

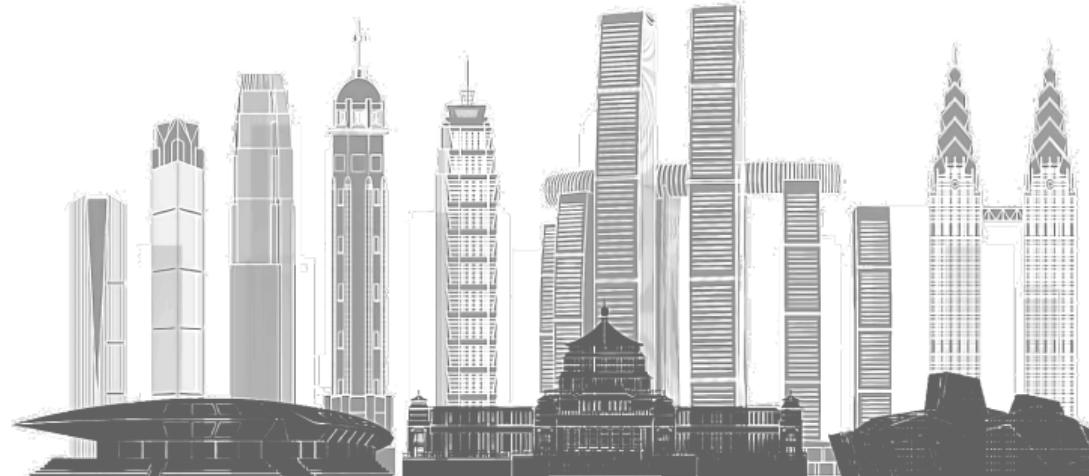


# 数据库系统概论

第 8 章 数据库编程

黄俊杰

2023 Fall





# Table of Contents

## 1 概述

- ▶ 概述
- ▶ 过程化 SQL
- ▶ JDBC 编程
- ▶ 基于 MVC 框架的数据库应用开发
- ▶ 本章小结



## 1 概述

### Section 1.1

## 1.1 SQL 语言表达能力的限制



# 【任务 1】打印课程的所有先修课

## 1 概述

### ■ 【任务 1】打印“数据库系统概论”课程的所有先修课信息

课程号Cno	课程名Cname	学分Ccredit	先修课程Cpno
81001	程序设计基础与C语言	4	NULL
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python语言	3	81002
81007	离散数学	4	NULL
81008	大数据技术概论	4	81003

Course表



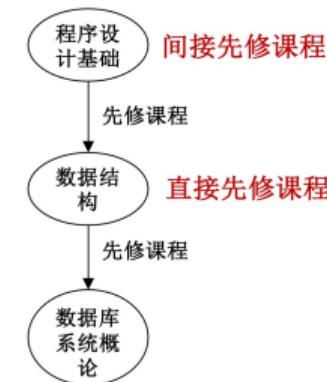
# 【任务 1】打印课程的所有先修课

## 1 概述

### ■ 【任务 1】打印“数据库系统概论”课程的所有先修课信息

Course表

课程号Cno	课程名Cname	学分Ccredit	先修课程Cpno
81001	程序设计基础与C语言	4	NULL
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python语言	3	81002
81007	离散数学	4	NULL
81008	大数据技术概论	4	81003





# 【任务 1】打印课程的所有先修课

## 1 概述

### ■ 任务分析：SQL 如何表达

- 难点：课程可能同时存在直接先修课和间接先修课
- 情况一：如何查询直接先修课：自身连接查询
- 情况二：如何查询间接先修课：递归查询（无法表达）



# 直接先修课的 SQL 语句表达

## 1 概述

### ■ 如何查询直接先修课：自连接

- ① 步骤 1：找出“数据库系统概论”课程的全部直接先修课：记为 L[1]；如果 L[1] 为空，则任务一结束

```
1 SELECT B.Cname  
2 FROM Course A, Course B  
3 WHERE A.Cname = '数据库系统概论'  
4 AND A.Cpno=B.Cno;
```



# 间接先修课的 SQL 语句表达

## 1 概述

### ■ 如何查询间接先修课：递归执行

- ② 步骤 i ( $i \geq 2$ )：找出集合  $L[i-1]$  中每一门课程的全部直接先修课，并计算它们的并集，记为  $L[i]$
- ③ 迭代执行步骤 i，直到并集  $L[i]$  为空，输出  $L[1] \cup \dots \cup L[i]$

```
1 SELECT B.Cname
2 FROM Course A, Course B
3 WHERE A.Cname = '数据结构' AND A.Cpno=B.Cno;
4
5
6 SELECT B.Cname
7 FROM Course A, Course B
8 WHERE A.Cname = '程序设计基础与C语言' AND A.Cpno=B.Cno;
```



