
# Java 程序设计

## 第15章 数据库

田英鑫 tyx@hit.edu.cn
哈尔滨工业大学软件学院

# 第15章 数据库

## & 本章导读

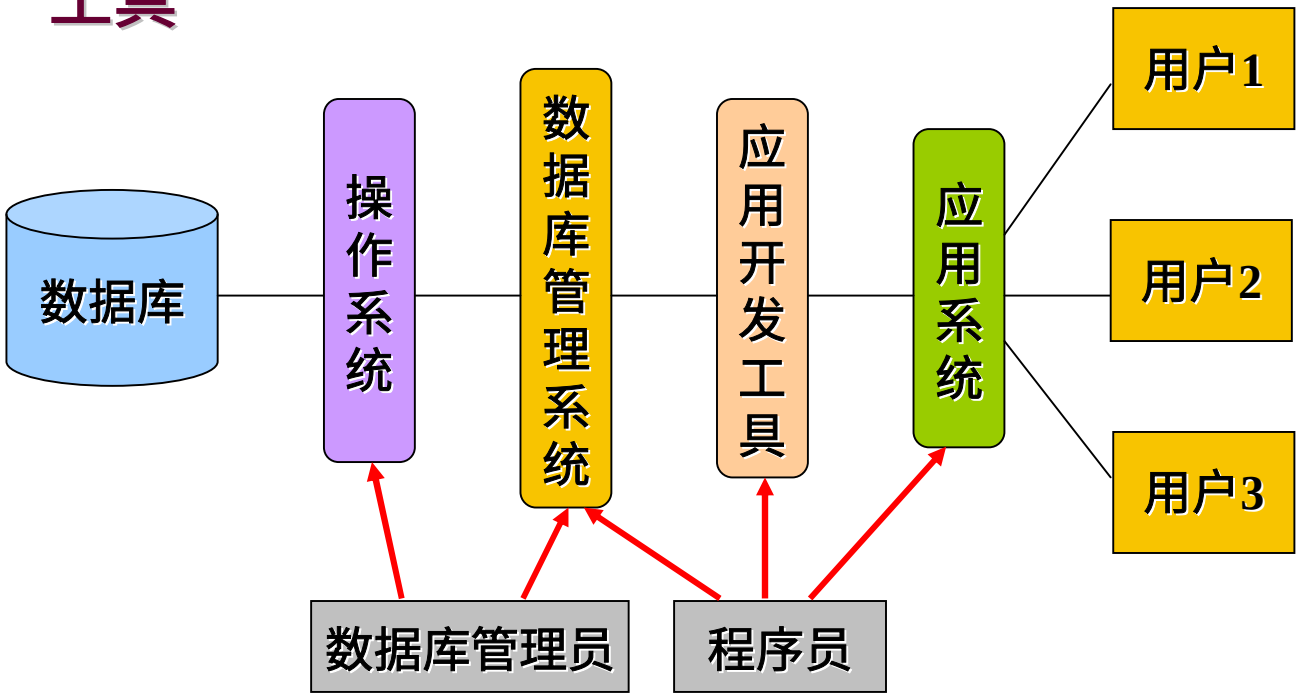
- 15.1 数据库简介
- 15.2 数据库管理系统简介
- 15.3 结构化查询语言 SQL
- 15.4 JDBC 简介
- 15.5 通过 ODBC 连接数据库
- 15.6 通过专用驱动连接数据库
- 15.7 预处理命令
- 15.8 调用存储过程


# 15.1 数据库简介

- 什么是数据库？
  - 数据库是为存储、组织和管理有规则的计算机数据提供的一种技术
  - 其主要目标是解决数据管理中数据的获取、编码、组织、存储、访问和处理等问题
  - 通常我们使用的是基于SQL语言的关系数据库

# 15.2 数据库管理系统简介

- 什么是数据库管理系统？
  - 数据库管理系统（DBMS）是一种操纵和管理数据库的软件，是用于建立、使用和维护数据库的工具



# 15.2 数据库管理系统简介

- 常用关系数据库系统
  - 大型数据库
    - Oracle
    - IBM DB2
    - Sybase
    - Microsoft SQL Server
    - Informix
  - 中小型数据库
    - MySQL、Access
    - FoxPro、FoxBase、dBase

# 15.3 结构化查询语言 SQL

- 数据定义语言（DDL）
  - create、alter、drop
- 数据控制语言（DCL）
  - grant、commit、rollback
- 数据操纵语言（DML）
  - select
  - insert、delete、update

# 15.3 结构化查询语言 SQL

- 学生表结构
  - id（学号）、字符串类型、主键
  - name（姓名）、字符串类型
  - age（年龄）、整型

| id | name | age |
|---|---|---|
| 1001 | 张三 | 18 |
| 1002 | 李四 | 19 |
| 1003 | 小明 | 20 |

# 15.3 结构化查询语言 SQL

- select 语句
  - select * from student
    - 查询student表中所有学生的所有信息
  - select * from student where id = '1001'
    - 查询student表中学号为“1001”的学生的所有信息
  - select id,name from student where avgscore >= 60 order by id
    - 查询student表中平均分及格的所有学生的学号和姓名，并按学号从小到达排序

# 15.3 结构化查询语言 SQL

- insert 语句
  - insert into student values('1001','张三',18)
    - 向student表中插入一条记录
    - values必须提供student表的所有字段的数据
  - insert into student(id,name) values('1001','张三')
    - 向student表中插入一条记录
    - values只须提供id和name字段的数据即可，但age字段在定义时必须可空

# 15.3 结构化查询语言 SQL

- delete 语句
  - delete from student
    - 删除student表中的所有数据
    - 慎用
  - delete from student where id = '1001'
    - 删除student表中学号为1001的学生

