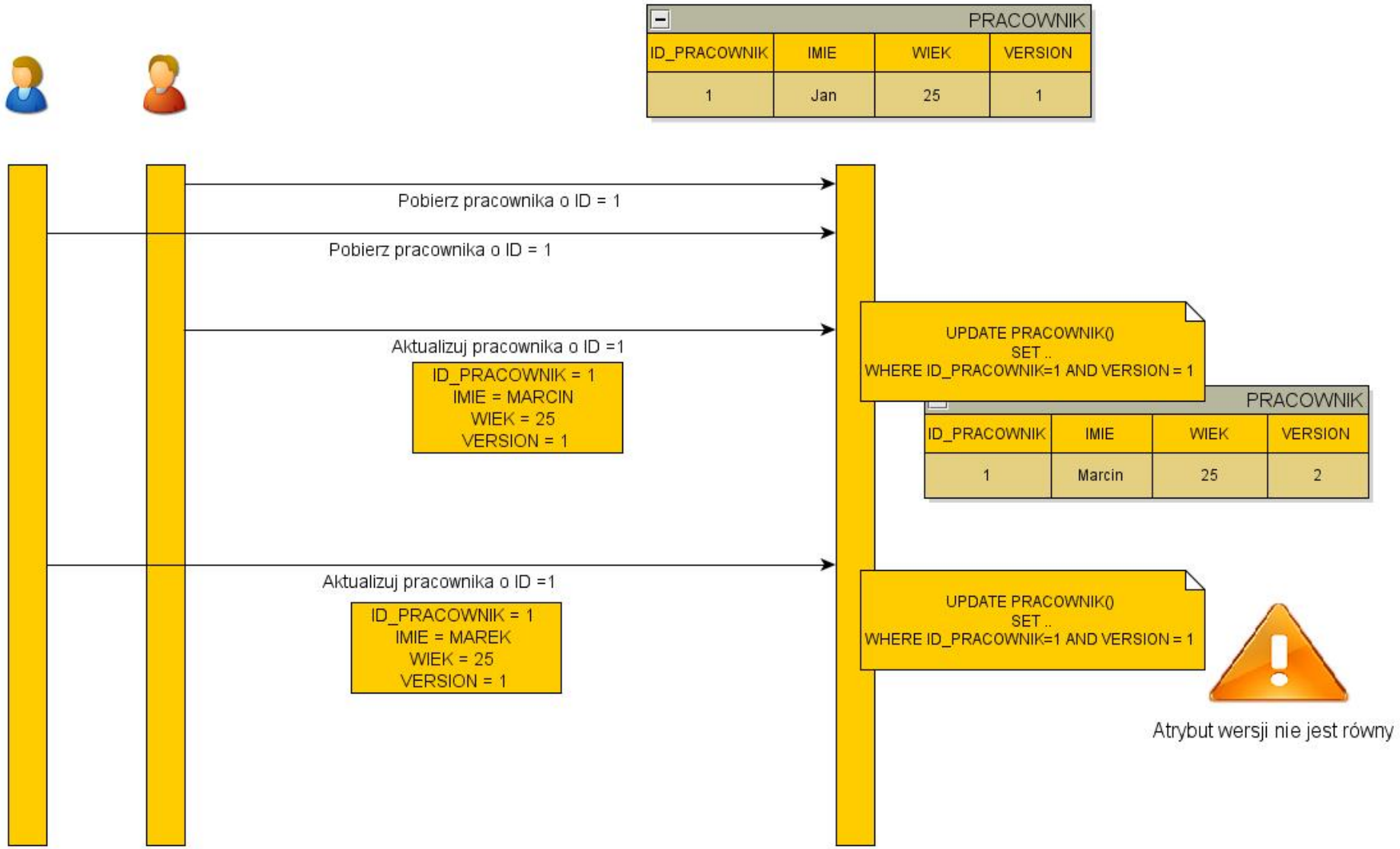
Krótkie blokady bazodanowe

Napełnione Batch'e

```
<property name="hibernate.jdbc.batch_size">50</property>
<property name="hibernate.order_inserts">true</property>
<property name="hibernate.order_updates">true</property>
```