# 间接先修课的 SQL 语句表达

## 1 概述

- 在第 2 章图 2.1 所示的 Course 表中，执行步骤 1，找出“数据库系统概论”课程的直接先修课“数据结构”，即为  $L[1]$
- 执行步骤 2，找出“数据结构”课程的先修课“程序设计基础与 C 语言”，即为  $L[2]$
- 步骤 3，找出“程序设计基础与 C 语言”的先修课，得到  $L[3]$
- 可以发现  $L[3]$  为空，递归查询结束。根据计算结果  $L[1] \cup L[2]$ ，任务 1 的输出如表所示

Cpno	Cname
81002	数据结构
81001	程序设计基础与C语言



# 【任务 2】打印一周内将过生日的学生信息

## 1 概述

### ■ 【任务 2】 打印一周内将过生日的学生信息

Student表

学号 <b>Sno</b>	姓名 <b>Sname</b>	性别 <b>Ssex</b>	生日 <b>Sbirthdate</b>	主修专业 <b>Smajor</b>
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术



# 【任务 2】打印一周内将过生日的学生信息

## 1 概述

### ■ 任务分析：SQL 如何表达

- 任务 2 需要数据库系统提供内置函数（本任务主要是日期数）。其求解思路为：扫描所有学生的出生日期，例如 2000-6-12，并执行以下步骤
  - ① 把出生日期的年份换成当前日期所在的年份（例如 2021），即 2021-6-12
  - ② 获取当前系统的日期，例如 2021-6-9
  - ③ 确定过生日的日期范围 [2021-6-9, 2021-6-16]，即以当前日期为下界，当前日期后的第七天作为上界
  - ④ 判断出生日期是否在上述日期范围内，如果是，输出该学生信息



## 【任务 3】计算学生的平均学分绩点 GPA

### 1 概述

- 【任务 3】给定学生学号，计算学生的平均学分绩点 GPA

学号为“**20180001**”的学生的选修课程

学号 <b>Sno</b>	课程号 <b>Cno</b>	成绩 <b>Grade</b>	选课学期 <b>Semester</b>	教学班 <b>Teachingclass</b>
<b>20180001</b>	<b>81001</b>	<b>85</b>	<b>20192</b>	<b>81001-01</b>
<b>20180001</b>	<b>81002</b>	<b>96</b>	<b>20201</b>	<b>81002-01</b>
<b>20180001</b>	<b>81003</b>	<b>87</b>	<b>20202</b>	<b>81003-01</b>



# 【任务 3】计算学生的平均学分绩点 GPA

## 1 概述

### ■ 任务分析：SQL 如何表达

- 难点：需要用户自主设计业务处理逻辑？
- 本任务的求解思路：给定学生学号，找出该学生所有选修课程的学分、成绩；
- 根据每门课程的成绩，参照“成绩和绩点对照表”，确定该成绩所处的范围，找出该门课程对应的绩点。



## 【任务 4】教学评价浏览与反馈

### 1 概述

- 【任务 4】教学评价浏览与反馈：学生通过交互界面提交对某一位任课老师的教学评价意见，教师浏览这些评价意见并提供反馈信息

教师教学评价表

学号 Sno	职工号 Tno	教学班号 TCno	意见内容 Assess	意见类型 CAtype	教师反馈 Feedback
20180001	19950018	81001-01	作业难度比较合适	正面	感谢肯定
20180003	19950018	81001-01	老师和助教也很耐心	正面	感谢肯定
20180002	19910101	81001-02	实验框架较为复杂	负面	根据同学们的建议， 简化框架



# 【任务 4】教学评价浏览与反馈

## 1 概述

### ■ 任务分析：SQL 如何表达

- 难点：需要建立交互功能
- 一方面，学生需要找到指定的教学班和授课教师，建立如图所示的交互界面并输入课程评价。
- 另一方面，还要建立如图所示的交互界面，教师浏览教学班学生的评价意见，并针对每条评价逐一做出回复

课程评价	
教学班	信息 81001-01
课程	程序设计基础与C语言
任课老师	董山
课程评价	<p>总体上老师讲的很好。在数据库设计部分，建议增加更丰富的应用实例</p> <p>填写课程评价</p> <p>填写完毕后点击添加</p> <p>添加</p>

图 学生输入并提交课程评价

查看教学班：81001-01的学生评教		
学号	评论	操作
20180001	感谢老师	感谢认可
20180002	感谢老师	感谢认可
20180003	总体上老师讲的很好。在数据库设计部分，建议增加更丰富的应用实例	<input type="checkbox"/> 回复 教师输入对学生评价的反馈 输入完毕后 点击回复

