第10章
数据库应用

目 录

Java数据库编程技术

- 数据库系统概述
- JDBC概述
- JDBC编程步骤

Java数据库编程技术

1. 数据库系统概述

- 数据库系统
- SQL语言
- 常用SQL句型

2. JDBC概述

- JDBC的概述
- JDBC的基本功能
- JDBC API和JDBC Driver API接口

3. JDBC编程步骤

1. 数据库系统概述

数据库系统

- 数据库系统看作是管理系统中大量的、持久的、可靠的、共享的数据的工具，这些数据具有最小冗余度和较高的数据与程序的独立性，而且数据库应该能保持数据的安全性、维护数据的一致性。
- 数据库系统一般由数据库、数据库管理系统、应用系统、数据库管理员和数据库用户所组成。
- 数据模型是数据库系统的核心和基础，主要包括网状模型、层次模型、关系模型等。
- 现在商用的数据库系统几乎都是关系数据库系统。
  - **Oracal、DB2、Microsoft SQL Server、MySQL、FoxPro、Access……**

1. 数据库系统概述

学生表（Student）

S_No	S_Name	S_Sex	S_Age
201201	Edward	M	32
201202	Catty	F	23

课程表（Course）

C_No	C_Name	C_Credit
1001	DataBase	3
1002	DataStructure	2

选课表（SC）

SC_Sno	SC_Cno	SC_Score
201201	1001	98
201201	1002	95
201202	1001	95
201202	1002	86

1. 数据库系统概述

- SQL语言
  - SQL是Structured Query Language的缩写。
  - SQL语言的命令一般分为四类：
    - 查询语言：select…from…where…
    - 操纵语言：insert、update、delete…
    - 定义语言：create、alter、drop
    - 控制语言：grant、revoke、commit、rollback…

1. 数据库系统概述

- 常用SQL句型

数据表创建

create table <表名> (字段1 类型1(长度),字段2 类型2(长度) ……)

[例] 创建学生数据表students，包含学号，姓名，性别，生日和系名信息。

```
create table students
    (sno varchar(10), name varchar(20), sex varchar(5),
       birthday datetime,  department varchar(50))
```

删除数据表

drop table <表名>

[例] 删除数据表students。

```
drop table students
```

1. 数据库系统概述

常用SQL句型

查询语句

select<输出结果列表>from<表>
[where<选择条件>]
[order by<排序条件>]
[group by<分组条件>]

[例] 查询所有同学的学号和姓名。

```
select sno, name from students
```

[例] 按照年龄从大到小的次序列出男同学的学号和姓名。

```
select sno, name from students
      where sex='男' order by birthday
```

[例] 查询姓“赵”的同学的姓名、所属院系和出生年月日。

```
select name, department, birthday from students
where name like '赵%'
```

1. 数据库系统概述

- 常用SQL句型
  - 插入语句

insert into<表名>[(<列表名>)]
VALUES(<对应列的值表>)

[例] 把一位学号为S200、姓名为何十、性别为男、出生年月日为1993年11月19日法学院学生的记录插入到表students中。

```
insert into students
(sno, name, sex, birthday, department)
values('S200','何十','男','1993/11/19','法学院')
```

  - 修改语句

update<表名>set<列>=<值>[,<列>=<值>][where<定位条件>]

[例] 把学号为S200的学生从法学院调到信息学院。

```
update students
set department='信息学院'
where sno='S200'
```

  - 删除语句

delete from<表名>[where<条件>]

[例] 把学号为S200的学生记录删除。

```
delete from students
where sno='S200'
```

2. JDBC概述

JDBC概述

- JDBC(Java DataBase Connectivity)是一套允许Java与SQL数据库对话的程序设计接口，它是用于执行SQL语句的Java API，是Java应用程序连结数据库、存取数据的一种机制，可以为多种关系数据库提供统一的访问，它由一组用Java语言编写的类和接口组成。

