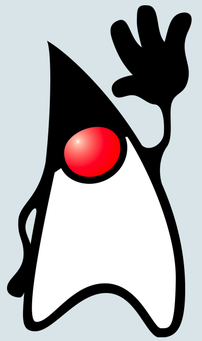
СБЕРБАНК

Корпоративный университет



Базы данных. Практическое применение

Занятие №10

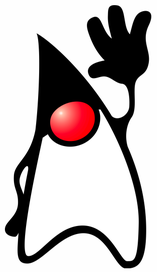
СБЕРБАНК
Корпоративный университет

Куценко Виталий

Что сможем после занятия

СБЕРБАНК
Корпоративный университет

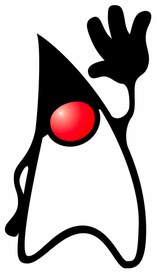
- Абстракции. ORM
- Hibernate ORM
- Написать своё приложение с БД



Договорённости

СБЕРБАНК
Корпоративный университет

- Активно участвуем. Не стесняйтесь задавать вопрос.
- Но off-topic обсуждаем в Telegram @sb_ku_java_2019_10
- Не стесняйтесь просто спрашивать в telegram.
- ДЗ - работаем над библиотекой



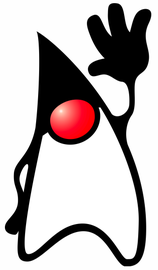
Договорились?
Поехали!

План занятия

- **Абстракции. ORM**

- Hibernate ORM

- Написать своё приложение с БД



01

Абстракции. ORM

CAP-теорема

Consistency, Availability и Partition Tolerance.
Можно выбрать только два из трех.

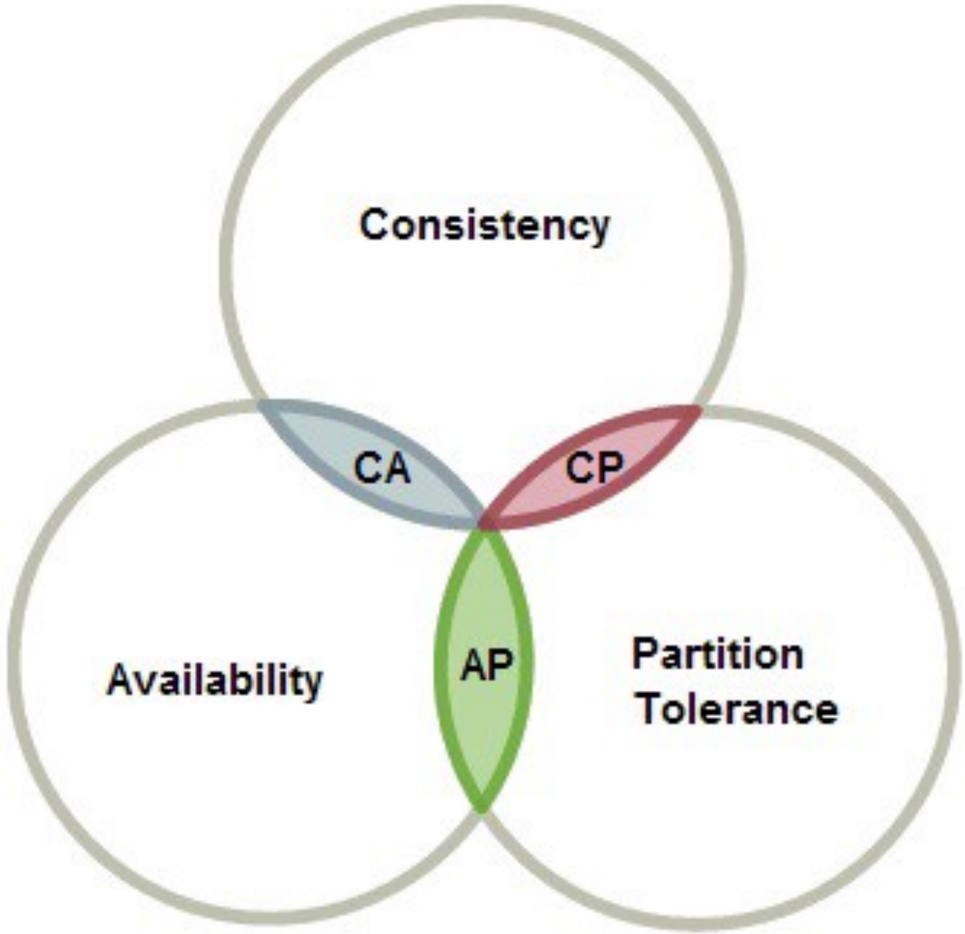
Partition Tolerance – система продолжает работать, несмотря на сетевые проблемы между узлами.

Mongo тут

Consistency – каждое чтение возвращает самую последнюю запись.

РСУБД тут

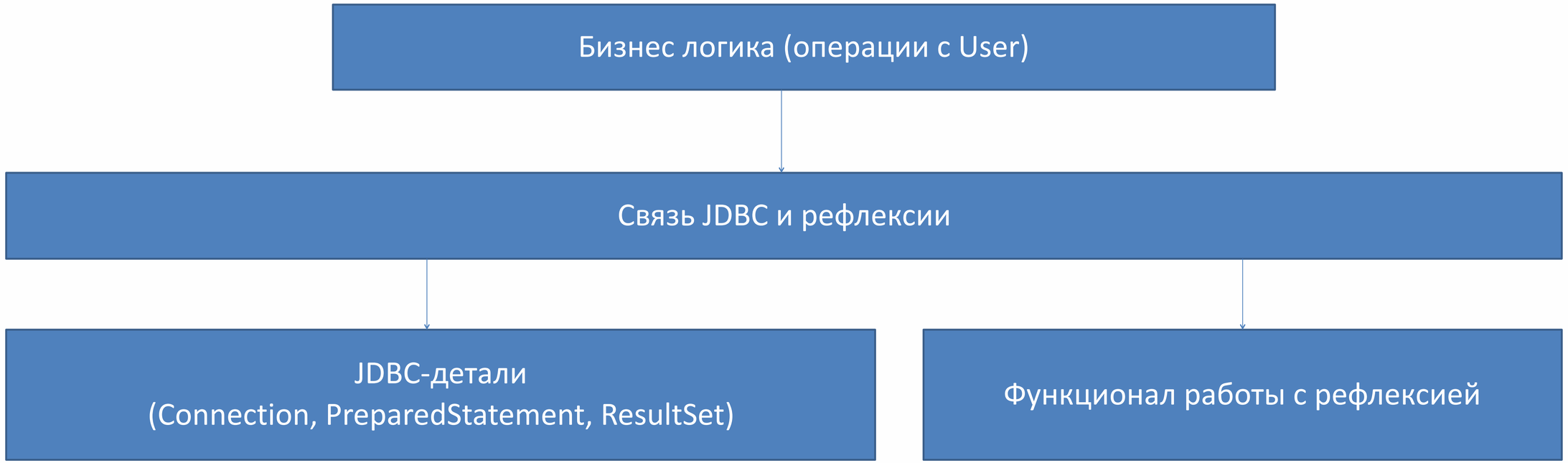
Availability – каждый запрос возвращает ответ без гарантий возврата последней записи.