Bezpieczeństwo

Wersjonowanie



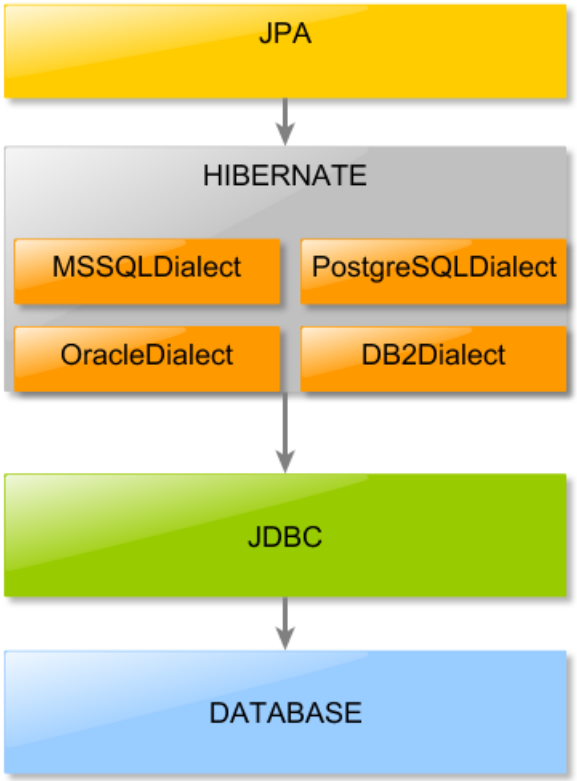
SQL Injection, named query

```
Query query = em.createQuery("from Student where email = : email");
query.setParameter("email", "j.kowalski@gmail.com");

Student student = query.getUniqueResult();
```

```
@Entity
@Table(name="STUDENT")
@NamedQuery(
    name="findByEmail",
    query="from Student where email = : email")
public class Student {


Query query = em.getNamedQuery("findByEmail");
```

Niezależność

JPA - Niezależność



- Dialect - org.hibernate.dialect
  - AbstractHANADialect - org.hibernate.dialect
  - AbstractTransactSQLDialect - org.hibernate.dialect
    - SQLServerDialect - org.hibernate.dialect
      - SQLServer2005Dialect - org.hibernate.dialect
    - SybaseDialect - org.hibernate.dialect
  - Cache71Dialect - org.hibernate.dialect
  - CUBRIDDialect - org.hibernate.dialect
  - DB2Dialect - org.hibernate.dialect
  - FrontBaseDialect - org.hibernate.dialect
  - H2Dialect - org.hibernate.dialect
  - HSQLDialect - org.hibernate.dialect
  - InformixDialect - org.hibernate.dialect
  - IngresDialect - org.hibernate.dialect
  - InterbaseDialect - org.hibernate.dialect
  - JDataStoreDialect - org.hibernate.dialect
  - MckoiDialect - org.hibernate.dialect
  - MimerSQLDialect - org.hibernate.dialect
  - MySQLDialect - org.hibernate.dialect
  - Oracle8iDialect - org.hibernate.dialect
  - Oracle9Dialect - org.hibernate.dialect
  - PointbaseDialect - org.hibernate.dialect
  - PostgreSQL81Dialect - org.hibernate.dialect
  - ProgressDialect - org.hibernate.dialect
  - RDMSOS2200Dialect - org.hibernate.dialect
  - SAPDBDialect - org.hibernate.dialect
  - TeradataDialect - org.hibernate.dialect
  - TimesTenDialect - org.hibernate.dialect

Podsumowanie

- JPA standard w świecie JAVA Enterprise
- Najczęściej wybierany mechanizm dostępu do bazy danych
- Zwiększa produktywność poprzez prace na obiektach
- Mechanizmy dirty checking i cache zwiększają wydajność
- Wbudowane mechanizmy wersjonowania
- Dialecty gwarantujące niezależność

Pytania

Dziękuje za uwagę