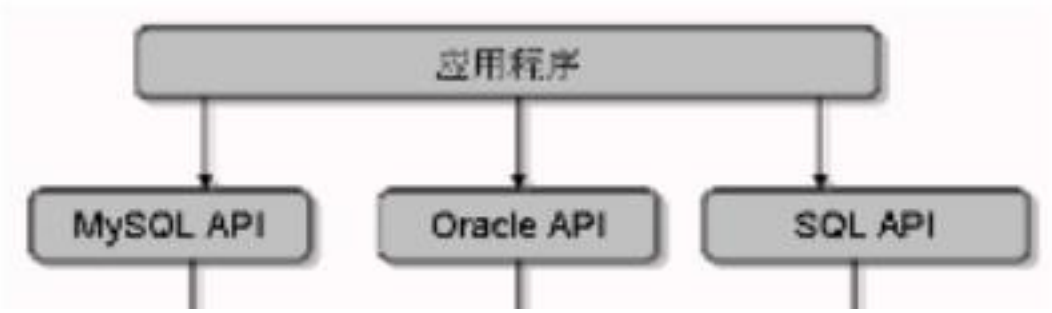
JDBC的基本功能

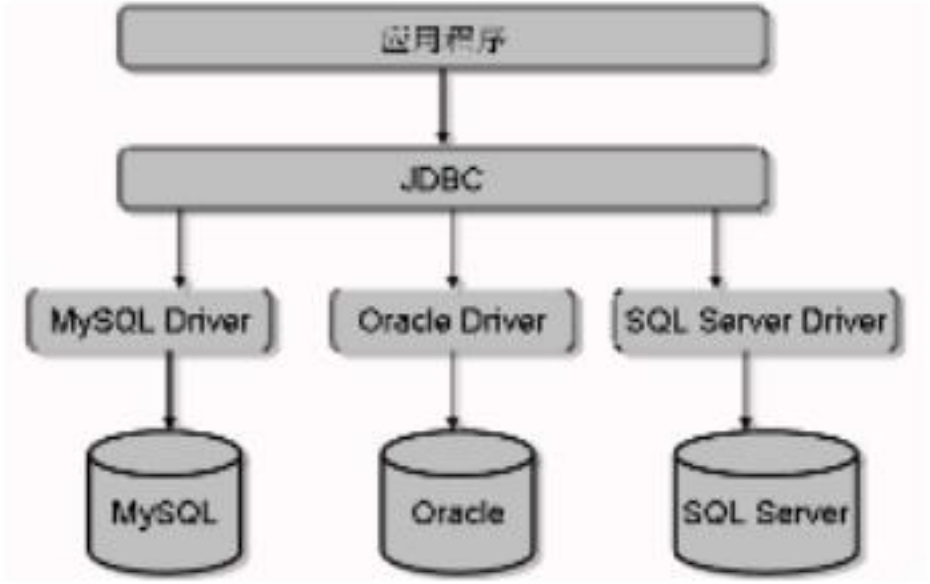
①加载JDBC驱动程序；

②建立与数据库的连接；

③使用SQL语句进行数据库操作并处理结果；

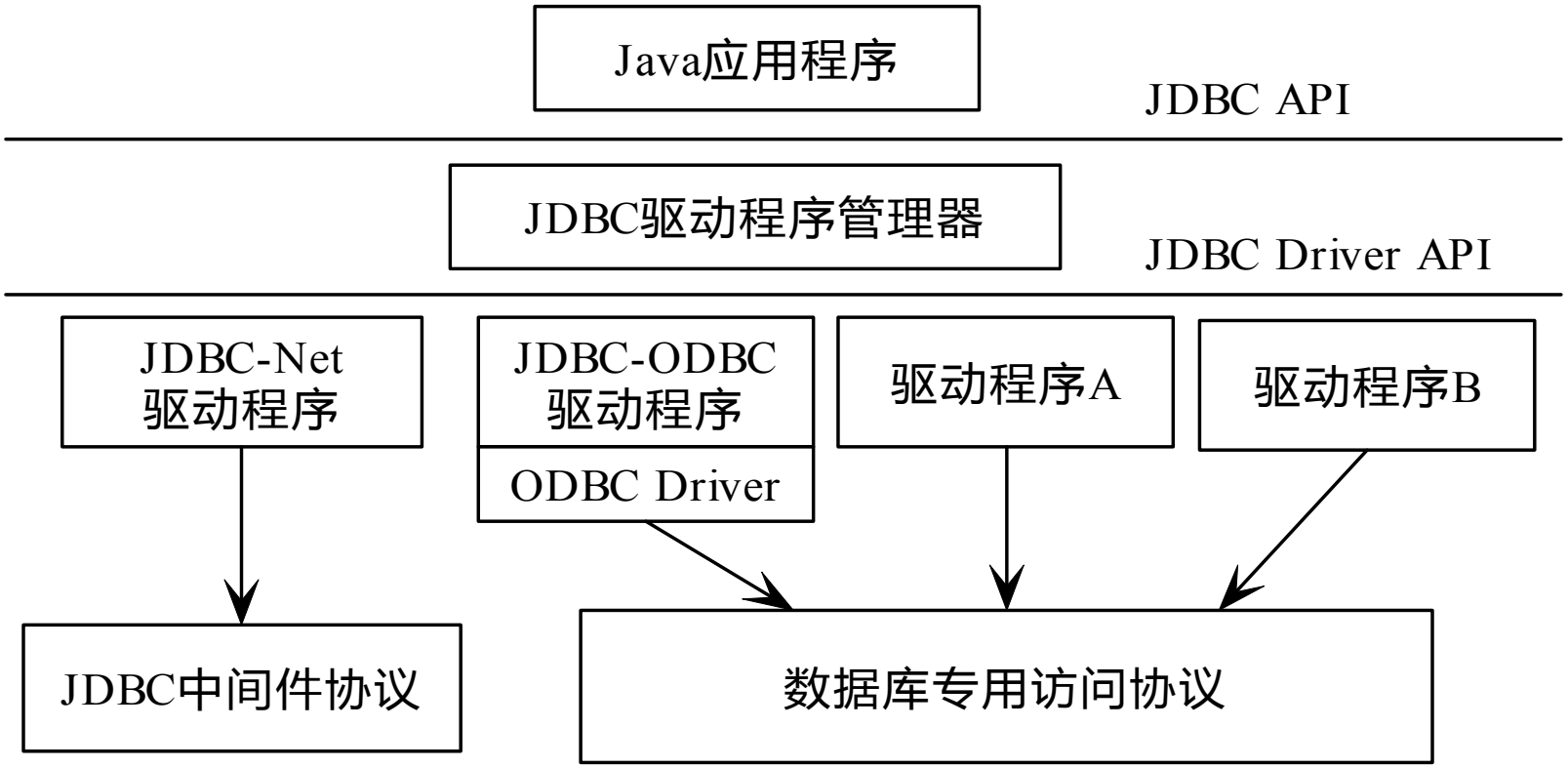
④关闭相关连接。


应用程序
MySQL API
Oracle API
SQL API


应用程序
JDBC
MySQL Driver
Oracle Driver
SQL Server Driver
MySQL
Oracle
SQL Server

2. JDBC概述

- JDBC主要提供两个层次的接口：
  - （一）面向程序开发人员的**JDBC API**（**JDBC应用程序接口**）
  - （二）面向系统底层的**JDBC Driver API**（**JDBC驱动程序接口**）



JDBC结构示意图

2. JDBC概述

- **JDBC API：** 被描述成一组抽象的接口，JDBC的接口和类定义都在包java.sql中，利用这些接口和类可以使应用程序很容易地对某个数据库打开连接、执行SQL语句、并且处理结果。
- java.sql.DriverManager
  - JDBC的管理层，管理JDBC驱动程序。
- java.sql.Connection接口
  - 处理与数据库的连接。
- java.sql.Statement接口
  - 用于执行静态SQL语句并且返回所生成的结果集。
- java.sql.ResultSet接口
  - 提供了对于数据库表的访问，一个ResultSet对象通常由执行一个SQL语句而生成。

java.sql.DriverManager

JDBC的管理层，管理JDBC驱动程序。

- getDriver(String url)：根据指定url定位一个驱动；
- getDrivers()：获得当前调用访问的所有加载的JDBC驱动；
- getConnection()：使用给定的url建立一个数据库连接，并返回一个Connection接口对象；
- registerDriver(java.sql.Driver dirver)：登记给定的驱动；
- setCatalog(String database)：确定目标数据库。

java.sql.DriverManager接口URL列表

- Connection cn;
- MySQL：Class.forName("org.gjt.mm.mysql.Driver");

  cn = DriverManager.getConnection(
  "jdbc:mysql://MyDbComputerNameOrIP:3306/myDatabaseName", sUsr, sPwd);
- PostgreSQL(http： Class.forName("org.postgresql.Driver");

  cn = DriverManager.getConnection(
  "jdbc:postgresql://MyDbComputerNameOrIP/myDatabaseName", sUsr, sPwd);
- Oracle：Class.forName("oracle.jdbc.driver.OracleDriver");

  cn = DriverManager.getConnection(
  "jdbc:oracle:thin:@MyDbComputerNameOrIP:1521:ORCL", sUsr, sPwd);
- Sybase：Class.forName("com.sybase.jdbc2.jdbc.SybDriver");

  cn = DriverManager.getConnection("jdbc:sybase:Tds:MyDbComputerNameOrIP:2638", sUsr, sPwd);
- Microsoft SQLServer：Class.forName("net.sourceforge.jtds.jdbc.Driver");

  cn = DriverManager.getConnection(
  "jdbc:jtds:sqlserver://MyDbComputerNameOrIP:1433/master", sUsr, sPwd);
- Java DB: Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

  cn = DriverManager.getConnection("jdbc:derby:helloDB;create=true", sUsr, sPwd);
- ODBC：Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

  cn = DriverManager.getConnection("jdbc:odbc:" + sDsn, sUsr, sPwd);
- DB2：Class.forName("com.ibm.db2.jdbc.net.DB2Driver");