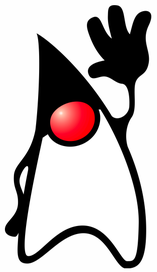


Абстракции приложений: ORM на рефлексии

СБЕРБАНК
Корпоративный университет

Работа с объектом User.





Закон протекания абстракций

СБЕРБАНК
Корпоративный университет

[The Law of Leaky Abstractions]

NOVEMBER 11, 2002 by JOEL SPOLSKY (CEO и сооснователь Stack Overflow)

Суть «протекания» - детали и проблемы нижнего уровня влияют на верхние.

Примеры из статьи.

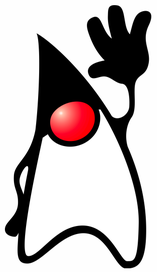
1. TCP/IP протокол. Проблемы нижнего слоя IP влияют на верхний слой TCP.
2. Производительность SQL-запросов часто зависит от порядка операндов.

Закон:

“All non-trivial abstractions, to some degree, are leaky”.

Интересная мысль:

“So the abstractions save us time working, but they don’t save us time learning”.



Блокировки данных

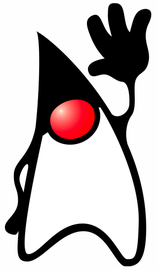
Подходы к организации блокировок:

–оптимистичный

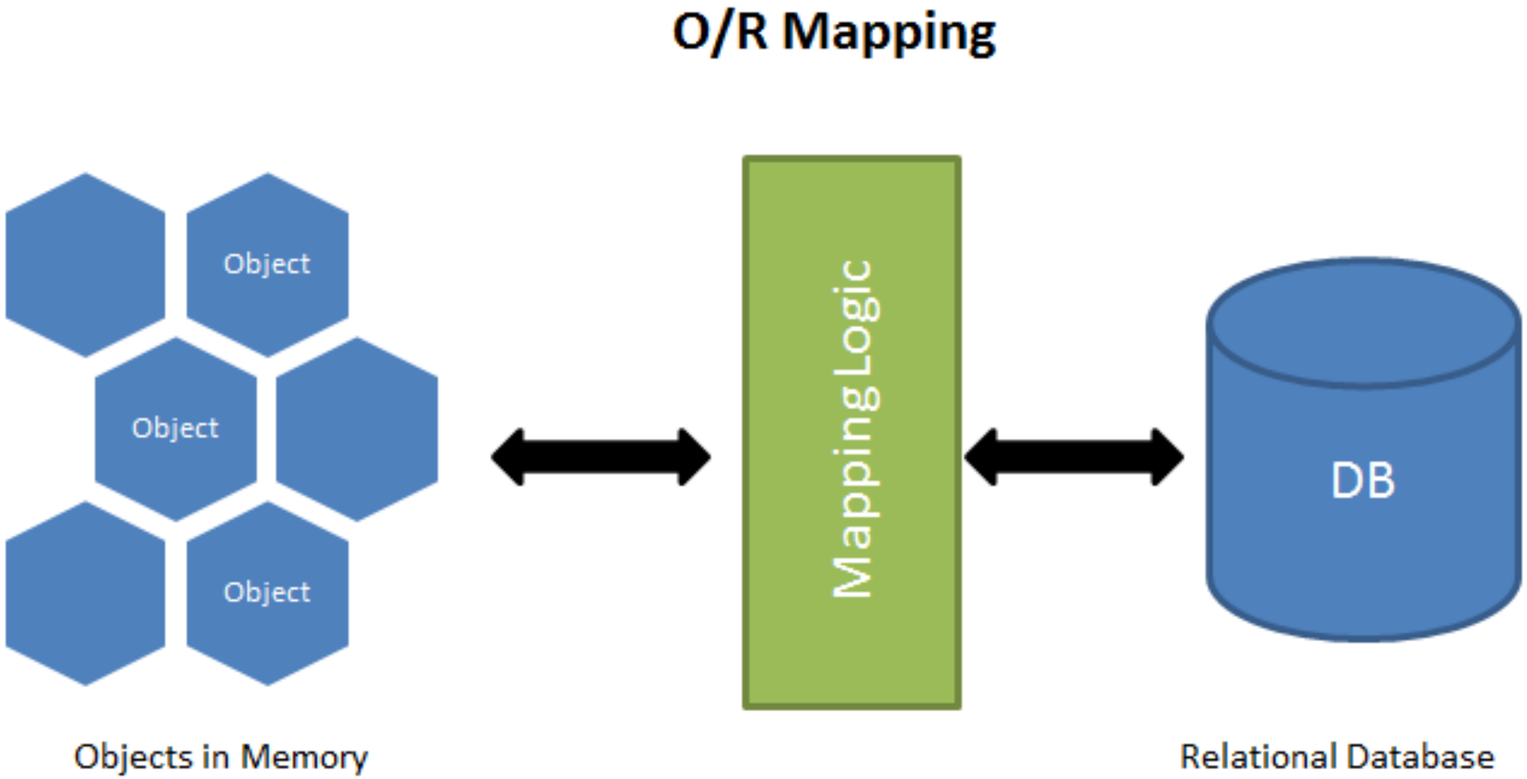
основан на «ревизиях», сначала меняем, потом проверяем

–пессимистичный

сначала блокируем, потом меняем.



