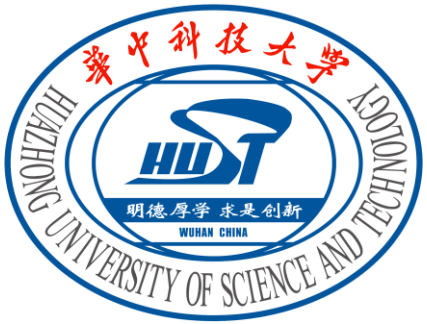
基于Java的面向对象程序设计

陈维亚

weiya_chen@hust.edu.cn

华中科技大学软件学院

第27-28讲：Java 数据库



目录

1. 数据库简介

❑ 数据库 Database

1951：Univac系统使用磁带和穿孔卡片作为数据存储。

1956：IBM公司在其Model 305 RAMAC中第一次引入了磁盘驱动器。

1961：通用电气(GE)公司的 Charles Bachman开发了第一个数据库管理系统——IDS

1969：E.F.Codd发明了关系数据库。

1973：由John J. Cullinane领导Cullinane公司开发了IDMS——一个针对IBM主机的基于网络模型的数据库。

1976：Honeywell公司推出了Multics Relational DataStore——第一个商用关系数据库。

1979：Oracle公司引入了第一个商用SQL关系数据库管理系统。

1983：IBM推出了DB2数据库产品。

1985：为Procter&Gamble系统设计的第一个商务智能系统产生。

1991：Bill Inmon发表了“构建数据仓库”。

人们不满足于使用一般文件来管理数据

1. 数据库简介

❑ 数据库 Database

数据库是一种结构化的数据文件，相比较一般的文件，优化了对数据的**访问**、**查询**和**修改**。

A database is a collection of data or information that is organized so that it can easily be accessed, managed, and updated.

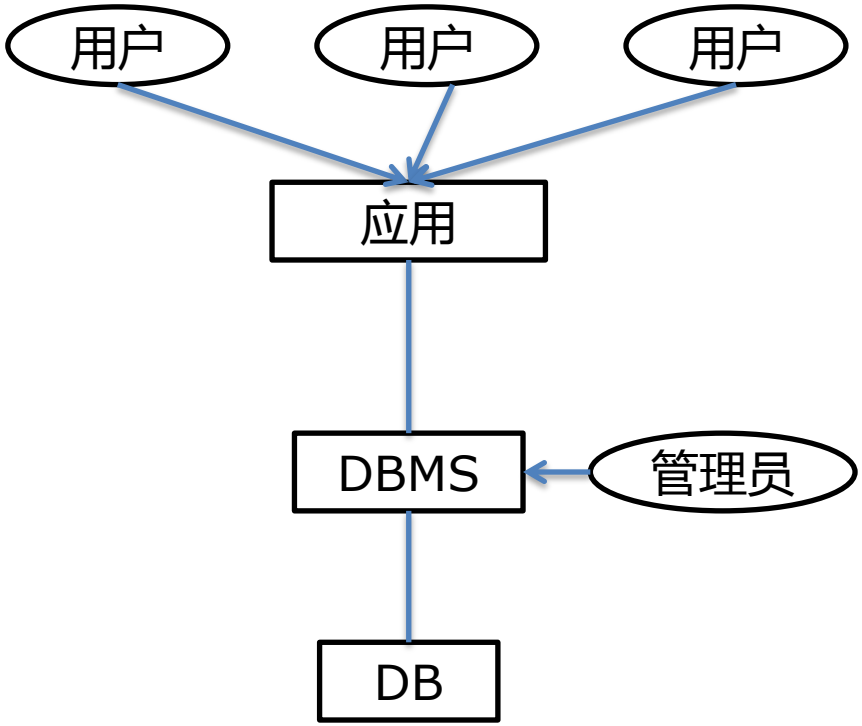
1. 数据库简介

❑ 数据库管理系统 Database Management System (DBMS)

DBMS是操纵和管理数据库的大型软件

用于**建立**、**使用**和**维护**数据库

便利性、安全性、完整性、并发性



ORACLE

Microsoft SQL Server 2008

MySQL

PostgreSQL

1. 数据库简介

❑ 数据库分类

➢ 关系型数据库

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

一个数据库通常包含一个或多个表。表包含带有数据的记录（行）。

➢ 其它数据库

键值对：Redis

文档数据库：MongoDB

图片数据库：Neo4J

列存储：Hbase

2. SQL

❑ 结构化查询语言 Structured Query Language

- 针对**关系型数据库**，实现对数据的增删改查等操作，而不仅是查询；
- 以**数据表**为基本操作单位；
- 是一种符合ANSI（美国国家标准化组织）标准的计算机语言；

❑ 语法

SQL 分为两个部分：

数据操作语言 (DML)

数据定义语言 (DDL)

查询和更新指令构成了DML 部分：

- **SELECT** - 从数据库表中获取数据
- **UPDATE** - 更新数据库表中的数据
- **DELETE** - 从数据库表中删除数据
- **INSERT INTO** - 向数据库表中插入数据

数据定义语言 (DDL) 部分负责创建或删除表格，也可以规定表之间的链接和约束：

- **CREATE DATABASE** - 创建新数据库
- **ALTER DATABASE** - 修改数据库
- **CREATE TABLE** - 创建新表
- **ALTER TABLE** - 变更（改变）数据库表
- **DROP TABLE** - 删除表