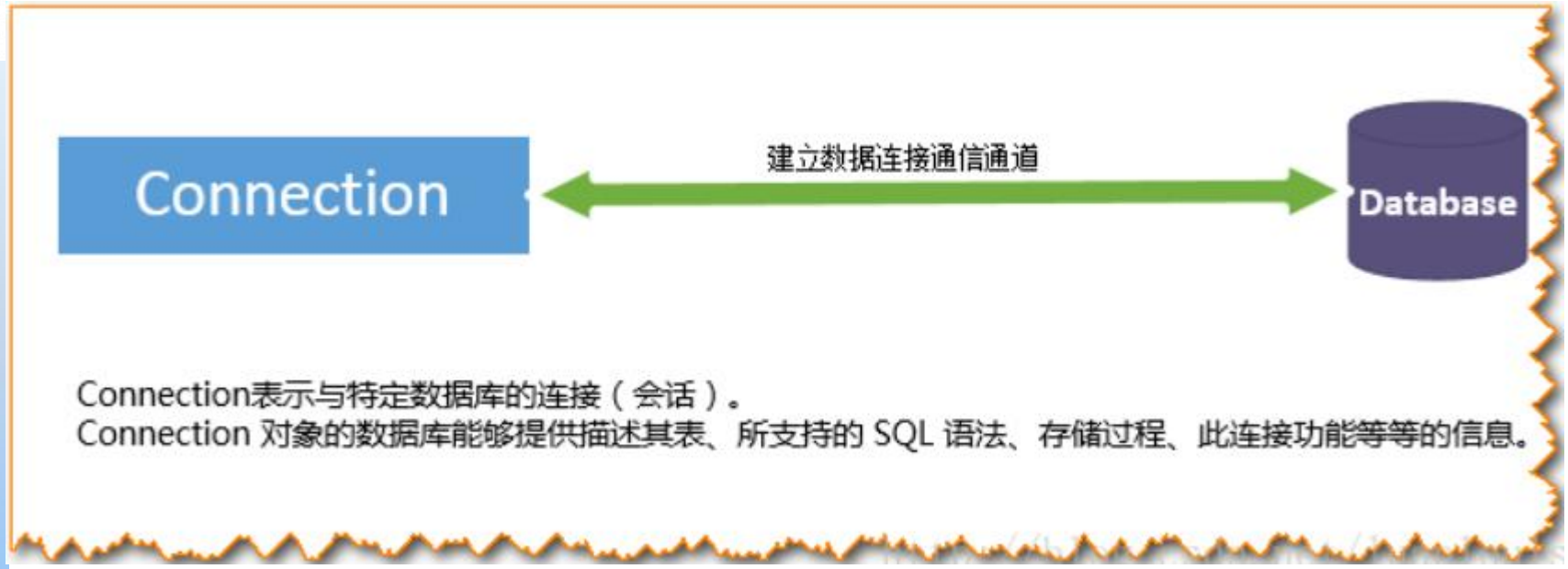
  cn = DriverManager.getConnection("jdbc:db2://192.9.200.108:6789/SAMPLE", sUsr, sPwd);

https://blog.csdn.net/fengyuzhengfan/article/details/38086743 --Class

java.sql.Connection接口

处理与数据库的连接。

- **createStatement()**：在本连接上生成一个Statement对象，该对象可对本连接的特定数据库发送SQL语句；
- **setAutoCommit(Boolean autoCommit)**：设置是否自动提交；
- **getAutoCommit()**：获得自动提交状态；
- **commit()**：提交数据库上当前的所有待提交的事务；
- **close()**：关闭当前的JDBC数据库连接。


Connection
建立数据连接通信通道
Database
Connection表示与特定数据库的连接（会话）。
Connection 对象的数据库能够提供描述其表、所支持的 SQL 语法、存储过程、此连接功能等等的信息。

❖ java.sql.Statement接口

用于执行静态SQL语句并且返回所生成的结果集

```
try{
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
Connection cn = DriverManager.getConnection("jdbc:derby:he
cn.createStatement().e
}
catch(Exception e){
    e.printStackTrace
}
```

equals(Object obj) : boolean - Object
execute(String sql) : boolean - Statement
execute(String sql, int autoGeneratedKeys) : boolean
execute(String sql, int[] columnIndexes) : boolean - St
execute(String sql, String[] columnNames) : boolean -
executeBatch() : int[] - Statement
executeQuery(String sql) : ResultSet - Statement
executeUpdate(String sql) : int - Statement
executeUpdate(String sql, int autoGeneratedKeys) : ir
executeUpdate(String sql, int[] columnIndexes) : int -
executeUpdate(String sql, String[] columnNames) : in