图：教师提交对学生评价的反馈



## 小结

### 1 概述

#### ■ SQL 语言表达能力的限制

- 无法表达递归等复杂操作—例如间接先修课的查询
- 无法对数据进行复杂操作—查询一周内将过生日的同学
- 无法自主设计业务处理逻辑—计算学生平均学分绩点
- 无法进行交互式操作—教学评价与反馈



## 1 概述

Section 1.2

### 1.2 扩展 SQL 的功能



# 扩展 SQL 语言的功能

## 1 概述

- 1. 引入新的 SQL 子句
- 2. 引入新的内置函数
- 3. 引入 PL/SQL 与存储过程/存储函数



# 引入新的 SQL 子句

## 1 概述

- 以【任务 1】为例，SQL 标准引入了 WITH RECURSIVE 子句，可执行递归查询
- 类似于 WITH RECURSIVE 子句的 SQL 扩展还有很多，例如面向联机分析处理的窗口子句，面向空间数据管理、文档数据管理的 SQL 语言扩展等
- 在介绍 WITH RECURSIVE 子句之前，先了解 WITH 子句的用法



# WITH 子句格式

## 1 概述

### ■ WITH 子句的一般格式：

```
1 WITH RS1 [(<目标列>,<目标列>)] AS      /* RS1为临时结果集的  
   命名*/  
2 (SELECT 语句1) [,                      /* RS1对应SELECT 语句  
   的执行结果*/  
3 /*SELECT语句1中的目标列与RS1中的目标列必须保持一致*/  
4 RS2 [(<目标列>,<目标列>)] AS      /* RS2为临时结果集的  
   命名*/  
5 (SELECT 语句2) ,…]                      /* RS2对应SELECT 语句  
   的执行结果*/  
6 /*SELECT语句2中的目标列与RS2中的目标列必须保持一致*/  
7 SQL语句;                                /* 执行与RS1, RS2, …,  
   相关的查询*/
```



# WITH 子句示例

## 1 概述

- 例：求 81001-01 和 81001-02 两个教学班之间学生选课平均成绩的差异。

```
1 WITH
2 RS1(Grade)
3     AS
4         (SELECT AVG(Grade) FROM SC
5          WHERE Teachingclass = '81001-01'),
6 RS2(Grade)
7     AS
8         (SELECT AVG(Grade) FROM SC
9          WHERE Teachingclass = '81001-02')
10 SELECT RS1.Grade-RS2.Grade FROM RS1,RS2;
```



# WITH RECURSIVE 子句格式

## 1 概述

### ■ WITH RECURSIVE 子句的一般格式：

```
1 WITH RECURSIVE RS AS
2   (
3     SEED QUERY          /* 初始化查询的临时结果集，记
4       为 L[1] */
5     UNION [ALL]          /* 是否需要保留重复记录，加
6       ALL 为保留 */
7     RECURSIVE QUERY
8     /* 执行递归查询，得到全部临时结果集，即 L[2] … L[i
9     ] */
10    )
11  SQL 语句           /* 执行与 RS 相关的查询 */
```

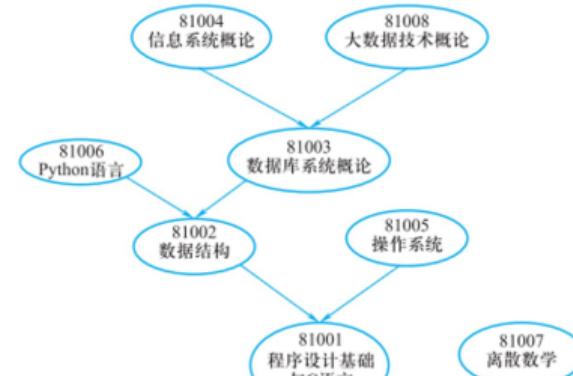


# WITH RECURSIVE 子句示例

## 1 概述

### ■【任务 1】打印“数据库系统概论”课程的所有先修课信息

课程号Cno	课程名Cname	学分Ccredit	先修课程Cpno
81001	程序设计基础与C语言	4	NULL
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python语言	3	81002
81007	离散数学	4	NULL
81008	大数据技术概论	4	81003