- SQL 对大小写不敏感，但是推荐大家对关键字使用全大写；
- 某些数据库系统要求在每条 SQL 命令的末端使用分号；

❑ 语句

➢ SELECT

```
SELECT 列名 FROM 表名

SELECT * FROM 表名

SELECT DISTINCT 列名 FROM 表名
```

```
SELECT LastName, FirstName FROM Persons
```

LastName	FirstName
Adams	John
Bush	George
Carter	Thomas

Persons

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

❑ 语句

➢ WHERE 子句

```
SELECT 列名 FROM 表名 WHERE 列 运算符 值
```

```
SELECT LastName, FirstName FROM
Persons WHERE City = 'Beijing'
```

LastName	FirstName
Carter	Thomas

Persons

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

❑ 语句

➢ AND 和 OR 运算符

- 可在 WHERE 子语句中把两个或多个条件结合起来。
- 可多级嵌套使用。

```
SELECT LastName, FirstName FROM Persons
WHERE FirstName='Thomas' AND
LastName='Carter'
```

LastName	FirstName
Carter	Thomas

Persons

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	Xuanwumen 10	Beijing

❑ 语句

➢ ORDER BY

- 根据指定的列对结果集进行排序 。
- 默认按照升序对记录进行排序 ，可使用 DESC 降序。

```
SELECT 列名 FROM 表名 ORDER BY 列 (DESC)
```

Persons

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	Xuanwumen 10	Beijing

❑ 语句

➢ INSERT INTO

- 用于向表格中插入新的行 。

```
INSERT INTO 表名 VALUES (值1, 值2, …)

INSERT INTO 表名(列1, 列2, …) VALUES (值1, 值2, …)
```

Persons

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	Xuanwumen 10	Beijing

❑ 语句

➢ UPDATE

- 用于修改表中的数据 。

```
UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值
```

Persons

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	Xuanwumen 10	Beijing

❑ 语句

➢ DELETE

- 用于删除表格中的行 。

```
DELETE FROM 表名称 WHERE 列名称 = 某值

DELETE * FROM 表名称
```

Persons

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	Xuanwumen 10	Beijing

❑ 函数

AVG(column)	返回某列的平均值
COUNT(column)	返回某列的行数（不包括 NULL 值）
COUNT(*)	返回被选行数
FIRST(column)	返回在指定的域中第一个记录的值
LAST(column)	返回在指定的域中最后一个记录的值
MAX(column)	返回某列的最高值
MIN(column)	返回某列的最低值
STDEV(column)	标准差
SUM(column)	求和

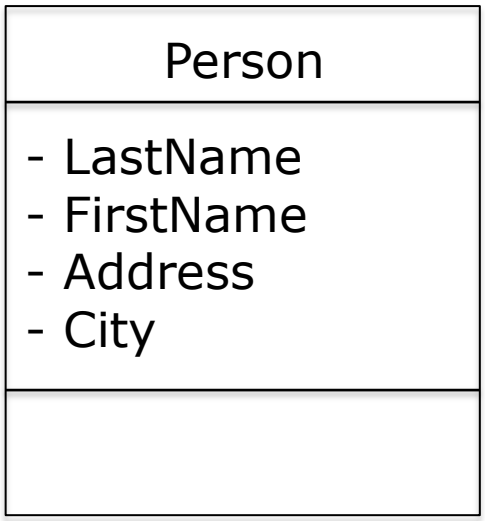
举例

```
SELECT COUNT(列名) FROM 表名

SELECT COUNT(DISTINCT 列名) FROM 表名
```

- SQL 测试链接

http://www.w3school.com.cn/sql/index.asp

❑ 对象与数据表



Persons

Id	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing
4	Carter	William	Xuanwumen 10	Beijing

保存对象时，我们只需保存其属性值

属性值以数据表的形式保存

一行记录对应一个对象的信息

❑ JDBC

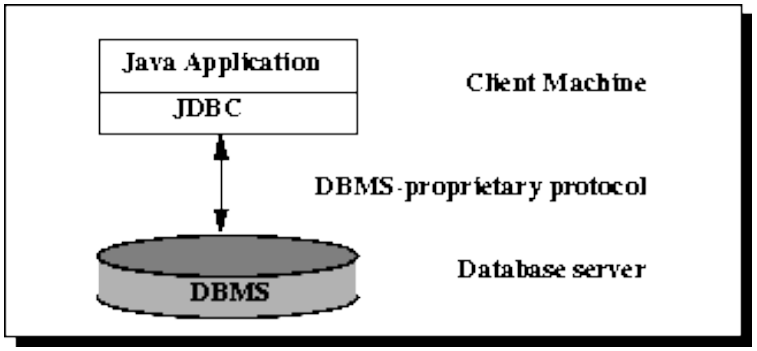
▪ JDBC（Java DataBase Connectivity, java数据库连接）是一种用于执行SQL语句的Java API，为多种关系数据库提供统一访问;

▪ 它由一组用Java语言编写的类和接口组成。JDBC为工具/数据库开发人员提供了一个标准的API，据此可以构建更高级的工具和接口，使数据库开发人员能够用纯 Java API 编写数据库应用程序。