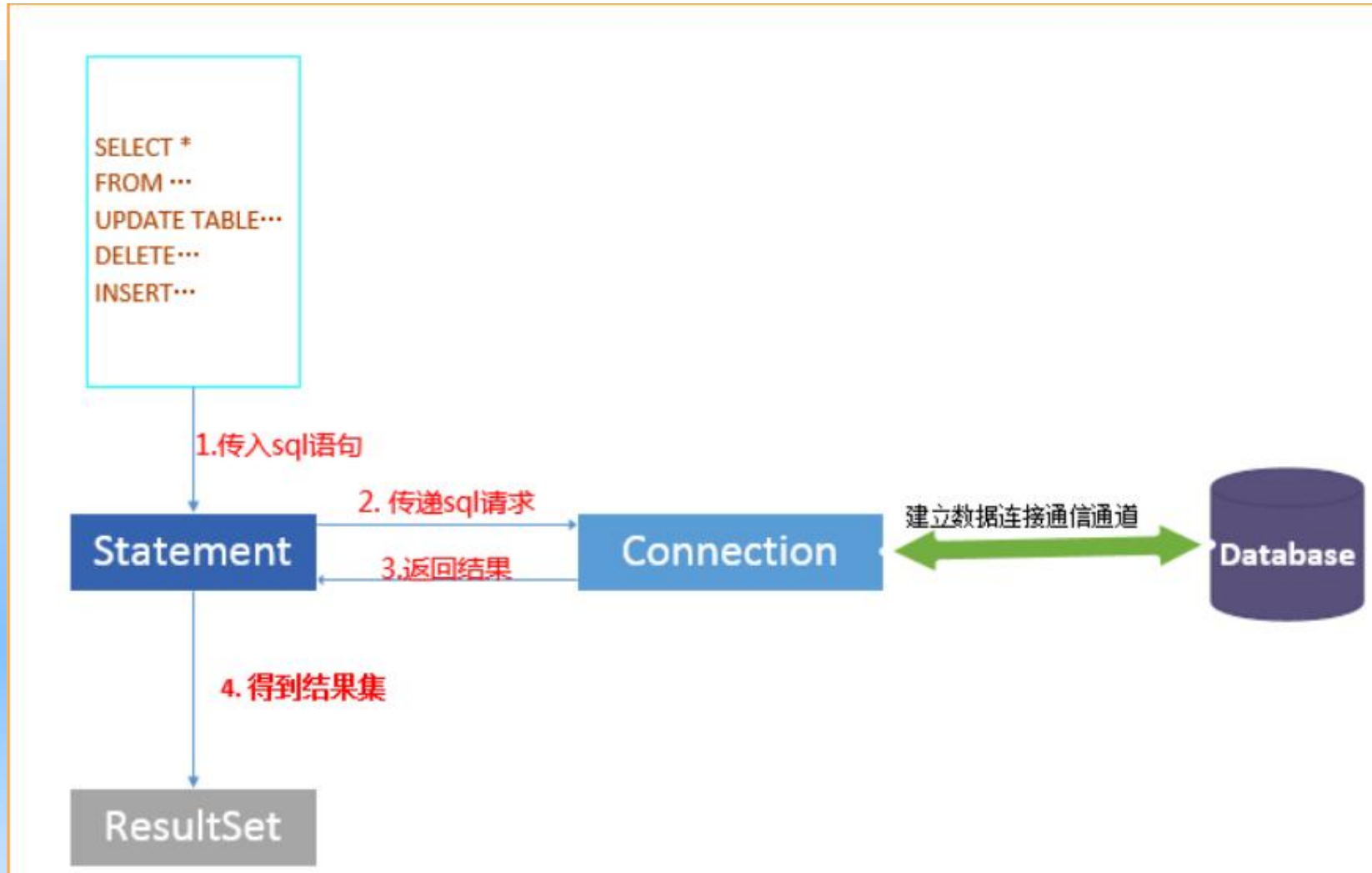
Press 'Alt+/' to show Template Proposals

avadoc | Declaration | Console

wtest [Java Application] D:\Program F

va.net.URLClassLoader$1.run(Unk

va.net.URLClassLoader$1.run(Unk



Statement的工作模式

java.sql.ResultSet接口

提供了对于数据库表的访问，一个**ResultSet**对象通常由执行一个**SQL**语句而生成。

- **next()**：将数据指针往下移动一行，如果成功返回**true**；否则返回**false**；第一条指向**null**，因为正常执行时第一句是**.next()**.

```
try{
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
Connection cn = DriverManager.getConnection("jdbc:derby:helloDB;create=true
cn.createStatement().executeQuery("").
```

Moves the cursor to the given row number in this `ResultSet` object.

If the row number is positive, the cursor moves to the given row number with respect to the beginning of the result set. The first row is row 1, the second is row 2, and so on.

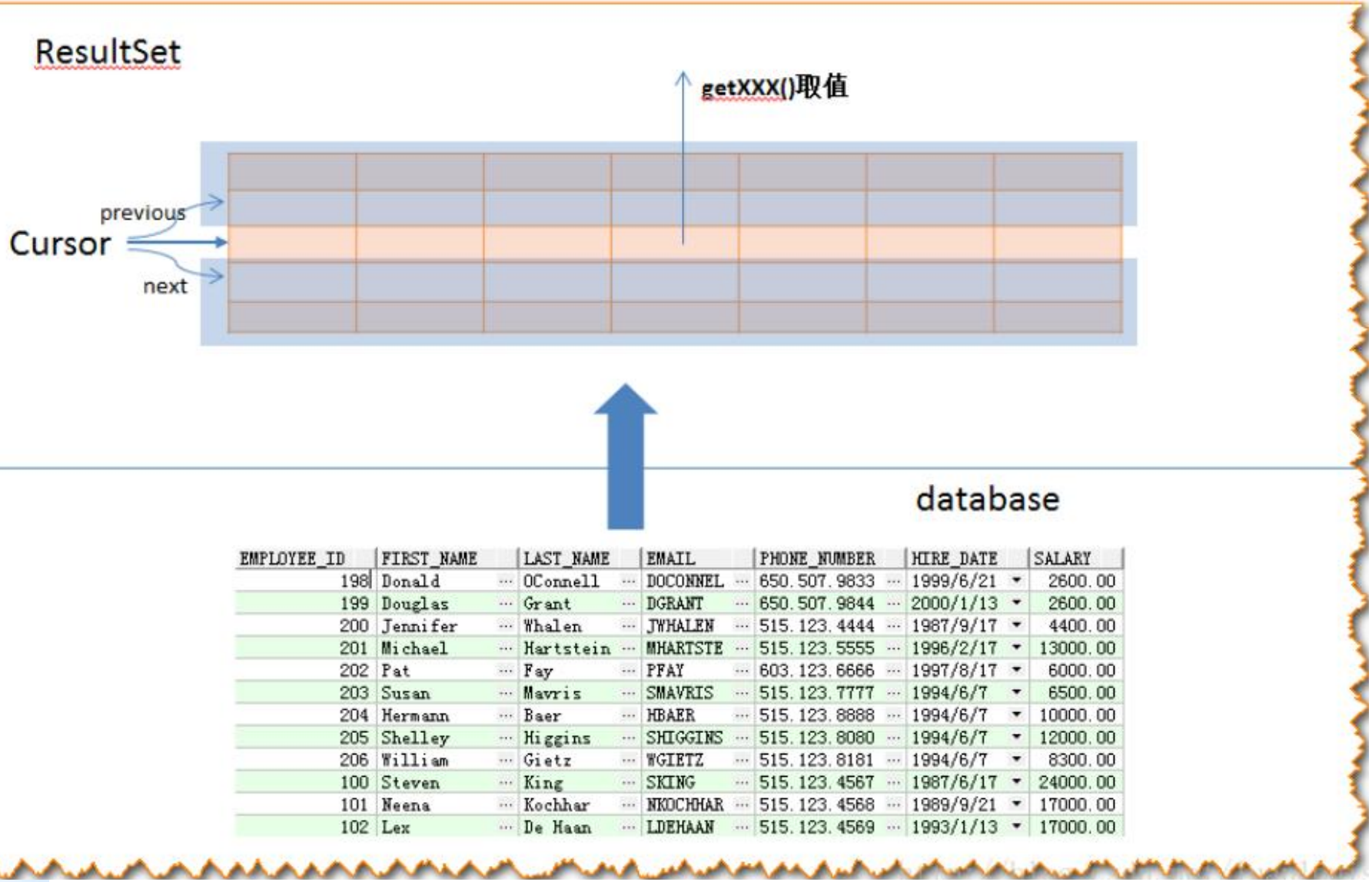
If the given row number is negative, the cursor moves to an absolute row position with respect to the end of the result set. For example, calling the method `absolute(-1)` positions the cursor on the last row; calling the method `absolute(-2)` moves the cursor to the next-to-last row, and so on.