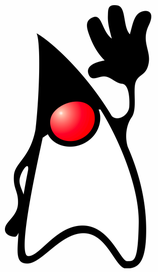
ORM (Object-relational mapping)



В терминах Java, нам надо:

1) Сформировать sql-запрос

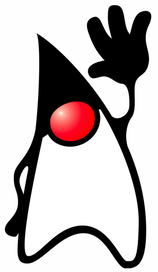
2) Результат запроса переложить в объект



ORM на рынке (что выбрать)

СБЕРБАНК
Корпоративный университет

- jdbc 💀
- Spring JDBC Template
- MyBatis
- Hybernate (JPA)
- Самодельный ORM ☠



Ваши вопросы?

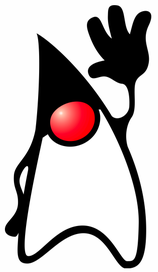
Если что – их можно задать потом

План

СБЕРБАНК
Корпоративный университет

- Абстракции. ORM

- **Hibernate ORM**
- Написать своё приложение с БД

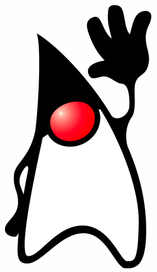


02

Hibernate ORM

БД

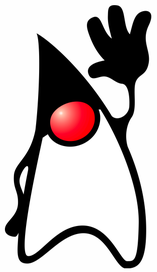
- В реляционных БД данные хранятся в таблицах
- Каждый кортеж суть строка
- Каждый атрибут кортежа - это столбец
- Между таблицами есть связи



В итоге

СБЕРБАНК
Корпоративный университет

- Класс сущности отображается на одну таблицу
- Поля класса в основном отображены на поля таблицы
- Если поле класса тоже сущность - зри в начало

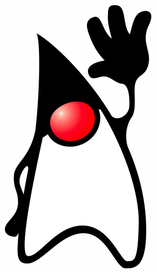


Impedance mismatch

СБЕРБАНК
Корпоративный университет

Impedance mismatch - это такое смешное слово для обозначения несоответствия парадигм

- Классы только кажутся таблицами, а объекты строчками
- Объекты – это, скорее, граф в математическом понимании

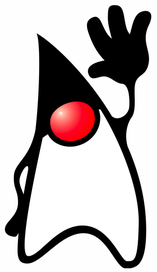


Impedance mismatch

СБЕРБАНК
Корпоративный университет

Чем будут поля в таблице?
Сколько таблиц и классов?

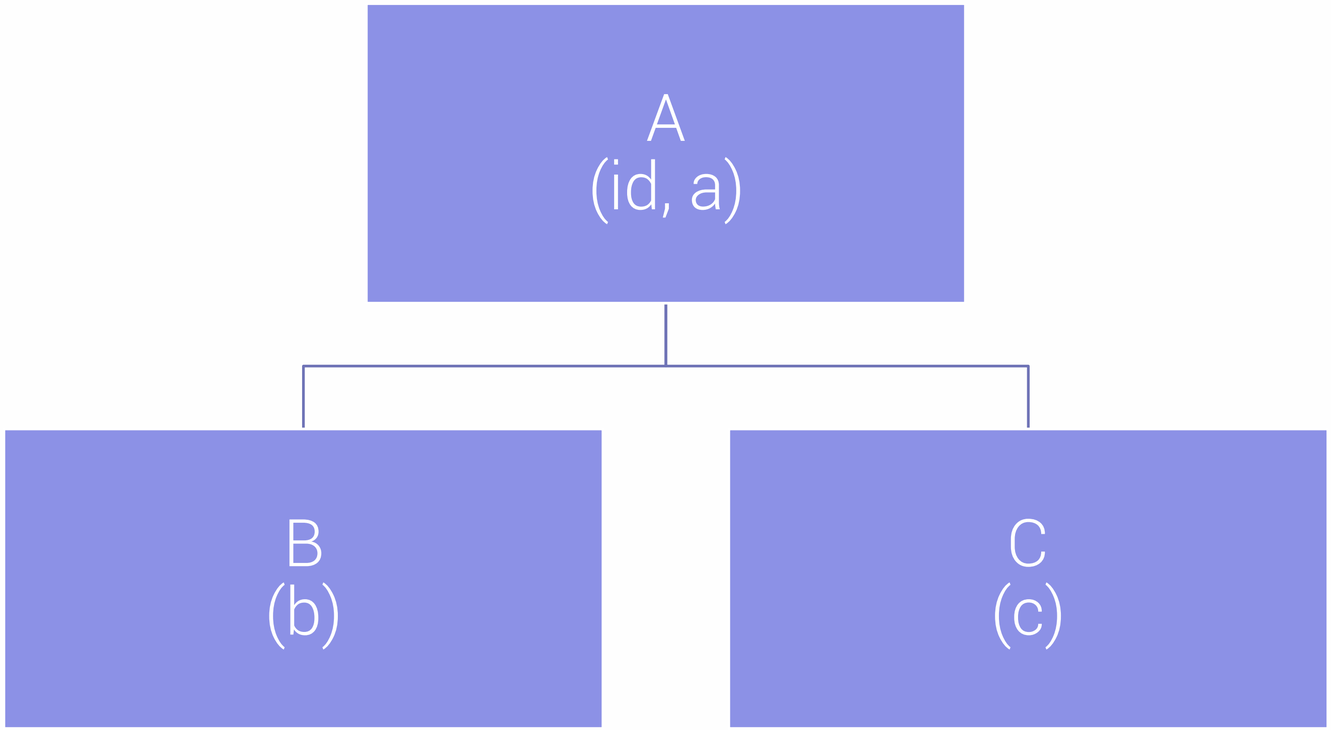
```
@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue
    private int id;

    private Email login;

    private String password;

    private Address address;
```