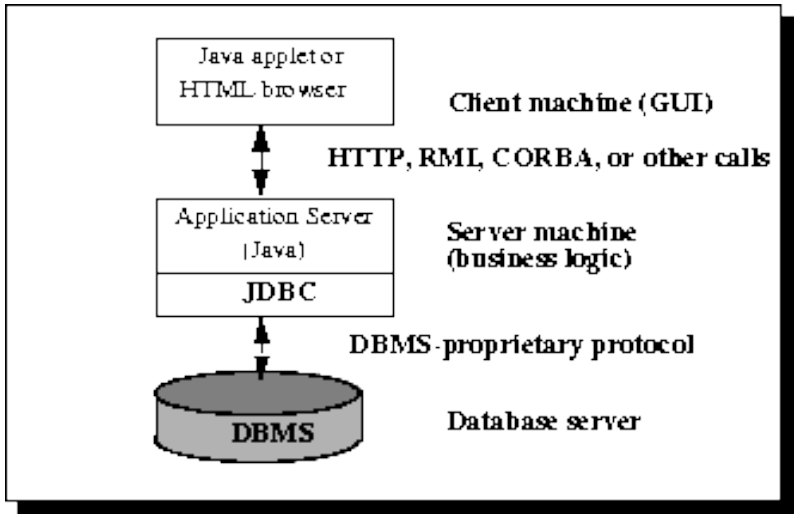
▪ JDBC 位于 java.sql 包中，主要包含如下元素：

- DriverManager 类
- Connection 接口
- Statement 接口
- ResultSet 接口

JDBC 构架



双边处理模型



三边处理模型

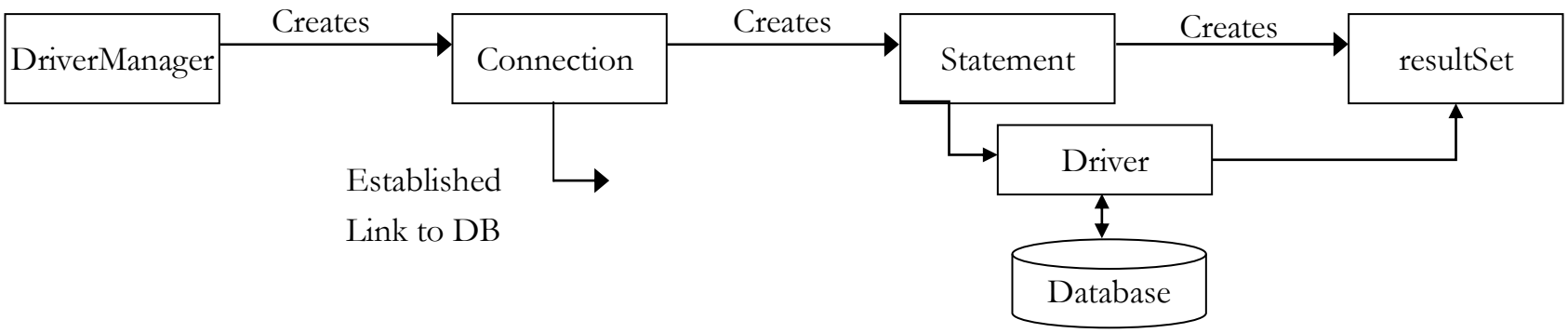
JDBC 的功能

1）连接数据源，比如数据库

2）向数据库发送所要执行的增删改查语句

3）从数据库接收语句执行的结果

3. Java 数据库开发




```
public void connectToAndQueryDatabase(String username, String
password) {
    Connection con = DriverManager.getConnection(
                        "jdbc:myDriver:myDatabase",
                        username,
                        password);

    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");

    while (rs.next()) {
        int x = rs.getInt("a");
        String s = rs.getString("b");
        float f = rs.getFloat("c");
    }
}
```

JDBC

Class Name	Description
DriverManager	getConnection method establishes a connection to the database and returns a connection instance
Connection	createStatement method returns a statement instance, which is used to execute SQL statements
Statement	contains methods that execute SQL statements: executeQuery for SQL `select` executeUpdate for SQL INSERT, UPDATE, and DELETE
ResultSet	结果集：存储查询结果的对象（类）。 **ResultSet methods:** next, first, last, and previous point the cursor to a specific row. Methods getInt, getString, etc., retrieve data from a row.