# WITH RECURSIVE 子句示例

## 1 概述

- 【任务 1】打印“数据库系统概论”课程的所有先修课信息



# WITH RECURSIVE 子句示例

## 1 概述

```
1 WITH RECURSIVE RS AS
2   (SELECT Cpno FROM Course WHERE Cname = '数据库系统概
3     论'
4   /*初始化RS，假设结果集为L[1]，即“数据库系统概论”的
5     所有直接先修课*/
6   UNION
7   SELECT Course.Cpno FROM Course,RS WHERE RS.Cpno =
8     Course.Cno)
9   /*递归查询第i层(i>=1)的数据，即第i-1层数据的直接先修
10    课课程号，并更新RS*/
11  SELECT Cno, Cname FROM Course WHERE Cno IN (SELECT Cpno
12    FROM RS);
13 /*根据RS中记录的所有先修课程号，通过查找课程表，输出课程
14    号与课程名*/
```



# 引入新的内置函数

## 1 概述

■ SQL 常用的内置函数可以分为：

- 数学函数（如绝对值函数等）
- 聚合函数（如求和、求平均函数等）
- 字符串函数（如求字符串长度、求子串函数等）
- 日期和时间函数（如返回当前日期函数等）
- 格式化函数（如字符串转 IP 地址函数等）
- 控制流函数（如逻辑判断函数等）
- 加密函数（如使用密钥对字符串加密函数等）
- 系统信息函数（如返回当前数据库名、服务器版本函数等）



# 日期和时间函数示例

## 1 概述

### ■【任务 2】打印一周内将过生日的学生信息

```
1 SELECT Sno, Sname, Ssex, Sbirthdate, Smajor
2 FROM Student
3 WHERE to_date(to_char(current_date, 'yyyy') || '-' ||
     to_char(Sbirthdate, 'mm-dd'))
4 BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7' DAY;
```

### ■ 在 WHERE 语句中使用了以下内置函数:

- ① 内置函数 `current_date` 返回当前的系统日期, 例如 2021-6-9
- ② 内置函数 `to_char(current_data, 'yyyy')`返回当前系统日期的年份, 例如 2021;  
`to_char(Sbirthdate, 'mm-dd')`返回学生出生日期中具体的月份和日期, 例如 6-9



# 日期和时间函数示例

## 1 概述

- ③ `to_date(to_char(current_date, 'yyyy')||'-'|| to_char(Sbirthdate, 'mm-dd'))` 表示把当前年份与出生日期用'-'连在一起。符号'||'用于把其左右两边的字符串连在一起
  - ④ 内置函数 `to_date()` 的作用是将字符类型按一定格式转化为日期类型。
  - ⑤ `CURRENT_DATE + INTERVAL '7' DAY` 是对当前的日期调整后的日期。参数 `interval` 是年 (yyyy)、季度 (q)、月 (m)、日 (d)、时 (h) 等粒度的时间单位。例如，  
`CURRENT_DATE + INTERVAL '7' DAY` 获得当前日期之后第七天的日期，即返回 2021-6-16
- 通过执行此 WHERE 语句，判断学生表中每位学生转换后的出生日期是否在 [2021-6-9, 2021-6-16] 区间内，如果是，打印该学生的信息



# 引入 PL/SQL 与存储过程/存储函数

## 1 概述

- 关系数据库管理系统中引入 PL/SQL、存储过程和自定义函数等方法，使得用户可以自定义程序逻辑，开发完成业务逻辑复杂的应用系统。



## 1 概述

### Section 1.3

## 1.3 通过高级语言实现复杂应用



# 通过高级语言实现复杂应用

## 1 概述

### ■ 通过动态链接库调用的方式

- 关系数据库管理系统的功能被包装成一个子程序，由应用程序通过动态链接库调用来获得数据管理的功能

### ■ 基于嵌入式 SQL 的方式

- 将 SQL 嵌入到高级语言中混合编程，SQL 语句负责操纵数据库，高级语言语句负责控制逻辑流程

### ■ 基于 ODBC/JDBC 的中间件方式

- 建立了连接不同数据库的一组规范。无论使用什么数据库，都采用同样的一组 API 来访问数据库



# Table of Contents

## 2 过程化 SQL

- ▶ 概述
- ▶ 过程化 SQL
- ▶ JDBC 编程
- ▶ 基于 MVC 框架的数据库应用开发
- ▶ 本章小结



## 2 过程化 SQL

Section 2.1

### 2.1 过程化 SQL 的块结构



## 2.1 过程化 SQL 的块结构

### 2 过程化 SQL

#### ■ 过程化 SQL

- SQL 的扩展
- 增加了过程化语句功能
- 基本结构是块
  - 块之间可以互相嵌套
  - 每个块完成一个逻辑操作

#### ■ 过程化 SQL 块的基本结构

- ① 定义部分: DECLARE 变量、常量、游标、异常等
  - 定义的变量、常量等只能在该基本块中使用
  - 当基本块执行结束时, 定义就不再存在



## 2.1 过程化 SQL 的块结构

### 2 过程化 SQL

#### ② 执行部分

```
1 BEGIN  
2     SQL语句、过程化SQL的流程控制语句  
3     EXCEPTION  
4         异常处理部分  
5 END;
```

- 遇到不能继续执行的情况称为异常
- 在出现异常时，采取措施来纠正错误或报告错误



## 2 过程化 SQL

Section 2.2

# 变量和常量的定义



## 2.2 变量和常量的定义

### 2 过程化 SQL

#### ① 变量和常量的定义

- 变量名 数据类型 [[NOT NULL] := 初值表达式]
- 变量名 数据类型 [[NOT NULL] 初值表达式]