If the row number specified is zero, the cursor is moved to

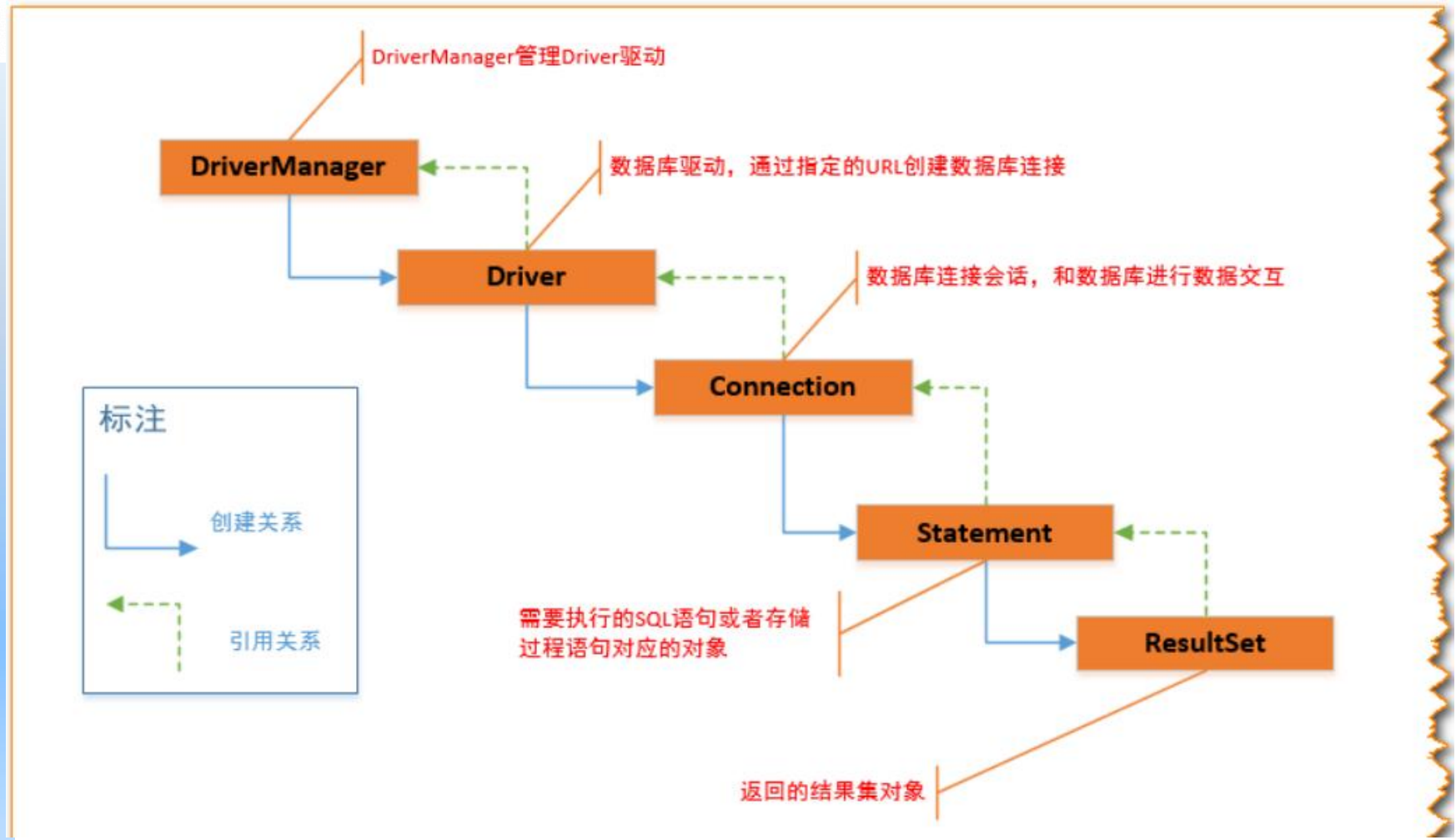
Press 'Tab' from proposal table or click for focus

- absolute(int row) : boolean - ResultSet
- afterLast() : void - ResultSet
- beforeFirst() : void - ResultSet
- cancelRowUpdates() : void - ResultSet
- clearWarnings() : void - ResultSet
- close() : void - ResultSet
- deleteRow() : void - ResultSet
- equals(Object obj) : boolean - Object
- findColumn(String columnLabel) : int - ResultSet
- first() : boolean - ResultSet
- getArray(int columnIndex) : Array - ResultSet