ODBC

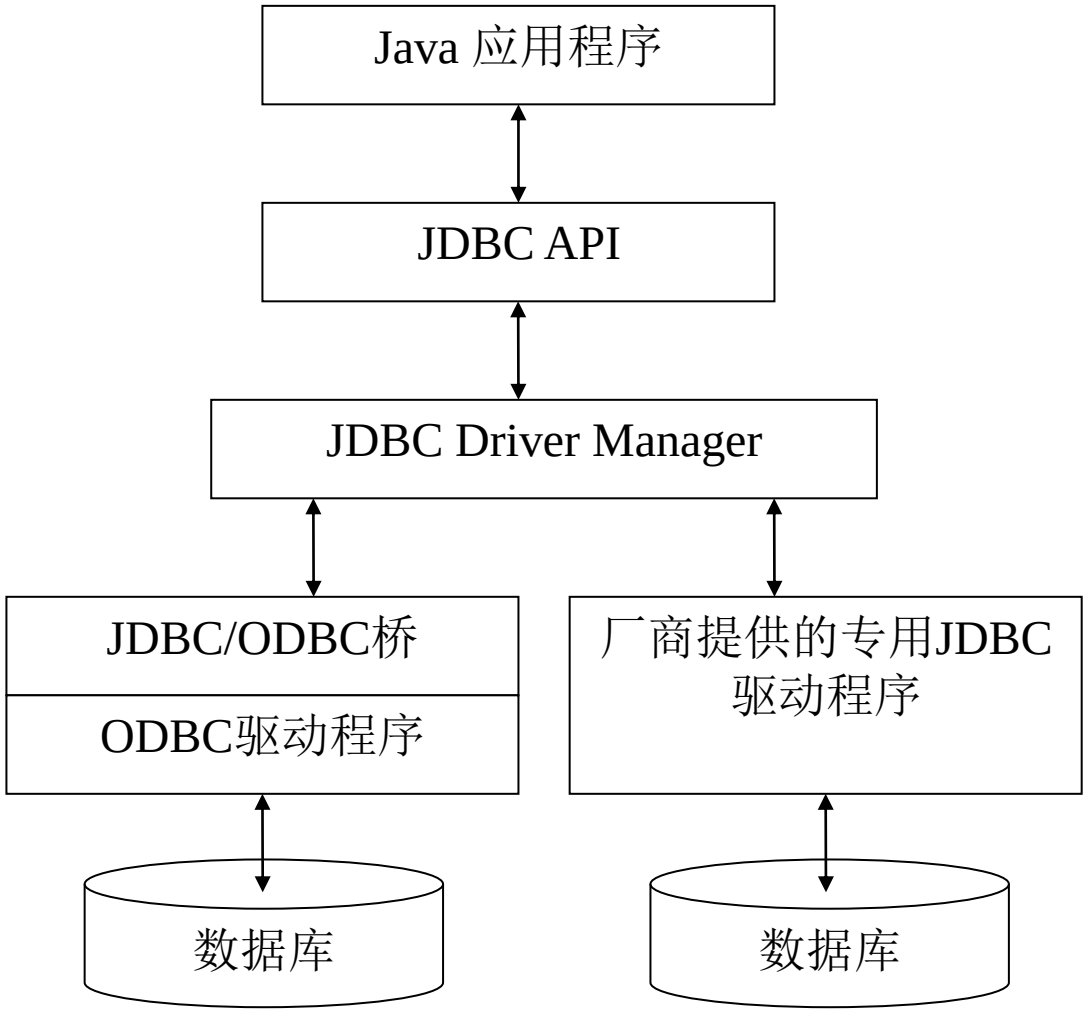
- ODBC (Open Database Connectivity)

- ODBC是微软（MS）提出的一套数据库连接标准（A set of APIs for Database access），它和JDBC优点类似，主要是为了设置数据库连接接口的统一。

- ODBC 主要用于访问微软的数据库 (Microsoft SQL Server, Microsoft Access)

- 要通过JDBC访问(存取) MS 数据库， 需通过ODBC，在JDBC和ODBC之间必须有一个相应的“桥”（JdbcOdbcDriver）实现连接，将JDBC调用转换为ODBC调用。

- ODBC


Java 应用程序
JDBC API
JDBC Driver Manager
JDBC/ODBC桥
ODBC驱动程序
厂商提供的专用JDBC
驱动程序
数据库
数据库

❑ 演化

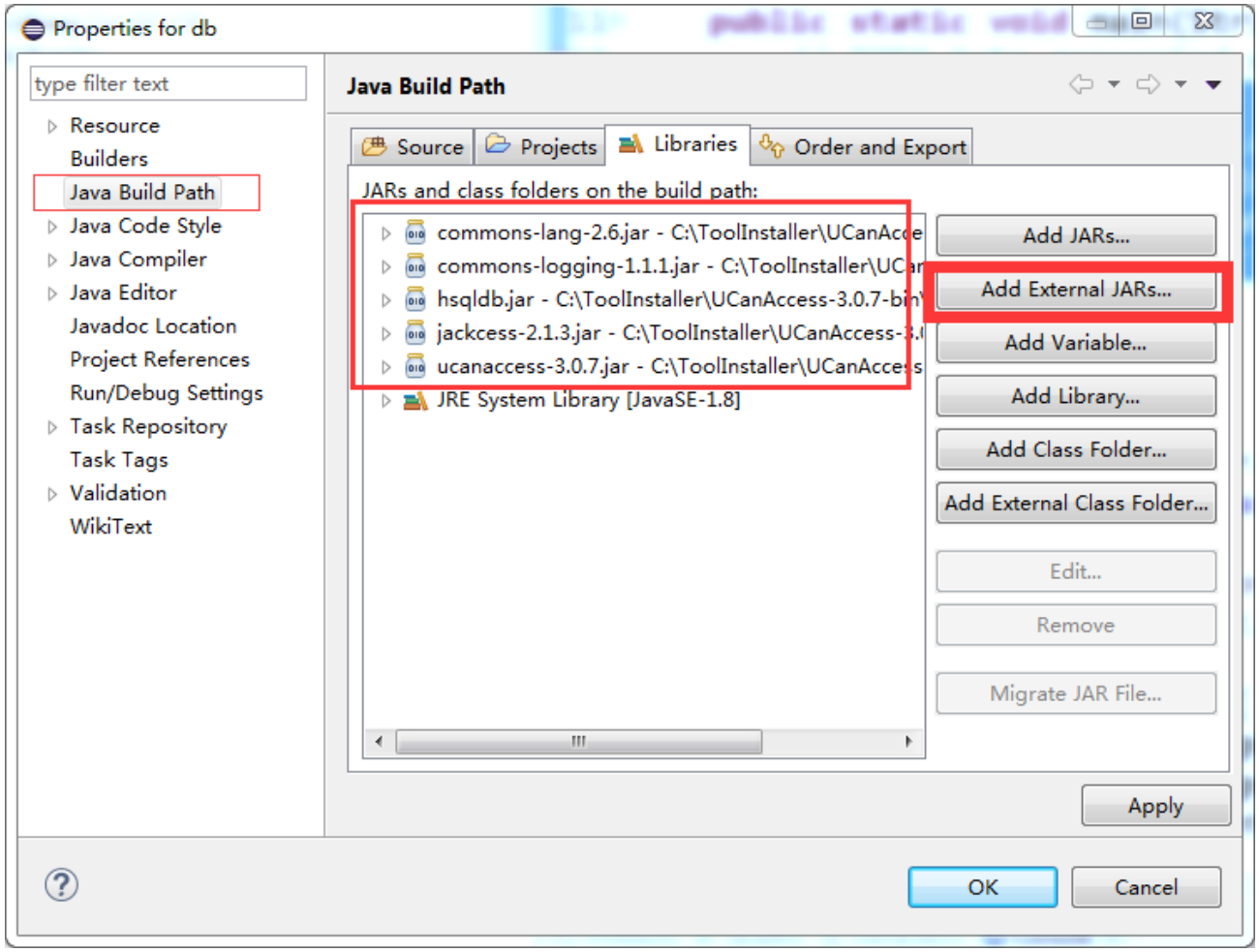
java.lang.ClassNotFoundException: sun.jdbc.odbc.JdbcOdbcDriver Exception occuring

