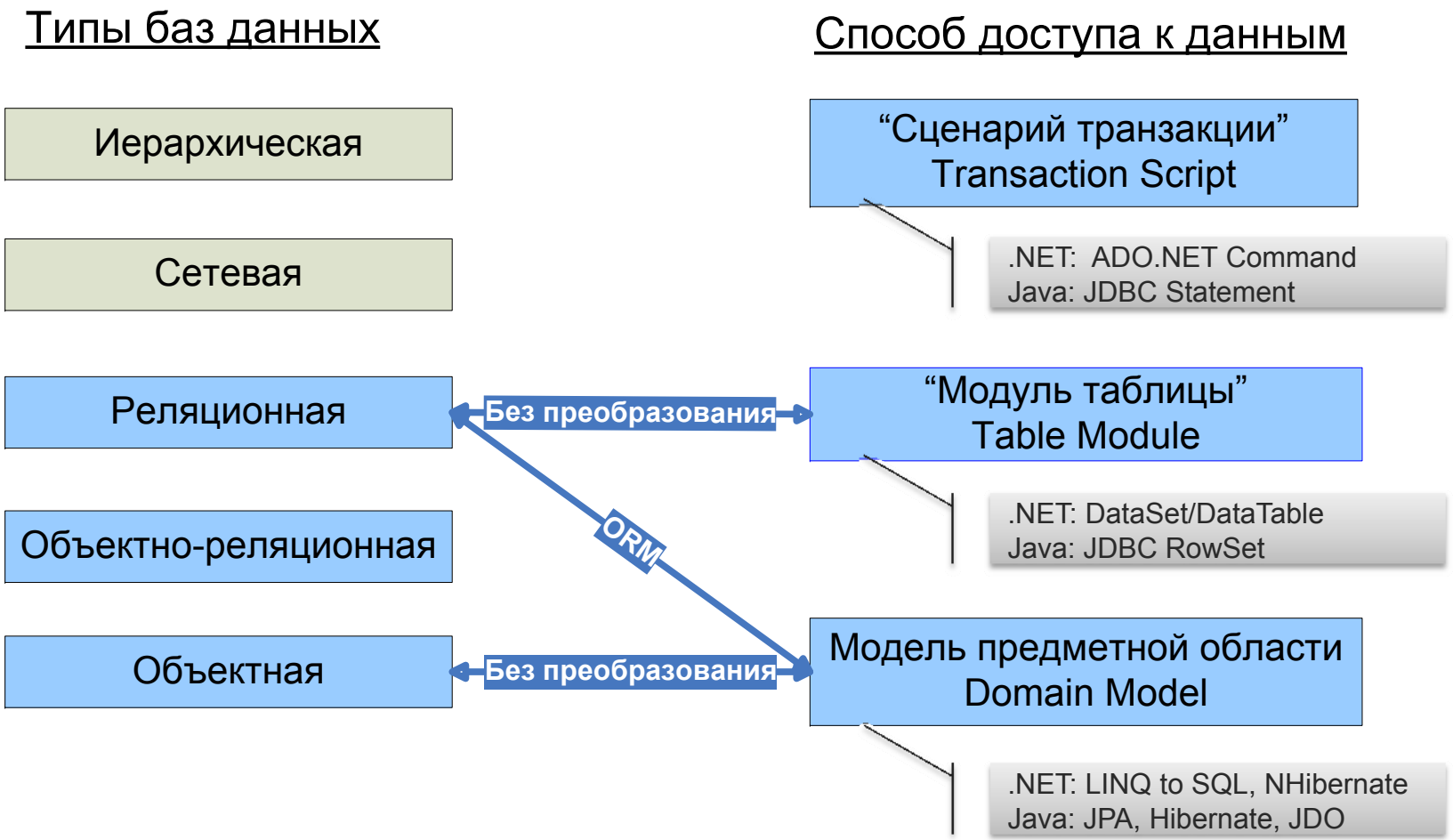
Object Relational Mapping (ORM)