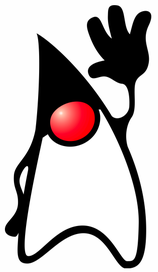


Impedance mismatch. Наследование

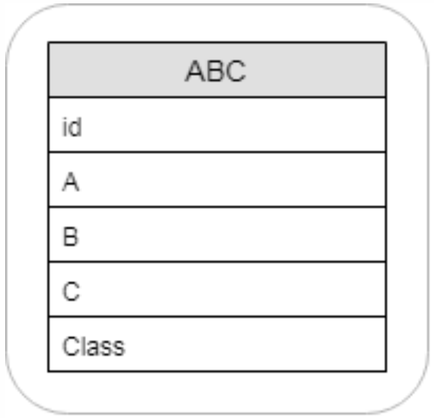
СБЕРБАНК
Корпоративный университет

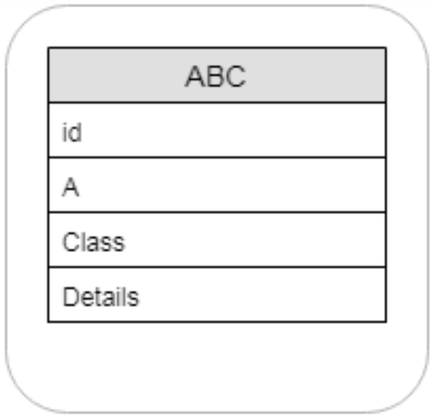
Как это реализовать в БД?
(5 способов)

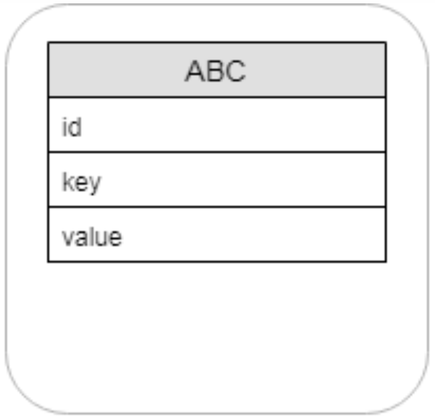


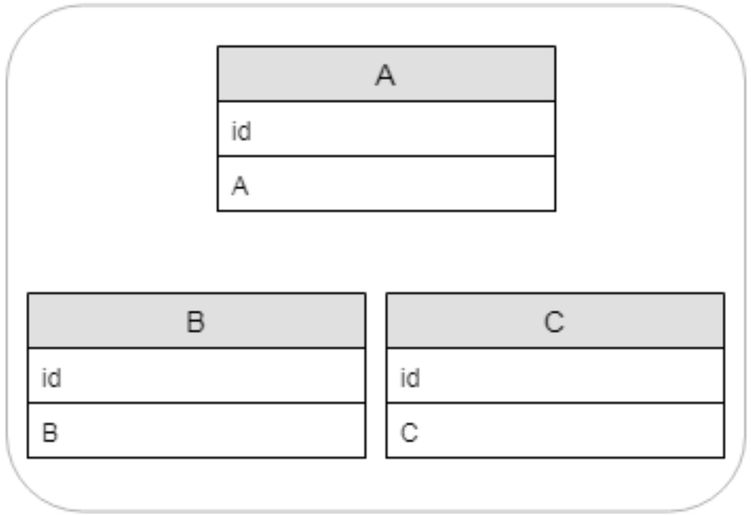


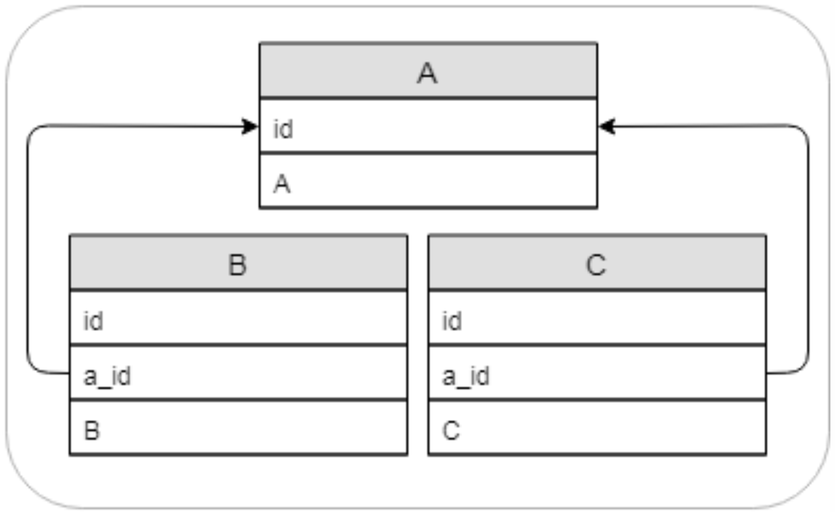
Impedance mismatch. Наследование









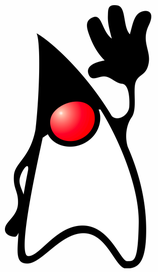


Impedance mismatch. Ассоциация

СБЕРБАНК
Корпоративный университет

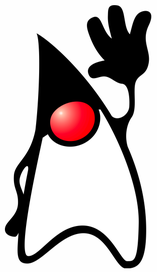
Как это реализовать в БД?

```
public class Company {

    private List<Person> employees;
```



Навигация

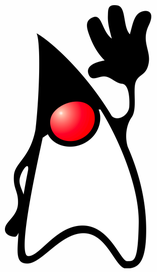
```
public class Company {

    private Person director;


    public String getDirectorName() {
        return director.getName();
    }
```



Проблема

Для того, получить данные из БД в виде объекта сущности нужно:

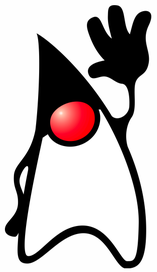
- Написать код для обхода результатов запроса
- И отображению результатов запроса на объект(ы)
- И еще придумать как достать связанные сущности
- И для этого тоже написать код



Решение №1. Паттерн Executor, JdbcTemplate

СБЕРБАНК
Корпоративный университет

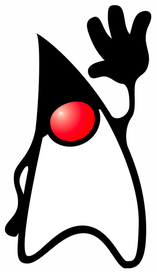
- JdbcTemplate – реализация паттерна Executor от Spring
- Результаты запроса больше не нужно обходить самим. Это делают за нас (но не всегда)
- Плюс немного легче становится отображать результаты
- Но нам все еще нужно самим заботится о том как доставать связанные сущности



Решение №2. ORM

Идея! А как бы получить то, что нам нужно, но при этом ничего самим не делать?!