#### ② 常量定义

- 常量名 数据类型 CONSTANT := 常量表达式
- 常量必须要赋予一个值，并且该值在存在期间或常量的作用域内不能改变。如果试图修改它，过程化 SQL 将返回一个异常

#### ③ 赋值语句

- 变量名称 := 表达式



## 2 过程化 SQL

Section 2.3

# 流程控制



## 2.3 流程控制

### 2 过程化 SQL

#### ■ 过程化 SQL 功能

##### ① 条件控制语句: IF-THEN, IF-THEN-ELSE 和嵌套的 IF 语句

- IF-THEN 语句格式
- IF-THEN-ELSE 语句格式
- 在 THEN 和 ELSE 子句中还可以再包含 IF 语句, 即 IF 语句可以嵌套



## 2.3 流程控制

### 2 过程化 SQL

```
1 IF condition THEN  
2     Sequence_of_statements;  
3 END IF;  
4  
5 IF condition THEN  
6     Sequence_of_statements1;  
7 ELSE  
8     Sequence_of_statements2;  
9 END IF;
```



## 2.3 流程控制

### 2 过程化 SQL

#### ② 循环控制语句: LOOP, WHILE-LOOP, FOR-LOOP

```
1 LOOP  
2   Sequence_of_statements;  
3 END LOOP;
```

- 多数数据库服务器的过程化 SQL 都提供 EXIT、BREAK 或 LEAVE 等循环结束语句，保证 LOOP 语句块能够在适当的条件下提前结束



## 2.3 流程控制

### 2 过程化 SQL

```
1 WHILE condition LOOP  
2     Sequence_of_statements;  
3 END LOOP;
```

- 每次执行循环体语句之前，首先对条件进行求值
- 如果条件为真，则执行循环体内的语句序列
- 如果条件为假，则跳过循环并把控制传递给下一个语句



## 2.3 流程控制

### 2 过程化 SQL

```
1 FOR count IN [REVERSE] bound1 … bound2 LOOP  
2     Sequence_of_statements;  
3 END LOOP;
```



## 2.3 流程控制

### 2 过程化 SQL

#### ③ 错误处理

- 如果过程化 SQL 在执行时出现异常，则应该让程序在产生异常的语句处停下来，根据异常的类型去执行异常处理语句
- SQL 标准对数据库服务器提供什么样的异常处理做出了建议，要求过程化 SQL 管理器提供完善的异常处理机制



## 2 过程化 SQL

Section 2.4

# 游标的定义与使用



## 2.4 游标的定义与使用

### 2 过程化 SQL

#### ■ 游标

- 在过程化 SQL 中，如果SELECT语句只返回一条记录，可以将该结果存放到变量中
- 当查询返回多条记录时，就要使用游标对结果集进行处理。一个游标与一个 SQL 语句相关联
- 游标的用户接口：
  1. 声明游标
  2. 打开游标
  3. 使用游标
  4. 关闭游标



## 2.4 游标的定义与使用

### 2 过程化 SQL

#### ① 声明游标

```
1 DECLARE 游标名 [(参数1 数据类型, 参数2 数据类型, …)]  
2 CURSOR FOR  
3 SELECT 语句;
```

- 定义游标仅仅是一条说明性语句，这时关系数据库管理系统并不执行SELECT语句



## 2.4 游标的定义与使用

### 2 过程化 SQL

#### ② 打开游标

```
1 OPEN 游标名 [(参数1 数据类型, 参数2 数据类型, …)];
```

- 打开游标实际上是执行相应的SELECT语句，把查询结果取到缓冲区中。这时游标处于活动状态，指针指向查询结果集中的第一条记录



## 2.4 游标的定义与使用

### 2 过程化 SQL

#### ③ 使用游标

```
1 FETCH 游标名 INTO 变量1[, 变量2, …];
```

- 变量必须与SELECT语句中的目标列表达式一一对应
- 用 FETCH 语句把游标指针向前推进一条记录，同时将缓冲区中的当前记录取出来送至变量供过程化 SQL 进一步处理
- 循环执行 FETCH 语句，逐条取出结果集中的行进行处理