Объектно-реляционное отображение

29.03.2010

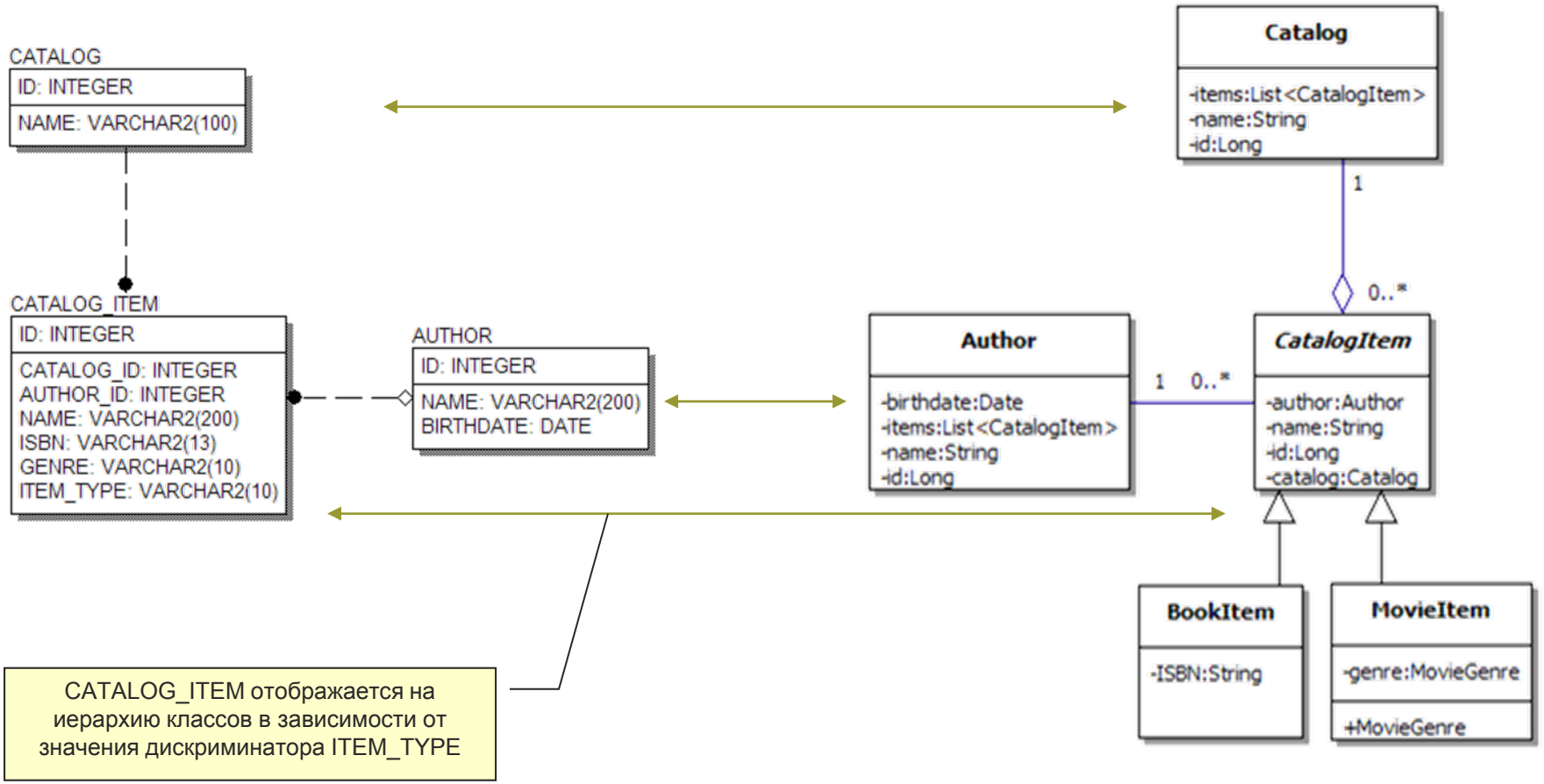
Взаимодействие между типами баз данных и подходами доступа к данным