For Java 7 you can simply **omit** the Class.forName() statement as it is not really required.

For Java 8 you **cannot** use the JDBC-ODBC Bridge because it **has been removed**. You will need to use something like **UCanAccess** instead.

UCanAccess is a pure Java JDBC driver that allows us to read from and write to **Access** databases without using ODBC. It uses two other packages, *Jackcess* and *HSQLDB*

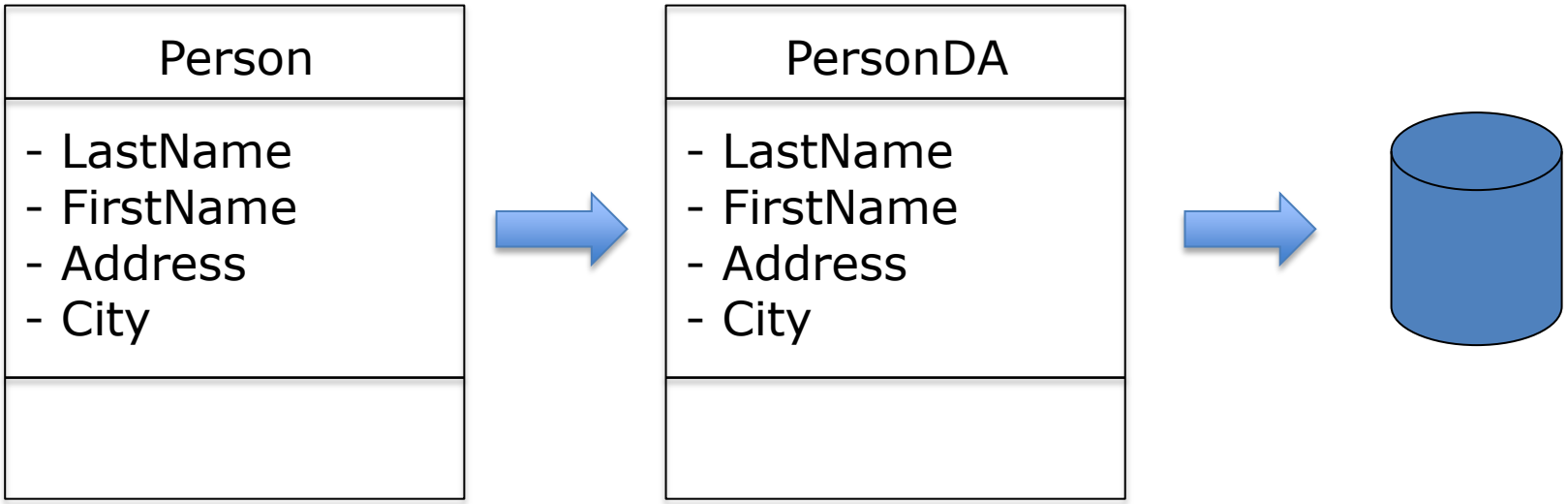
❑ 使用 UCanAccess



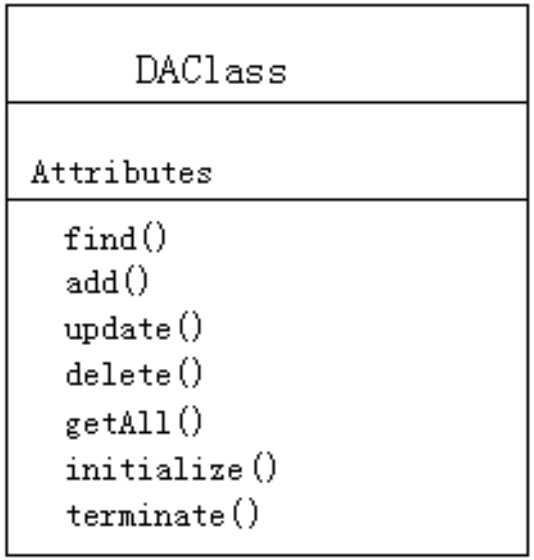
❑ [举例] 使用 UCanAccess

```
try {
    Connection con = DriverManager.getConnection(
        "jdbc:ucanaccess://C:\\path\\Persons.accdb", "", "");

    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT FirstName, LastName
FROM Persons");

    while (rs.next()) {
        String x = rs.getString("FirstName");
        String s = rs.getString("LastName");
        System.out.println(x + " : " + s);
}

} catch (SQLException e) {
    e.printStackTrace();
}
```