## 2.4 游标的定义与使用

### 2 过程化 SQL

#### ④ 关闭游标

```
1 CLOSE 游标名；
```

- 游标被关闭后就不再和原来的查询结果集相联系
- 但被关闭的游标可以再次被打开，与新的查询结果相联系



## 2.4 游标的定义和使用（例）

### 2 过程化 SQL

- 根据给定学号 20180001，使用游标输出该学生的全部选课记录

```
1 DECLARE
2     CnoOfStudent CHAR(10);
3     GradeOfStudent INT;
4     mycursor CURSOR FOR
5         SELECT Cno,Grade FROM SC WHERE Sno = '20180001';
6 BEGIN
7     OPEN mycursor;          /* 打开游标 */
8     LOOP                  /* 循环遍历游标 */
9         FETCH mycursor INTO CnoOfStudent, GradeOfStudent; /* 检索游标 */
10        EXIT WHEN mycursor \%NOTFOUND;
11        RAISE NOTICE 'Sno:20180001, Cno:\%, Grade:\%', CnoOfStudent,
12             GradeOfStudent;
13    END LOOP;
14    CLOSE mycursor;        /* 关闭游标 */
15 END;
```



## 2 过程化 SQL

Section 2.5

# 存储过程



# 存储过程

## 2 过程化 SQL

### ■ 存储过程

- 类似于高级语言程序，过程化 SQL 程序也可以被命名和编译，并保存在数据库中，称为存储过程（stored procedure）或存储函数（stored function），供其他过程化 SQL 调用
- 存储过程或存储函数也是一类数据库的对象，需要有创建、删除等语句。这里的存储函数指自定义函数

### ■ 存储过程的用户接口

- ① 创建存储过程
- ② 执行存储过程
- ③ 修改存储过程
- ④ 删除存储过程



# 存储过程-创建存储过程

## 2 过程化 SQL

### ① 创建存储过程

```
1 CREATE OR REPLACE PROCEDURE 过程名(
2     [[IN|OUT|INOUT]] 参数1 数据类型 ,
3     [[IN|OUT|INOUT]] 参数2 数据类型 ,
4     … ]
5 )                                /*存储过程首部*/
6 AS <过程化SQL块>;           /*存储过程体，描述该存储过程的操作*/
```

□ 过程名：数据库服务器合法的对象标识



# 存储过程-创建存储过程

## 2 过程化 SQL

- 参数列表：存储过程提供了 IN、OUT、INOUT 三种参数模式，分别对应输入、输出、输入输出三种语义，不声明参数模式时，缺省为 IN 类型。输入参数在被调用时需要指定参数值，输出参数调用时不传入参数值，而是作为返回值返回。输入输出参数调用时需要传入初始值，并会返回操作后的最终值。参数列表中需要指定参数模式、参数名、以及参数的数据类型
- 过程体：是一个 < 过程化 SQL 块 >，包括声明部分和可执行语句部分



# 存储过程-创建存储过程

## 2 过程化 SQL

■ 给定学生学号，计算学生的平均学分绩点 GPA

■ 求解思路

- 给定学生学号，找出该学生所有选修课程的学分、成绩
- 根据每门课程的成绩，参照成绩和绩点对照表，确定该成绩所处的范围，找出该门课程对应的学分绩点

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass	编码	成绩下限	成绩上限	绩点
20180001	81001	85	20192	81001-01	1	0	59	0
20180001	81002	96	20201	81002-01	2	60	69	1
20180001	81003	87	20202	81003-01	3	70	79	2
					4	80	89	3
					5	90	100	4



# 存储过程-创建存储过程

## 2 过程化 SQL

■ 给定学生学号，计算学生的平均学分绩点 GPA

■ 求解思路

- “81001” 课程考试成绩为 85 分，参照右下表，85 分对应成绩范围为 [80,89]，该门课程对应的学分绩点为 3。类似地，课程号为“81002” 对应的学分绩点为 4，课程号为“81003” 对应的学分绩点为 3
- 81001-81003 三门课程的学分都是 4。根据平均学分绩点 GPA 的计算公式 = 总学分绩/总学分 = (每门课程的学分 \* 对应课程的绩点) 的总和/12 =  $(3*4 + 4*4 + 3*4)/12 = 3.33$

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass	编码	成绩下限	成绩上限	绩点
20180001	81001	85	20192	81001-01	1	0	59	0
20180001	81002	96	20201	81002-01	2	60	69	1
20180001	81003	87	20202	81003-01	3	70	79	2
					4	80	89	3
					5	90	100	4