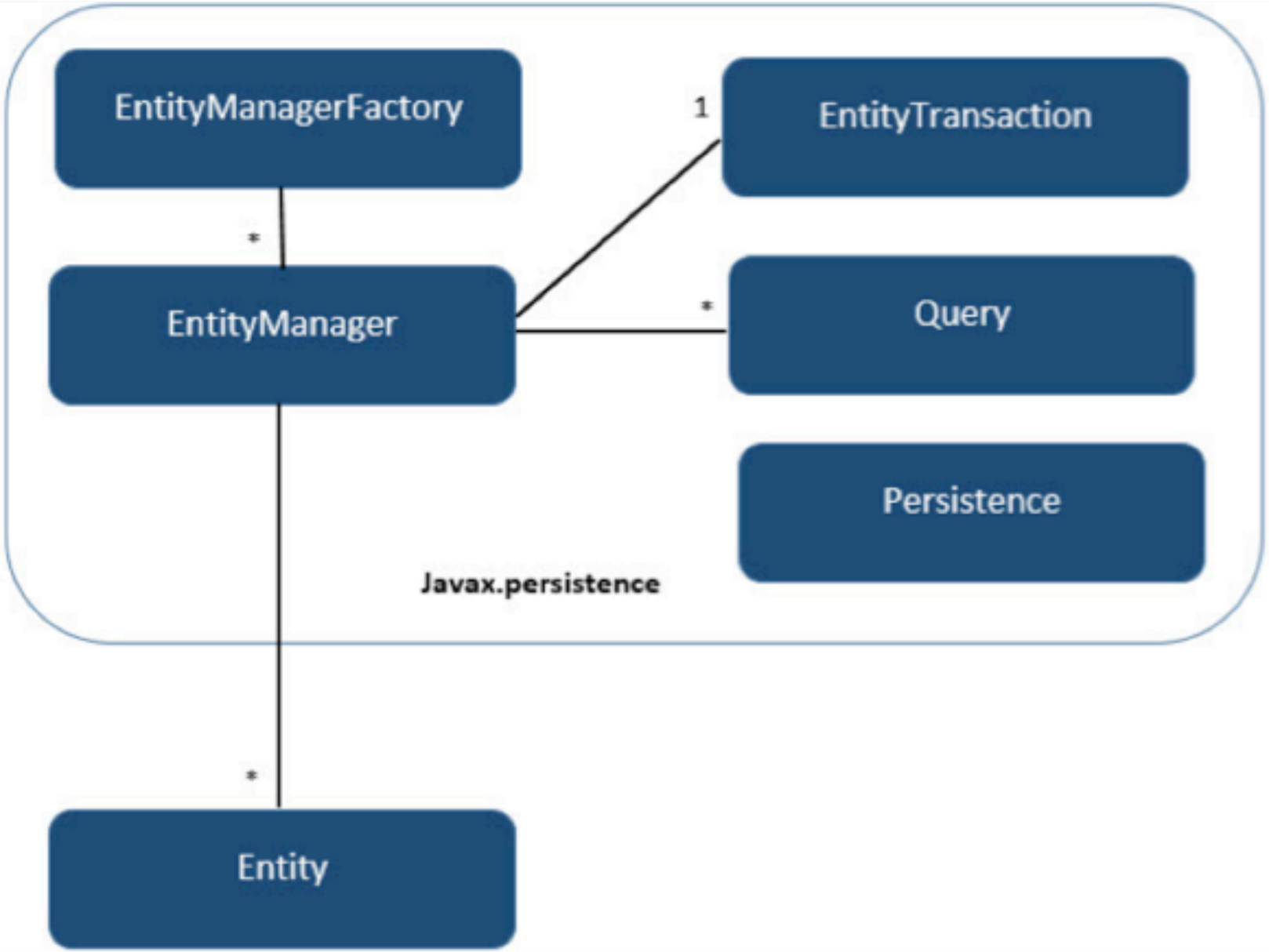
- ORM (Object Relational Mapping) - позволяет отображать сущности на БД и наоборот. Помогает ей в этом JPA
- JPA (Java Persistence API) - набор интерфейсов, в основном аннотаций, для разметки классов-сущностей, который позволяет объяснять ORM, как нужно с этими классами работать
- Т.к JPA это только аннотации и интерфейсы, то для работы с ними нужна реализация. Или провайдер. Таким провайдером является Hibernate. Не смотря на то что это не единственная реализация JPA Hibernate считается промышленным стандартом и практически ассоциируется с аббревиатурами ORM и JPA



Ваши вопросы?

Если что – их можно задать потом

Классы


EntityManagerFactory
1
EntityTransaction
*
EntityManager
*
Query
Persistence
Javax.persistence
*
Entity

Разметка сущности

СБЕРБАНК
Корпоративный университет

```java
@Entity // Указывает, что данный класс является сущностью
@Table(name = "otus_students") // Задает имя таблицы, на которую будет отображаться сущность
public class OtusStudent {
    @Id // Позволяет указать какое поле является идентификатором
    @GeneratedValue(strategy = GenerationType.IDENTITY) // Стратегия генерации идентификаторов
    private long id;

    // Задает имя и некоторые свойства поля таблицы, на которое будет отображаться поле сущности
    @Column(name = "name", nullable = false, unique = true)
    private String name;

    // Указывает на связь между таблицами "один к одному"
    @OneToOne(targetEntity = Avatar.class, cascade = CascadeType.ALL)
    // Задает поле, по которому происходит объединение с таблицей для хранения связанной сущности
    @JoinColumn(name = "avatar_id")
    private Avatar avatar;

    // Указывает на связь между таблицами "один ко многим"
    @OneToMany(targetEntity = EMail.class, cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinColumn(name = "student_id")
    private List<EMail> emails;

    // Указывает на связь между таблицами "многие ко многим"
    @ManyToMany(targetEntity = Course.class, fetch = FetchType.LAZY)
    // Задает таблицу связей между таблицами для хранения родительской и связанной сущностью
    @JoinTable(name = "student_courses", joinColumns = @JoinColumn(name = "student_id"),
            inverseJoinColumns = @JoinColumn(name = "course_id"))
    private List<Course> courses;
}
```