- 数据访问类 Data Access (DA) Class



❑ 数据访问类 Data Access (DA) Class



➢ 目的

分离代码中的业务逻辑和数据库访问部分

➢ 优点

- 模块化，减少维护成本
- 更便于分布式访问：GUI, PD, 和 DA 可以分布在不同的机器上
- 每个问题域 (PD) 类有一个单独的 DA 类与之对应
- DA 类中定义的方法仅被其对应的 PD 类中的相应方法调用

4. 使用举例

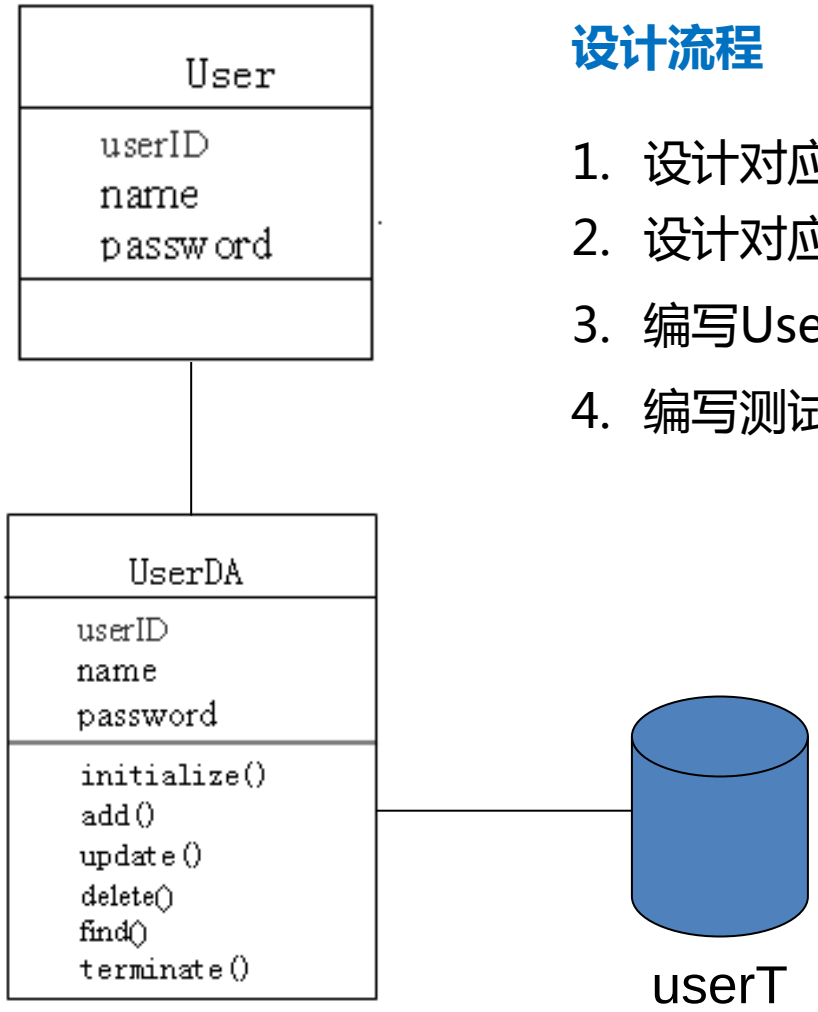
例1：单一表访问

例2： 1对1关系数据表访问

例3： 1对多关系数据表访问

4. 使用举例

❑ 单一表



设计流程

1. 设计对应User的DA类UserDA
2. 设计对应于User的关系数据表UserT
3. 编写User类，UserDA类和辅助类
4. 编写测试类TestUser进行测试

测试内容

1. 编写add()向数据库中增加数据
2. 编写find() 查询某用户记录是否存在
3. 编写delete()删除一条数据
4. 隐藏UserDA类

4. 使用举例

❑ 单一表

```
// User 类
public User(String userID, String name, String pw){
     setName(name);
     setUserID(userID);
     setPassword(pw);
}

public static void initialize(){
     UserDA.initialize();
}

public static void terminate(){
     UserDA.terminate();
}

public void add(){
     UserDA.add(this);
}

public static User find(String userID){
     return UserDA.find(userID);
}
```

4. 使用举例

❑ 单一表

```
// UserDA 类
public class UserDA {
    static User aUser;
    static Connection aConnection;
    static Statement aStatement;

    static String userID;
    static String name;
    static String password;

    public static Connection initialize(){
        try{
            aConnection =
DriverManager.getConnection("jdbc:ucanaccess://"+"C:/Users/weiya/Desktop/Use
r.accdb");
            aStatement = aConnection.createStatement();
        }
        catch(SQLException e){
            System.out.println(e);
        }
        return aConnection;
    }
```

4. 使用举例

```
    public static void terminate(){
        try{
            aStatement.close();
            aConnection.close();
        } catch(SQLException e){
            System.out.println(e);
        }
    }

    public static void add(User aUser){
        name = aUser.getName();
        userID = aUser.getUserID();
        password = aUser.getPassword();

        String sql = "INSERT INTO userT(userID, Uname, password)" +
"VALUES('" + userID +"','"
                + name + "','" + password + "')";

        try {
            int result = aStatement.executeUpdate(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

4. 使用举例

```
    public static User find(String key) {
        aUser = null;