# 存储过程-创建存储过程

## 2 过程化 SQL

### ■ 给定学生学号，计算学生的平均学分绩点 GPA

```
1 CREATE OR REPLACE PROCEDURE compGPA(      /*定义存储过compGPA*/
2     IN inSno CHAR(10),          /*输入参数： 学生学号inSno*/
3     OUT outGPA FLOAT)         /*输出参数： 平均学分绩outGPA*/
4
5 AS
6
7 DECLARE
8     courseGPA INT;           /*声明变量courseGPA，临时存储课程学分绩 */
9     totalGPA INT;            /*声明变量totalGPA，临时存储总学分绩 */
10    totalCredit INT;          /*声明变量totalCredit，临时存储总学分*/
11    grade INT;               /*声明变量grade，临时存储学生成绩 */
12    credit INT;              /*声明变量credit，临时存储课程学分 */
13    mycursor CURSOR FOR      /*声明游标mycursor */
14        SELECT Ccredit, grade FROM SC, Course
15        WHERE Sno = inSno AND SC.Cno = Course.Cno;
```



# 存储过程-创建存储过程

## 2 过程化 SQL

```
15    BEGIN
16        totalGPA := 0;
17        totalCredit := 0;
18        OPEN mycursor;          /* 打开游标mycursor */
19        LOOP                  /* 循环遍历游标 */
20            FETCH mycursor INTO credit, grade;      /* 检索游标 */
21            EXIT WHEN mycursor%NOTFOUND;
22            IF grade BETWEEN 90 AND 100 THEN courseGPA := 4.0;
23            ELSIF grade BETWEEN 80 AND 89 THEN courseGPA := 3.0;
24            ELSIF grade BETWEEN 70 AND 72 THEN courseGPA := 2.0;
25            ELSIF grade BETWEEN 60 AND 69 THEN courseGPA := 1.0;
26            ELSE courseGPA := 0;
27        END IF;    /* 参照表8.2, 根据成绩找出某门课程对应的学分绩点 */
28                totalGPA := totalGPA + courseGPA * credit;
29                totalCredit := totalCredit + credit;
30        END LOOP;
31        CLOSE mycursor;          /* 关闭游标mycursor */
32        outGPA:= 1.0 * totalGPA / totalCredit;
33    END;
```



# 存储过程

## 2 过程化 SQL

### ② 执行存储过程

```
1 CALL/PERFORM [PROCEDURE] 过程名([参数1, 参数2, ...]);
```

- 使用 CALL 或者 PERFORM 等方式激活存储过程的执行
- 在过程化 SQL 中，数据库服务器支持在过程体中调用其他存储过程



# 存储过程

## 2 过程化 SQL

- 查询学号为“20180001”学生的课程 GPA。

- 在调用含有输入参数和输入输出参数的存储过程时，需要指定具体的参数值。在调用含有输出参数的存储过程时，对应位置不需要传入参数值，但需要事先定义输出变量

```
1 DECLARE outGPA FLOAT;
2 BEGIN
3     CALL compGPA('20180001', outGPA);
4     RAISE NOTICE 'GPA: %', outGPA;
5 END;
```



# 存储过程

## 2 过程化 SQL

### ③ 修改存储过程

- 重命名/重新编译

```
1 ALTER PROCEDURE 过程名1 RENAME TO 过程名2;  
2 ALTER PROCEDURE 过程名1 COMPILE;
```

### ④ 删除存储过程

```
1 DROP PROCEDURE 过程名;
```



# 存储过程

## 2 过程化 SQL

### ■ 存储过程的优点

- ① 运行效率高
- ② 降低了客户机和服务器之间的通信量
- ③ 方便实施企业规则



## 2 过程化 SQL

Section 2.6

# 存储函数



# 存储函数

## 2 过程化 SQL

### ■ 存储函数和存储过程的异同

- ① 同：都是持久性存储模块
- ② 异：函数必须指定返回的类型



# 存储函数

## 2 过程化 SQL

### ① 函数的定义/执行/修改语句

```
1 CREATE OR REPLACE FUNCTION 函数名([参数1 数据类型, 参数2  
      数据类型, ...])  
2 RETURNS <类型>  
3 AS <过程化SQL块>;  
4  
5 CALL/SELECT 函数名 ([参数1, 参数2, ...]);  
6  
7 ALTER FUNCTION 函数名1 RENAME TO 函数名2;  
8 ALTER FUNCTION 函数名 COMPILE;
```