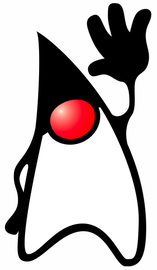
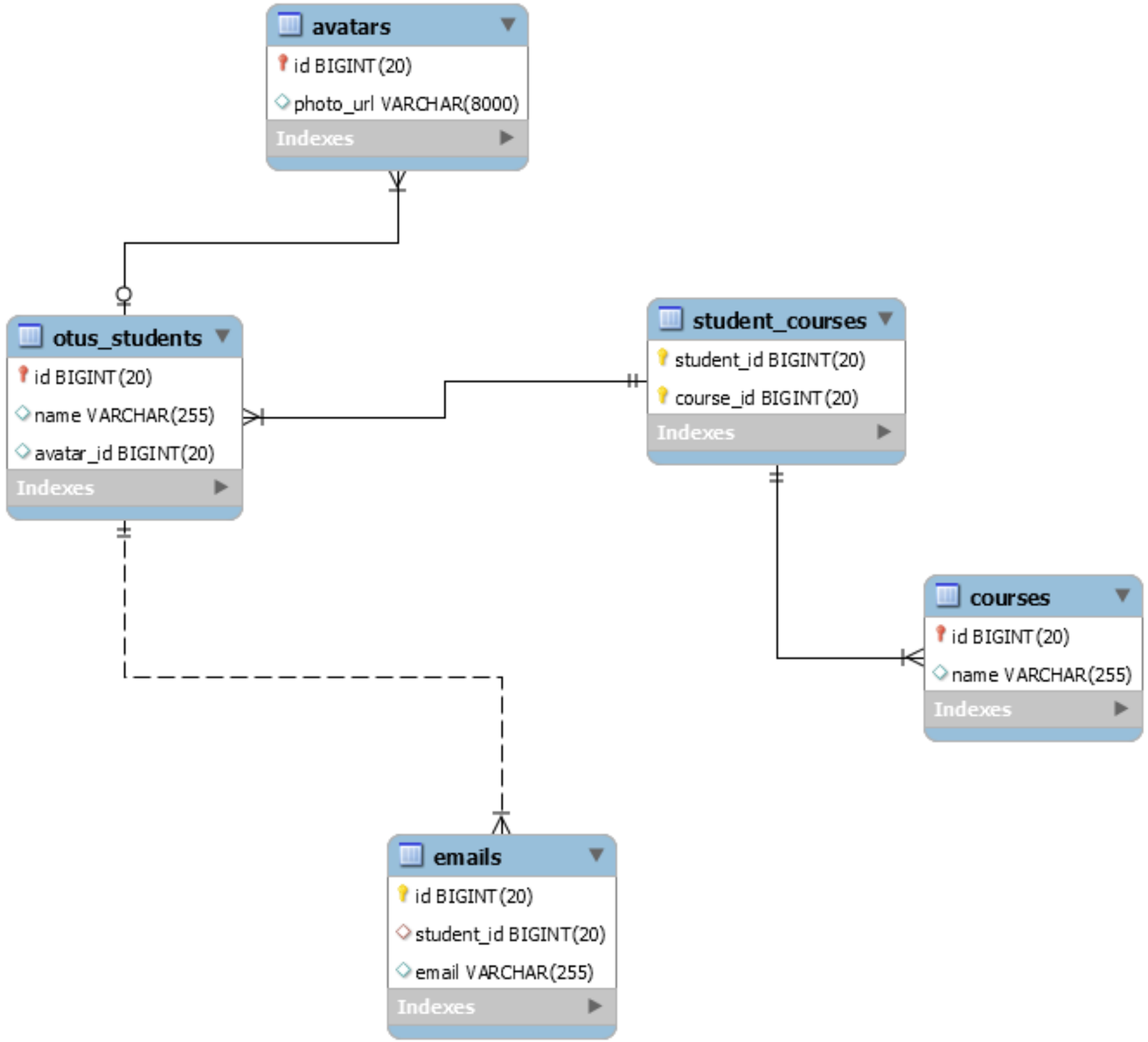


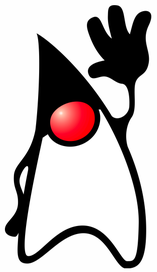
Диаграмма таблиц

СБЕРБАНК
Корпоративный университет



Для начала подключить зависимости. Для Hibernate и БД (например, H2)

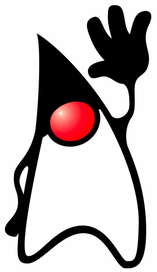
```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.4.4.Final</version>
</dependency>

<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>1.4.199</version>
</dependency>
```