        String sql = "SELECT userID, Uname, password FROM userT WHERE
userID='" + key + "'";

        try{
            ResultSet rs = aStatement.executeQuery(sql);
            boolean gotIt = rs.next();
            if(gotIt){
                userID = rs.getString(1);
                name = rs.getString(2);
                password = rs.getString(3);

                aUser = new User(userID, name, password);
            } else{
                System.out.println("Did not find this record!");
            }
            rs.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return aUser;
    }
```

4. 使用举例

- 1对1

题目

对学生基本信息及住址信息进行管理，例如，对这些信息添加、修改、删除和查询。

设计流程

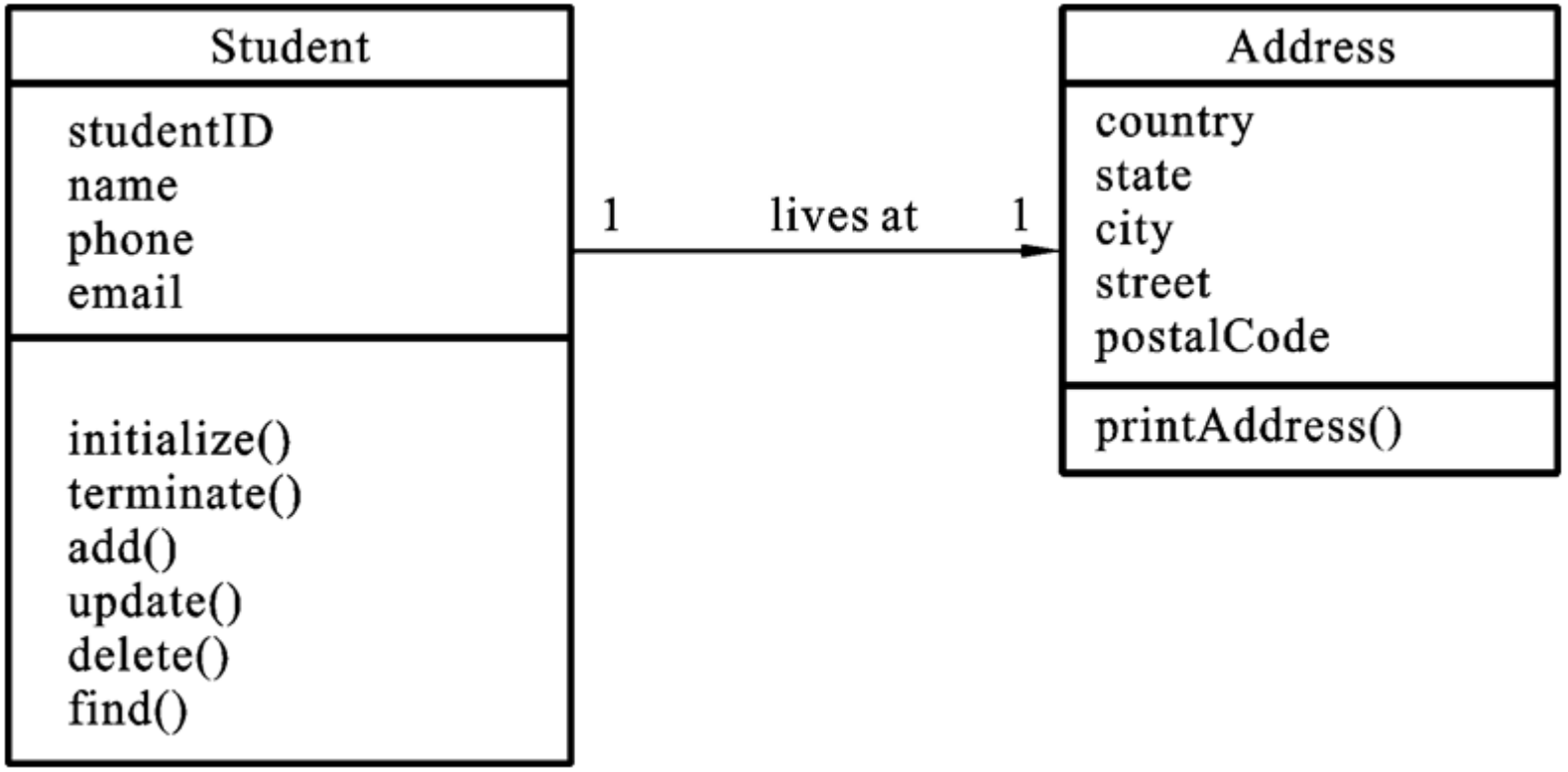
1. 找出问题域类画出类图
2. 据此设计相应的数据表
3. 定义PD classes 和相应的DA class
4. 编写测试程序

测试内容

1. 向数据库中增加一条数据
2. 查询某用户记录是否存在
3. 删除一条数据

4. 使用举例

- 1对1


Student
studentID
name
phone
email
initialize()
terminate()
add()
update()
delete()
find()
1
lives at
1
Address
country
state
city
street
postalCode
printAddress()

```
    public static Student find(String key) {
        aStudent = null;

        String sql = "SELECT studentID, Sname, addID, country  FROM
studentT, addressT WHERE studentID='" + key + "'" + "AND studentT.studentID
= addressT.studentID";

        ResultSet rs;
        try {
            rs = aStatement.executeQuery(sql);
            boolean got = rs.next();
            if(got){
                name = rs.getString(1);
                studentID = rs.getString(2);
                addID = rs.getString(3);
                country = rs.getString(4);

                aAddress = new Address(addID, country);
                aStudent = new Student(studentID, name, aAddress);
            } else{
                System.out.println("Did not find this record!");
            }
            rs.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return aStudent;
    }
```

4. 使用举例

java.sql.PreparedStatement