Примеры реляционной и объектной модели

Реляционная модель

Объектная модель предметной области



Отображение таблицы CATALOG и класса Catalog.java

```
1    package org.mai806.catalog.model;
2
3    import java.util.List;
4
5    import javax.persistence.*;
6
7    @javax.persistence.SequenceGenerator(
8        name="SEQ_ID",
9        sequenceName="SEQ_ID"
10   )
11   @Entity @Table(name="CATALOG")
12   public class Catalog {
13
14       private Long id;
15       private String name;
16       private List<CatalogItem> items;
17
18       @Id
@GeneratedValue(generator="SEQ_ID")
19       @Column(name="ID")
20       public Long getId() {
21           return id;
22       }
23       public void setId(Long id) {
24           this.id = id;
25       }
26
27       @Column(name="NAME")
28       public String getName() {
29           return name;
30       }
31       public void setName(String name) {
32           this.name = name;
33       }
34
35       @OneToMany(cascade=CascadeType.ALL,
fetch=FetchType.LAZY)
36       @JoinColumn(name="CATALOG_ID", nullable = false)
37       public List<CatalogItem> getItems() {
38           return items;
39       }
40       public void setItems(List<CatalogItem> items) {
41           this.items = items;
42       }
43
44   }
45
```

- Класс Catalog связан с таблицей CATALOG
- Первичный ключ – **ID**, связан со свойством id (функции **getId()/setId()**)
- Для генерации значений первичного ключа **ID** используется sequence **SEQ_ID**
- Атрибут **NAME** связан со свойством **name** (функции **getName()/setName()**)
- Объекты **Catalog** содержат список **CatalogItem**
  - связь “один-ко-многим” (**@OneToMany**),
  - обязательная (**nullable=false**)
  - Внешний ключ, определяющий связь – **CATALOG_ID**
- Список CatalogItem загружаются по-требованию (**FetchType.LAZY**)
- При сохранении объекта **Catalog** автоматически сохраняются все его items (**CascadeType.ALL**)

Работа с Hibernate API

```
17    // Получим сессию Hibernate
18    Session session = HibernateUtil.getSession();
19
20    // Начало транзакции
21    session.beginTransaction();
22
23    // Создание нового каталога и заполнение данными
24    Catalog catalog = new Catalog();
25    catalog.setName("Мой каталог №1");
26
27    List<CatalogItem> items =
28            new LinkedList<CatalogItem>();
29
30    Author fauler = new Author();
31    fauler.setName("Мартин Фаулер");
32    fauler.setBirthdate(new Date());
33
34    Author burton = new Author();
35    burton.setName("Тим Бартон");
36    burton.setBirthdate(new Date());
37
38    BookItem book = new BookItem();
39    book.setAuthor(fauler);
40    book.setName("UML Основы");
41    book.setISBN("1-232-12345-2");
42    book.setCatalog(catalog);
43
44    MovieItem movie = new MovieItem();
45    movie.setAuthor(burton);
46    movie.setName("Кошмар перед рождеством");
47    movie.setGenre(MovieItem.MovieGenre.MUSIC);
48    movie.setCatalog(catalog);
49
50    items.add(book);
51    items.add(movie);
52
53    catalog.setItems(items);
54
55    // Сохранение каталога в базу данных
56    session.save(catalog);
57    // Завершение транзакции
58    session.getTransaction().commit();
59
60    System.out.println("Id="+catalog.getId());
61
62    // Вывод списка каталогов
63    List<Catalog> list =
64      (List<Catalog>)session.createQuery("from Catalog").list();
65
66    for (Catalog cat : list) {
67        System.out.println(cat.getId() + " " + cat.getName());
68    }
```

- Работая с данными в объектно-ориентированном языке, мы работаем с объектами, заполняя и считывая значения полей, создавая новые или изменяя существующие объекты, определяя зависимости между объектами
- При операции save() мы передаем объект типа **Catalog**, который сохраняется в базу данных по описанным правилам отображения. В том числе сохраняются и все зависимые объекты (**CatalogItem**)
- Составляя запросы к базе данных, мы уже указываем не столбцы таблицы, а свойства объектов