# 15.3 结构化查询语言 SQL

- update 语句
  - update student set name = '李四',age=19
    - 将student表中的所有学生的姓名更新为“李四”，年龄更新为19
    - 慎用
  - update student set name = '李四',age=19 where id = '1001'
    - 将student表中学号为“1001”的学生的姓名更新为“李四”，年龄更新为19


# 15.4 JDBC 简介

- 什么是 JDBC ？
  - JDBC是Java语言连接数据库的驱动程序，其提供了一套标准的访问数据库的API，Java语言通过JDBC统一访问各种类型的数据库
  - 基于JDBC的数据库程序设计步骤
    - 连接数据库
    - 执行SQL语句操纵或查询数据
    - 关闭数据库连接
  - JDBC连接数据库的方式
    - 通过JDBC提供的ODBC驱动连接ODBC访问数据库
    - 通过数据库厂商或第三方厂商提供的JDBC驱动访问数据库（推荐）

# 15.5 通过 ODBC 连接数据库

- 通过 ODBC 连接数据库
  - ODBC已经得到了广泛的支持，JDBC可以借助ODBC访问各种数据库
  - 需要如下三步准备工作：
    - 第一步：导入必要的package
      - import java.sql.*;
    - 第二步：加载适当的驱动程序
      - Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    - 第三步：与数据库创建连接
      - Connection con=DriverManager.getConnection("jdbc:odbc:dsname");

# 15.5 通过 ODBC 连接数据库

- 建立ODBC数据源
  - 创建数据库
  - 在数据库中创建表：student
    - id varchar(10)
    - name varchar(10)
    - age int
  - 创建ODBC数据源
    - 控制面板 －> 管理工具 －> 数据源（ODBC）

# 15.5 通过 ODBC 连接数据库

- 连接数据库
  - Connection是用来连接数据库的接口
  - 构造Connection对象
    - Connection con = DriverManager.getConnection("jdbc:odbc:dsName"); //把dsName换成你建立的ODBC数据源名称
  - Connection的方法
    - public Statement createStatement()
      - 返回Statement对象
    - public void close()
      - 关闭连接
    - public boolean isClosed()
    - isReadOnly()、setReadOnly()

# 15.5 通过 ODBC 连接数据库

- 执行 SQL 语句
  - Statement是用来执行SQL语句的接口
  - 构造Statement对象
    - Statement stmt = con.createStatement();
  - Statement的方法
    - public ResultSet executeQuery(String sql)
      - 执行查询（SELECT）指令，返回产生的ResultSet
    - public int executeUpdate(String sql)
      - 执行INSERT、UPDATE、DELETE指令，返回影响到的记录数
    - public ResultSet getResultSet()
      - 返回Statement最近产生的ResultSet
    - close、setMaxRows和getMaxRows

# 15.5 通过 ODBC 连接数据库

- 处理结果集
  - ResultSet是用来接收查询返回结果集的接口
  - 构造ResultSet对象
    - ResultSet rs = stmt.executeQuery("select语句");
  - ResultSet的方法
    - public void close()
    - public XXX getXXX(int index)系列
    - public XXX getXXX(String columnName)系列
    - public boolean next()
    - ……

# 15.5 通过 ODBC 连接数据库

- 示例：查询学生信息
  - 在这个例子中，查询student表中的所有学生信息记录，在控制台上打印输出

参见例:TestDB.java

# 15.6 通过专用驱动连接数据库

- 使用数据库驱动连接数据库（需下载驱动）
  - Oracle
    - oracle.jdbc.driver.OracleDriver
    - jdbc:oracle:thin:@127.0.0.1:1521:dbname
  - SQL Server
    - com.microsoft.jdbc.sqlserver.SQLServerDriver
    - jdbc:sqlserver://127.0.0.1:1433;DatabaseName=dbname
  - MySQL
    - com.mysql.jdbc.Driver
    - jdbc:mysql://127.0.0.1:3306/dbname

# 15.7 预处理命令

- 使用 PreparedStatement 提高代码效率

```
Statement stmt = con.createStatement();
String sql = "UPDATE student SET age = 25 WHERE id = '1001'";
stmt.executeUpdate(sql);
```

```
PreparedStatement stmt
    = con.prepareStatement("UPDATE student SET age = ? WHERE id = ?");
stmt.setInt(1, 25);
stmt.setString(2, "1001");
stmt.executeUpdate();
```

# 15.8 调用存储过程

## Java 调用不带返回值的存储过程（Oracle）

```
CREATE OR REPLACE PROCEDURE AddStudent (
    p_id IN VARCHAR2,
    p_name IN VARCHAR2,
    p_age IN INTEGER )  AS
BEGIN
   INSERT INTO student (id, name, age) VALUES (p_id, p_name, p_age);
END AddStudent;
```

```
CallableStatement stmt = con.prepareCall("{ call SCOTT.AddStudent(?,?,?) }");
stmt.setString(1, "1001");
stmt.setString(2, "张三");
stmt.setInt(3, 18);
stmt.execute();
```

# 15.8 调用存储过程

- Java 调用带返回值的存储过程（Oracle）

```
CREATE OR REPLACE PROCEDURE GetName (
   p_id IN VARCHAR2,
   p_name OUT VARCHAR2 )  AS
BEGIN
   SELECT name INTO p_name FROM student WHERE id = p_id ;
END GetName;
```

```
CallableStatement stmt = con.prepareCall("{ call SCOTT.GetName(?,?) }");
stmt.setString(1, "1001");
stmt.registerOutParameter(2, Types.VARCHAR);
stmt.execute();
String name = stmt.getString(2);
```


# *Any Question?*