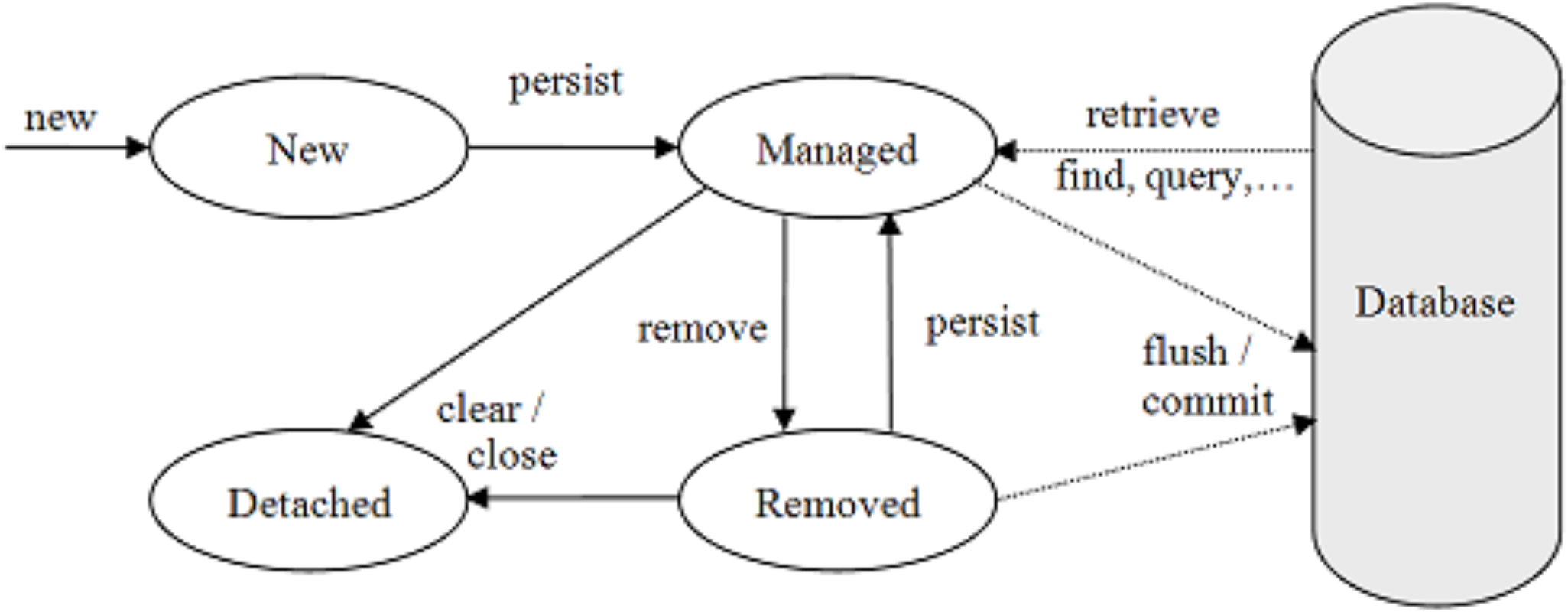
Persistence Context

СБЕРБАНК
Корпоративный университет

- Persistence Context – кэш первого уровня для объектов, которые сохраняются или загружаются в/из БД в рамках сессии
- В применении к данному контексту сущность может находиться в трех состояниях
- Transient – объект еще ни разу не был присоединен к контексту
- Persistent – объект находится под управлением контекста. Все изменения в нем попадут в БД
- Detached – объект был когда-то прикреплен к контексту, но теперь это не так. Объект переходит в это состояние, если исключить его из контекста, очистить или закрыть сессию



State entity



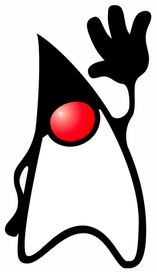
Ваши вопросы?

Если что – их можно задать потом

Как с этим жить. Работа на чистом JPA

СБЕРБАНК
Корпоративный университет

- Всеми операциями над сущностями занимается EntityManager
- Создается на основе настроек из “META-INF/persistence.xml” (в ресурсах)
- Где есть DI внедряется через @PersistenceContext

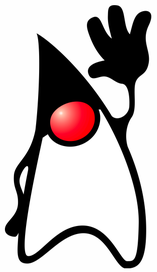


Как с этим жить. Работа на чистом JPA

СБЕРБАНК
Корпоративный университет

```java
public EntityManager createEntityManager() {
    EntityManagerFactory ef = Persistence.createEntityManagerFactory(UNIT_NAME);
    return ef.createEntityManager();
}
```

```xml
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
             version="2.1">
    <persistence-unit name="CommonUnit">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <class>ru.otus.example.hibernate.domain</class>
        <properties>
            <property name="javax.persistence.jdbc.driver" value="org.h2.Driver" />
            <property name="javax.persistence.jdbc.url"
                      value="jdbc:h2:mem:test;DB_CLOSE_DELAY=-1" />
            <property name="javax.persistence.jdbc.user" value="sa" />
            <property name="javax.persistence.jdbc.password" value="" />
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.hbm2ddl.auto" value="create-drop" />
        </properties>
    </persistence-unit>
</persistence>
```

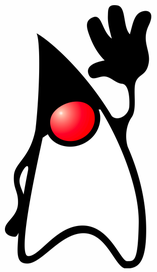


Как с этим жить. Работа через Session

СБЕРБАНК
Корпоративный университет

- Session – основной класс для работы с сущностями в Hibernate. Даже при работе через EntityManager мы в итоге работаем с Session

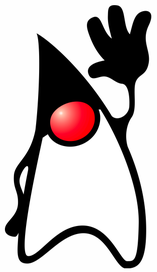
- Создается с помощью SessionFactory, которую нужно в свою очередь создать на основе настроек из “hibernate.cfg.xml" (в ресурсах). Это имя файла по умолчанию. Можно задать свое



Как с этим жить. Работа через Session

СБЕРБАНК
Корпоративный университет

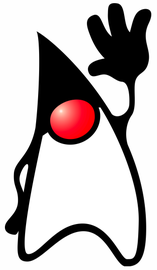
```
public final class HibernateUtils {

  private HibernateUtils() {
  }

  public static SessionFactory buildSessionFactory(String configResourceFileName,
                                                   Class ...annotatedClasses){

    Configuration configuration = new Configuration().configure(configResourceFileName);

    StandardServiceRegistry registry = new StandardServiceRegistryBuilder()
        .applySettings(configuration.getProperties()).build();

    MetadataSources mds = new MetadataSources(registry);
    Arrays.stream(annotatedClasses).forEach(mds::addAnnotatedClass);

    Metadata metadata = mds.getMetadataBuilder().build();
    return metadata.getSessionFactoryBuilder().build();
  }
}
```



Как с этим жить. Работа через Session

СБЕРБАНК
Корпоративный университет

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.H2Dialect</property>
    <property name="hibernate.connection.driver_class">org.h2.Driver</property>

    <property name="hibernate.connection.url">jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</property>