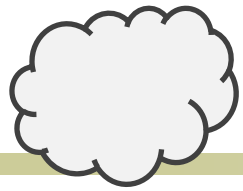
Протокол команд SQL

Oracle

```
Hibernate: select SEQ_ID.nextval from dual
Hibernate: select SEQ_ID.nextval from dual
Hibernate: select SEQ_ID.nextval from dual
Hibernate: select SEQ_ID.nextval from dual
Hibernate: select SEQ_ID.nextval from dual
Hibernate: insert into CATALOG (NAME, ID) values (?, ?)
Hibernate: insert into AUTHOR (NAME, BIRTHDATE, id) values (?, ?, ?)
Hibernate: insert into CATALOG_ITEM (NAME, AUTHOR_ID, CATALOG_ID, ISBN, ITEM_TYPE, id) values (?, ?, ?, ?, 'BOOK', ?)
Hibernate: insert into AUTHOR (NAME, BIRTHDATE, id) values (?, ?, ?)
Hibernate: insert into CATALOG_ITEM (NAME, AUTHOR_ID, CATALOG_ID, GENRE, ITEM_TYPE, id) values (?, ?, ?, ?, 'MOVIE', ?)
Hibernate: update CATALOG_ITEM set CATALOG_ID=? where id=?
Hibernate: update CATALOG_ITEM set CATALOG_ID=? where id=?
Hibernate: select catalog0_.ID as ID0_, catalog0_.NAME as NAME0_ from CATALOG catalog0_
```

SQL Server

```
Hibernate: insert into CATALOG (NAME) values (?)
Hibernate: insert into AUTHOR (NAME, BIRTHDATE) values (?, ?)
Hibernate: insert into CATALOG_ITEM (NAME, AUTHOR_ID, CATALOG_ID, ISBN, ITEM_TYPE) values (?, ?, ?, ?, 'BOOK')
Hibernate: insert into AUTHOR (NAME, BIRTHDATE) values (?, ?)
Hibernate: insert into CATALOG_ITEM (NAME, AUTHOR_ID, CATALOG_ID, GENRE, ITEM_TYPE) values (?, ?, ?, ?, 'MOVIE')
Hibernate: update CATALOG_ITEM set CATALOG_ID=? where id=?
Hibernate: update CATALOG_ITEM set CATALOG_ID=? where id=?
Hibernate: select catalog0_.ID as ID0_, catalog0_.NAME as NAME0_ from CATALOG catalog0_
```



Возможности ORM

- Загрузка связанных объектов “по требованию” (lazy loading)
- Обеспечение пессимистической/оптимистической блокировок
- Кэширование загруженных объектов
- SQL-подобные запросы по объектной модели

Преимущества ORM

- Нет необходимости писать рутинные insert/update/delete/select для CRUD операций
- Условия связи между объектами (строками таблиц) указываются декларативно в одном месте.
- Возможность использовать полиморфные запросы для иерархий классов
- Высокая степень независимости от конкретной СУБД

Недостатки ORM

- Возможны проблемы с производительностью для сложных запросов на объектном SQL.
- Затрудняет использование специфических конструкций языка SQL конкретной СУБД.
- **The object-relational impedance mismatch**

Реализации ORM

- Hibernate/NHibernate www.hibernate.org (Java / .NET 1.1-3.5)
- ADO.NET Entity Framework (.NET 3.5)
- Oracle® TopLink® (Java)
- iBatis framework (Java, .NET) http://ibatis.apache.org/
- JPOX Java Data Objects (Java) http://www.jpox.org
- …

Стандарты ORM

- EJB 1.1 Entity Beans (legacy)
  - Устаревший стандарт, используется только в legacy приложениях
- Java Data Object (JDO)
  - Редко используемый стандарт
  - Реализации: JPOX, OpenAccess JDO
- Java Persistence API (JPA)
  - Наиболее популярный
  - Составная часть стандартов EJB 3.0 и JEE 5
  - Реализации: Hibernate, Oracle TopLink, KODO (OpenJPA)

Литература и ссылки

Общая литература:

- Мартин Фаулер “Архитектура корпоративных программных приложений”. М., “Вильямс”, 2004
- http://en.wikipedia.org/wiki/Object-relational_mapping

Hibernate:

- http://www.hibernate.org/hib_docs/reference/ru/html_single/
- Java Persistence with Hibernate / Christian Bauer, Gavin King / Manning, 2006
- NHibernate in Action / Pierre Henri Kuaté, Tobin Harris, Christian Bauer, and Gavin King / Manning 2009