```
    public static void addNewTask(Task t){
        try {
            pstmt = aConnection.prepareStatement("INSERT INTO task_table
(name, sourceid, destinationid, reward, submittime, duetime, filepath)
VALUES(?,?,?,?,?,?,?)");

            pstmt.setString(1, t.getName());
            pstmt.setString(2, t.getSourceID());
            pstmt.setString(3, t.getDestinationID());
            pstmt.setFloat(4, t.getReward());
            pstmt.setTimestamp(5, new Timestamp(t.getSubmitTime().getTime()));
            pstmt.setTimestamp(6, new Timestamp(t.getDueTime().getTime()));
            pstmt.setString(7, t.getTextFilePath());

            pstmt.executeUpdate();

            pstmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

    }
```

4. 使用举例

❑ 1对多

题目

查询手机联系人的信息，包括姓名和电话号码。

设计流程

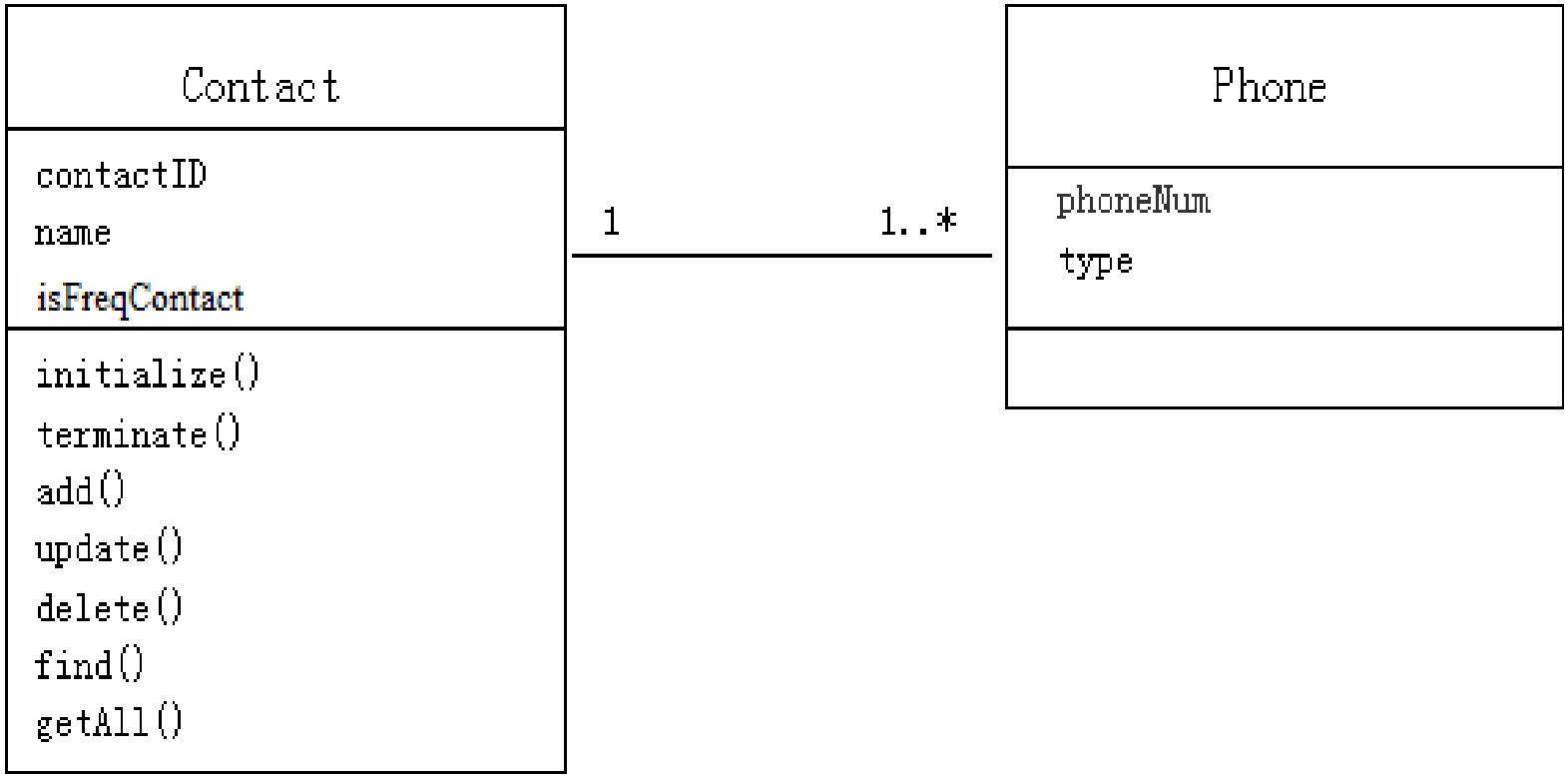
1. 找出问题域类画出类图
2. 据此设计相应的数据表
3. 定义PD classes 和相应的DA class
4. 编写测试程序

测试内容

1. 向数据库中增加一条数据
2. 查询某用户记录是否存在
3. 显示同一联系人时避免多次读取相同信息
4. 删除一条数据

4. 使用举例

- 1对多



如何从1对多的多张表中读取数据并重建对象？

总结

数据库的基本概念

SQL 简介

Java数据库开发：JDBC，PD-DA

掌握关系数据库的概念和使用

如何为问题域类设计数据访问类实现数据库访问

下节预告

代码规范化之路