    <property name="hibernate.connection.username">sa</property>
    <property name="hibernate.connection.password">sa</property>

    <property name="hibernate.show_sql">true</property>

    <property name="hibernate.hbm2ddl.auto">create</property>
    <property name="hibernate.enable_lazy_load_no_trans">false</property>
  </session-factory>
</hibernate-configuration>
```

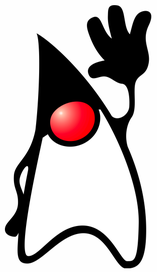


Как с этим жить. Работа через Session

СБЕРБАНК
Корпоративный университет

Основные операции (методы) Session:

- find – поиск и загрузка сущности
- persist – добавление сущности в БД
- merge – обновление сущности БД
- remove – удаление сущности
- createQuery – создание объекта запроса
- getEntityGraph – получение, ранее определенного графа объектов
- save
- update
- saveOrUpdate

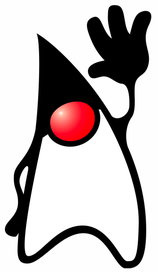


Как с этим жить. Работа через Session

СБЕРБАНК
Корпоративный университет

Основные операции (методы) Session:

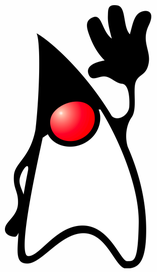
- **find, get** – поиск и загрузка объекта по его id. Объект сразу получается в состоянии persistent
- **load** – то же, что и get, но возвращается пустой proxy. Загрузка из БД произойдет в момент первого обращения к любому свойству объекта
- **persist** – меняет состояние объекта из transient в persistent. Выполняет insert. Бросает PersistentObjectException если задан id
- **merge** – меняет состояние объекта из transient или detached в persistent. Для transient работает аналогично persist. Для detached выполняет загрузку из БД и обновляет в контексте. По завершении сессии или коммита транзакции выполняет update
- **remove** – удаление объекта из БД и контекста. Меняет состояние на transient. IllegalArgumentException если объект в состоянии detached
- **createQuery** – создание объекта запроса
- **getEntityGraph** – получение, ранее определенного графа объектов
- **save** – то же, что и persist, но гарантированно вернет идентификатор. Плюс для detached объектов каждый раз генерирует новый id
- **update** – меняет состояние объекта из detached в persistent. Если объект в состоянии transient бросает исключение
- **saveOrUpdate** – вызывает save или update в зависимости от изначального состояния объекта



Как с этим жить. Важное

СБЕРБАНК
Корпоративный университет

- Работа только в рамках транзакции/сессии
- Пока в persistent изменения отображаются на БД
- LazyInitializationException – нет сессии/транзакции
- N+1 по умолчанию

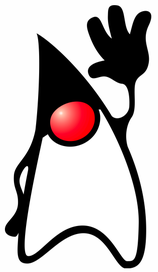


Итоги

Итоги

СБЕРБАНК
Корпоративный университет

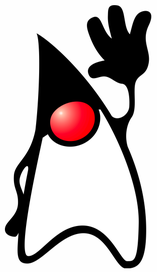
- Сами по результатам запросов не ходим
- Сами ничего не отображаем
- Связанные сущности достаются тоже сами. Думать не надо
- Кода минимум
- По сути дела весь SQL – в маппинге Entity
- Entity – обычно на слое домена (бизнеса)
- Поэтому DAO тоже на бизнес слое
- И DAO теперь принято называть репозиторием



Итоги. Плюсы ORM

СБЕРБАНК
Корпоративный университет

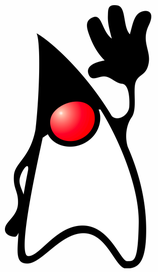
- ORM теперь позволяет оперировать бизнес-понятиями и ООП, поля вместо столбцов, объекты вместо таблиц (хотя они могут быть размазаны по разным таблицам)
- ORM теперь позволяет не знать SQL (нет)
- ORM позволяет абстрагироваться от диалекта SQL (!!!!!!)
- А с JPA и от конкретного провайдера JPA
- Тестировать – одно удовольствие, можно тестировать на H2 и вообще где угодно
- Тупо меньше кода



Итоги. Минусы ORM

СБЕРБАНК
Корпоративный университет

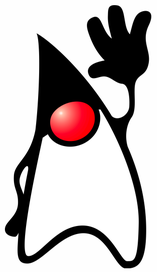
- Это магия
- Как и за любую магию приходится платить производительностью
- Нет нормального доступа к нативному SQL и плюшкам БД



Итоги. ORM vs JDBC

СБЕРБАНК
Корпоративный университет

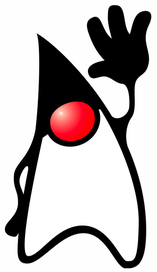
- Нужно решить, что проекту нужно, может это прототип
- Какая нагрузка (кто говорит, что у них будет большая нагрузка – скорее всего не правы)
- Сильно зависит от числа бизнес-объектов
- Больше 10-20 взаимосвязанных бизнес объектов – разрабатывать на SQL становится сильно тяжело (Связей – квадрат от числа entity)



Итоги. ORM vs JDBC

СБЕРБАНК
Корпоративный университет

- Если приложение действительно (!) под нагрузкой, то лучше использовать чистый JDBC. SQL не такой страшный
- Если много бизнес-объектов – ORM незаменим
- Периодически встречается комбинация



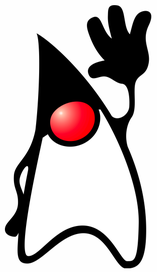
Ваши вопросы?

Если что – их можно задать потом

План

- Абстракции. ORM
- Hibernate ORM

- **Написать своё приложение с БД**



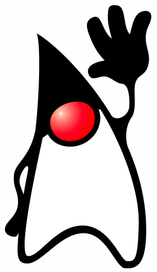
03

Своё приложение с БД

Задача

СБЕРБАНК
Корпоративный университет

- Библиотека.
- Книги могут приходить новые, могут выдаваться/возвращаться клиентами, списываться



Ваши вопросы?

Если что – их можно задать потом

Спасибо за внимание!