Press 'Alt+/' to show Template Proposals

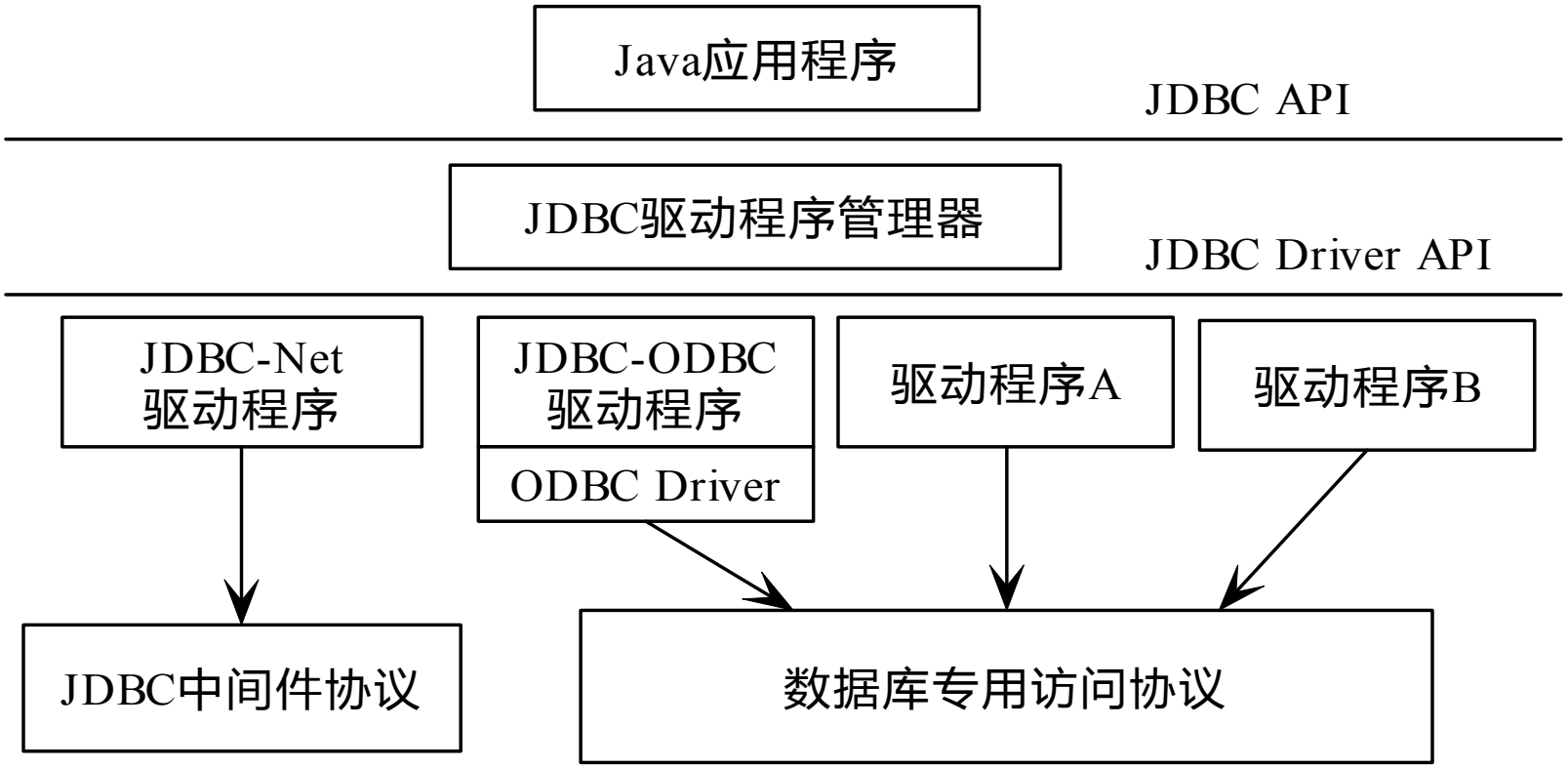


EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	SALARY
198	Donald	OConnell	DOCONNEL	650.507.9833	1999/6/21	2600.00
199	Douglas	Grant	DGRANT	650.507.9844	2000/1/13	2600.00
200	Jennifer	Whalen	JWHALEN	515.123.4444	1987/9/17	4400.00
201	Michael	Hartstein	MHARTSTE	515.123.5555	1996/2/17	13000.00
202	Pat	Fay	PFAY	603.123.6666	1997/8/17	6000.00
203	Susan	Mavris	SMAVRIS	515.123.7777	1994/6/7	6500.00
204	Hermann	Baer	HBAER	515.123.8888	1994/6/7	10000.00
205	Shelley	Higgins	SHIGGINS	515.123.8080	1994/6/7	12000.00
206	William	Gietz	WGIETZ	515.123.8181	1994/6/7	8300.00
100	Steven	King	SKING	515.123.4567	1987/6/17	24000.00
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989/9/21	17000.00
102	Lex	De Haan	LDEHAAN	515.123.4569	1993/1/13	17000.00


DriverManager管理Driver驱动
DriverManager
数据库驱动，通过指定的URL创建数据库连接
Driver
数据库连接会话，和数据库进行数据交互
Connection
标注
创建关系
引用关系
Statement
需要执行的SQL语句或者存储过程语句对应的对象
ResultSet
返回的结果集对象

2. JDBC概述

- JDBC主要提供两个层次的接口：
  - （一）面向程序开发人员的**JDBC API**（**JDBC应用程序接口**）
  - （二）面向系统底层的**JDBC Driver API**（**JDBC驱动程序接口**）



JDBC结构示意图

2. JDBC概述

- **JDBC Driver API：** 是面向驱动程序开发的编程接口。根据其运行条件的不同，常见的JDBC驱动程序主要有四种类型：

- **JDBC-ODBC桥加ODBC驱动程序（JDBC-ODBC bridge plus ODBC driver）**

  JDBC-ODBC桥驱动程序，将JDBC调用转换为ODBC的调用。

- **本地API、部分是Java的驱动程序（Native-API partly-Java driver）**

  该类型的驱动程序用于将JDBC的调用转换成主流数据库API的本机调用。

- **JDBC-Net的纯Java驱动程序（JDBC-Net pure Java driver）**

  这种类型的驱动程序将JDBC调用转换成与数据库无关的网络访问协议，利用中间件将客户端连接到不同类型的数据库系统。

- **本地协议的纯Java驱动程序（Native-protocol pure Java driver）**

  这种类型的驱动程序将JDBC调用直接转化为某种特定数据库的专用的网络访问协议，可以直接从客户机来访问数据库系统。

3. JDBC编程步骤

①导入java.sql.*包： import

②加载JDBC驱动程序： Class.forName()

③定义数据库的URL： String url="jdbc:derby:helloDB "

④连接数据库： getConnection(url)

⑤建立SQL语句对象： createStatement()

⑥执行SQL语句： executeQuery()、execute()、executeUpdate()

⑦处理结果集： next()、 previous()……

⑧关闭连接： commit()、 close()

示例：Java DB数据库操作程序

- Java DB
  - 安装了**JDK 6** 以后版本的会发现，除了传统的 **bin**、**jre** 等目录，**JDK 6** 新增了一个名为 **db** 的目录。
  - 一个纯 **Java** 实现、开源的数据库管理系统（**DBMS**），源于 **Apache** 软件基金会（**ASF**）名下的项目 **Derby**。**Java** 程序员不再需要耗费大量精力安装和配置数据库，就能进行安全、易用、标准、并且免费的数据库编程。
  - **Java DB** 是完全事务型、安全且基于标准的数据库服务器，完全支持 **SQL**、**JDBC API** 和 **Java EE** 技术。

示例：Java DB数据库操作程序

以JavaDB作为数据库操作对象，编程显示hellotable表中所有学生的名字和成绩信息：

（1） 在 DBMS 中创建了一个名为 helloDB 的数据库；

（2） 创建了一张数据表，取名为 hellotable，同时该表具有name和score 属性；

（3） 向表内插入了两条数据；

（4） 然后，查询数据并将结果打印在控制台上；

（5） 最后，删除表和数据库，释放资源。

示例：Java DB数据库操作程序

```
import java.sql.*;
public class HelloJavaDB {
    public static void main(String[] args) {
        try { // load the driver
            Class.forName(“org.apache.derby.jdbc.EmbeddedDriver”).newInstance();
            System.out.println(“Load the embedded driver”);
            Connection conn = null;

           //create and connect the database named helloDB
            conn=DriverManager.getConnection(“jdbc:derby:helloDB;create=true”,
“username”,”password”);
            System.out.println("create and connect to helloDB");

            // create a table and insert two records
            Statement s = conn.createStatement();
            s.execute("create table hellotable(name varchar(40), score int)");
            System.out.println("Created table hellotable");
            s.execute("insert into hellotable values('Ruth Cao', 86)");
            s.execute("insert into hellotable values ('Flora Shi', 92)");
```

第1步

第2步

第3步

第4步

第5步

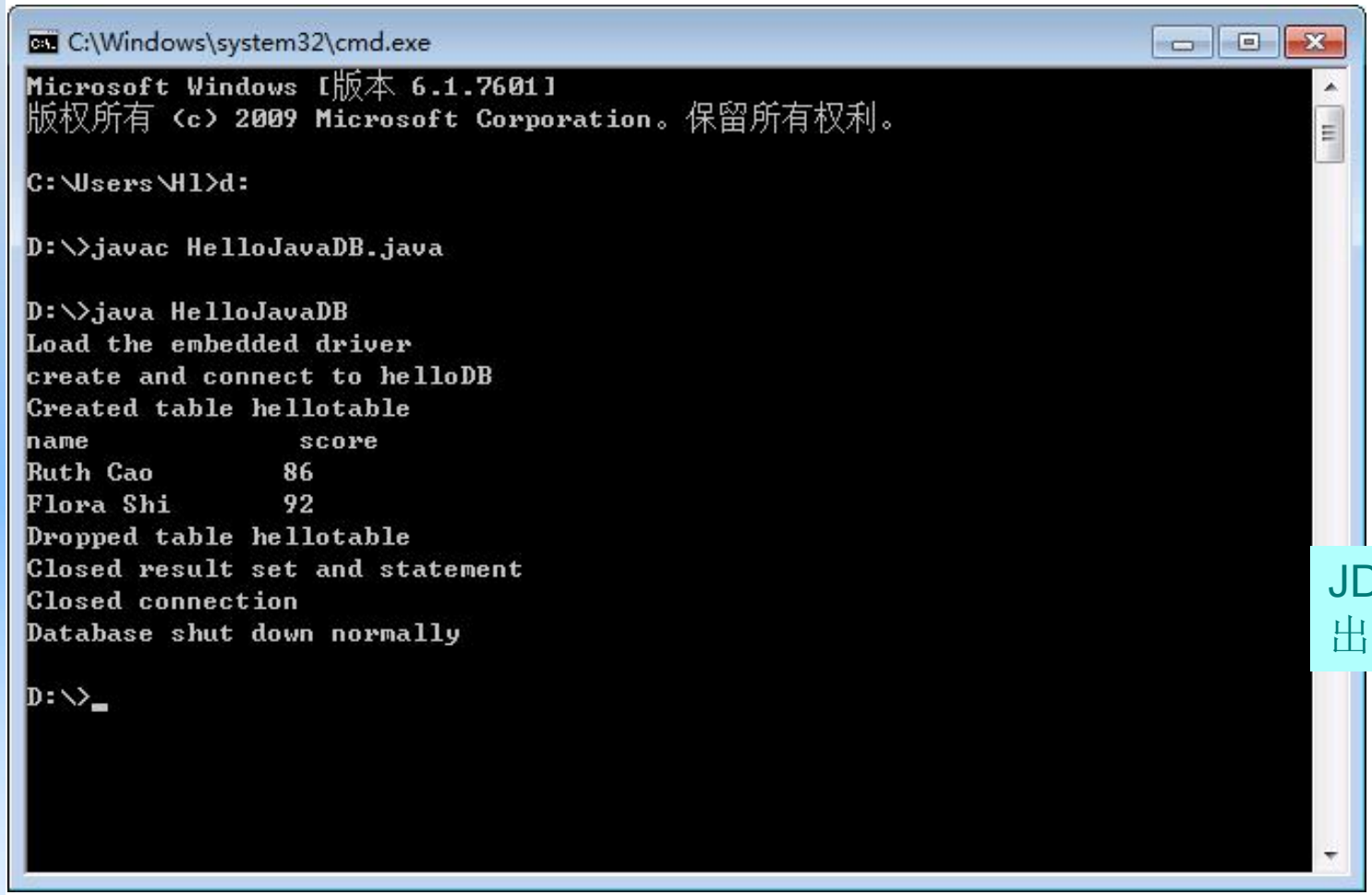
第6步

```
            ResultSet rs = s.executeQuery(
                "SELECT name, score FROM hellotable ORDER BY score");
            System.out.println("name\t\t score ");
            while(rs.next()) {
                StringBuilder builder = new StringBuilder(rs.getString(1));
                builder.append("\t");
                builder.append(rs.getInt(2));
                System.out.println(builder.toString());
            }
            // delete the table
            s.execute("drop table hellotable");
            System.out.println("Dropped table hellotable");

            rs.close();
            s.close();
            System.out.println("Closed result set and statement");

            conn.close();
            System.out.println(“Closed connection");

            try { // perform a clean shutdown
                DriverManager.getConnection("jdbc:derby:;shutdown=true");
            } catch (SQLException se) {
                System.out.println("Database shut down normally");
            }
        } catch (Throwable e) {
            // handle the exception
        }
        System.out.println("SimpleApp finished");
    }
}
```

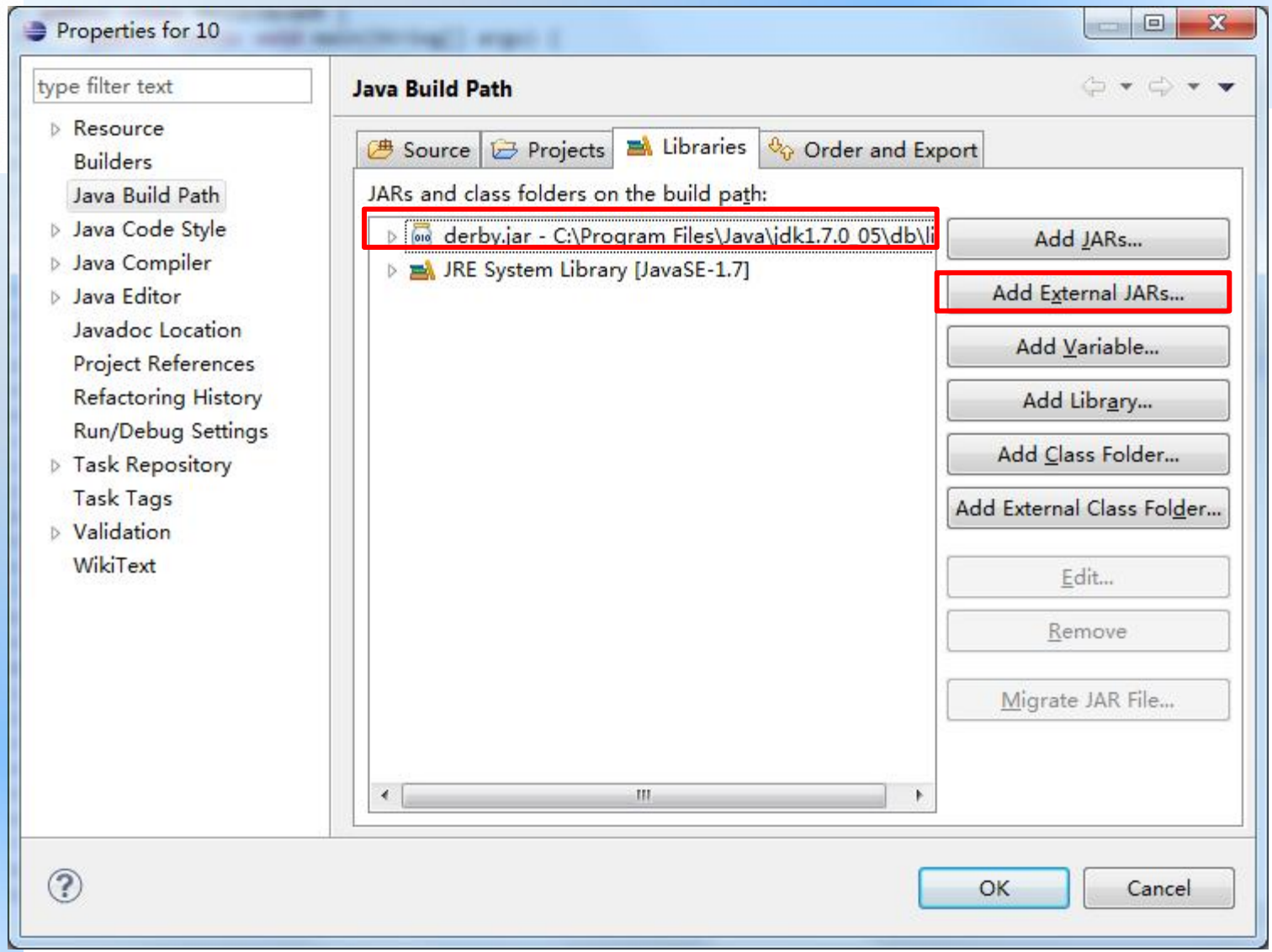
第7步

第8步

需要配置环境变量classpath为%JAVA_HOME%\db\lib\derby.jar

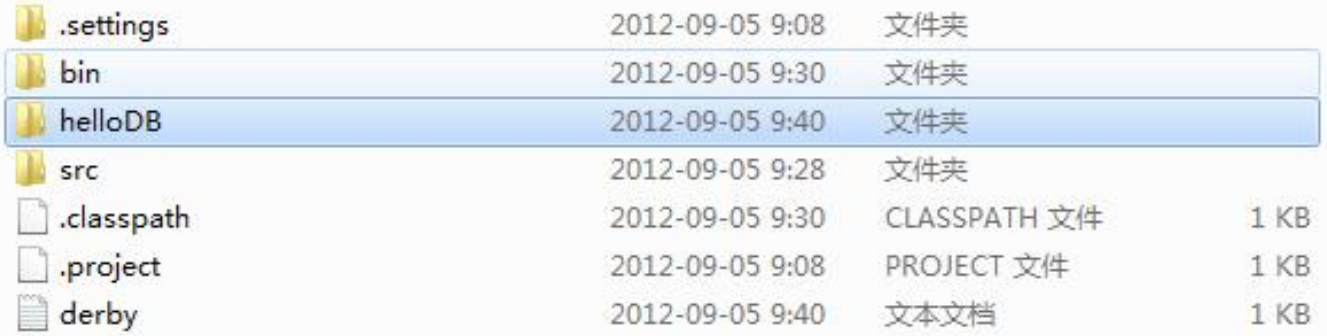


JDK下输出结果



Eclipse配置：右击工程名，选择Build Path/Configure Build Path…

程序运行完成后，可以在运行的目录看到一个helloDb的文件夹，其中就是此次创建的数据库。




```
Load the embedded driver
create and connect to helloDB
Created table hellotable
name                score
Ruth Cao           86
Flora Shi          92
Dropped table hellotable
Closed result set and statement
Closed connection
Database shut down normally
SimpleApp finished
```

Eclipse中输出结果

对象关系映射ORM

- 对象关系映射（Object Relational Mapping，简称ORM）模式是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术。简单的说，ORM是通过使用描述对象和数据库之间映射的元数据，将程序中的对象自动持久化到关系数据库中。再直白点说，就是把数据库中的表映射成类，把查询结果赋给Java对象。

- 流行的ORM框架
  - 传统的Entity EJB
  - Hibernate
  - IBATIS
  - Oracle的TopLink

Hibernate

- Gavin King
- Cirrus Technologies
- 用了EJB的Entity bean 1.1时，我总觉得我浪费了好多时间在处理Entity Bean的体系架构上，却没有花时间在核心业务逻辑的开发上，而且CMP给我们的限制太多了

Hibernate

- 可以应用在任何使用JDBC的场合,既可以在客户端程序使用,也可以在Servlet/jsp的web应用中使用,最具革命意义的是,Hibernate可以在应用EJB的j2ee框架中取代CMP,完成数据持久化的重任.

小结

- 数据库编程
  - JDBC API接口
  - 编程的8个步骤