# Table of Contents

## 3 JDBC 编程

- ▶ 概述
- ▶ 过程化 SQL
- ▶ JDBC 编程
- ▶ 基于 MVC 框架的数据库应用开发
- ▶ 本章小结



## 3 JDBC 编程

### Section 3.1

# JDBC **工作原理概述**



# JDBC 工作原理概述

## 3 JDBC 编程

- JDBC (Java DataBase Connection)
  - 由于不同的数据库管理系统的存在，在某个关系数据库管理系统下编写的应用程序就不能在另一个关系数据库管理系统下运行
  - 许多应用程序需要共享多个部门的数据资源，访问不同的关系数据库管理系统
- JDBC 是面向 Java 语言的软件开发工具包 (Java Development Kit, JDK) 中有关数据库的一个组成部分，其提供了一组访问数据库的应用程序编程接口 (Application Programming Interface, API)
- JDBC 约束力
  - 规范应用开发
  - 规范关系数据库管理系统应用接口



# JDBC 工作原理概述

## 3 JDBC 编程

- JDBC 应用系统的体系结构
  - ① 用户应用程序
  - ② JDBC 驱动程序管理器
  - ③ 数据源



# JDBC 工作原理概述

## 3 JDBC 编程





### 3 JDBC 编程

#### Section 3.2

## JDBC APIs **基础**



# JDBC APIs 基础

## 3 JDBC 编程

### ■ JDBC 中的常用类

- JDBC 进行应用程序开发涉及到的所有类都包含在 `java.sql` 包中
- 不同的 JDBC 版本接口名和使用略有差异

类名	路径	备注
驱动程序类	<code>java.sql.Driver</code>	由各数据库厂商提供
驱动程序管理类	<code>java.sql.DriverManager</code>	作用于应用程序与驱动程序之间
数据库连接类	<code>java.sql.Connection</code>	用于建立与指定数据库的连接
静态 <b>SQL</b> 语句执行类	<code>java.sql.Statement</code>	用于执行静态 <b>SQL</b> 语句并返回结果
动态 <b>SQL</b> 语句执行类	<code>java.sql.PreparedStatement</code>	用于执行含参 <b>SQL</b> 语句并返回结果
存储过程语句执行类	<code>java.sql.CallableStatement</code>	用于执行存储过程语句并返回结果
结果集处理类	<code>java.sql.ResultSet</code>	用于检索结果集中的数据



# JDBC APIs 基础

## 3 JDBC 编程

- 数据类型
- 由数据库管理系统的驱动程序完成自身数据类型和 JDBC 标准数据类型的映射
  - SQL 数据类型：用于数据源
    - VARCHAR、CHAR、BIT、NUMERIC、DATE 等
  - Java 数据类型：用于应用程序的 Java 代码
    - String、boolean、BigDecimal、byte、short、int 等

	SQL 数据类型	Java 数据类型
SQL 数据类型	数据源之间转换	应用程序变量传递给语句
Java 数据类型	从数据库中读取数据赋值给应用程序变量	应用程序变量之间转换

Table: SQL 数据类型和 java 数据类型之间的转换规则



## 3 JDBC 编程

### Section 3.3

# 使用 JDBC 操纵数据库的工作流程



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

- ① 步骤 1：加载驱动程序
- ② 步骤 2：建立与数据库的连接
  - 定义连接的 URL
  - 利用生成的 URL 建立与数据库的连接
- ③ 步骤 3：执行 SQL 语句
  - 创建语句执行类对象
  - 执行 SQL 语句，可以通过 executeQuery、executeUpdate、execute 三种方式执行
- ④ 步骤 4：处理结果集
- ⑤ 步骤 5：释放资源



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

### ① 步骤 1：加载驱动程序

- 驱动程序在 JDBC API 中实现定义数据交互的接口
- 【例 8.7】对 Kingbase、Oracle、SQL Server 加载数据库驱动

```
1 Class.forName("com.kingbase.Driver");      /* Kingbase */  
2 Class.forName("oracle.jdbc.OracleDriver");   /* Oracle */  
3 Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");  
4 /* SQL Server */
```



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

### ② 步骤 2：建立与数据库的连接

- 加载驱动后，可以通过 URL 地址与数据库建立连接
- URL 包含了连接数据库所需的协议、子协议和数据库名称，定义格式为：  
< 协议名 >:< 子协议名 >:< 数据库名称 >
- 【例 8.8】定义与 Kingbase、Oracle、SQL Server 数据库连接的 URL

```
1 strURL = "jdbc:kingbase://" + 服务器名 + ":" + 端口号 + "/" + 数据库名;
2 strURL = "jdbc:oracle:thin:@\" + 服务器名 + ":" + 端口号 + ":" + 数据库名
3 strURL = "jdbc:microsoft:sqlserver://" + 服务器名 + ":" + 端口号 + ":" +
           数据库名
4 // Kingbase、Oracle、SQL Server 的默认端口号分别为 54321、1521、1433
```



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

- 【例 8.9】建立与 Kingbase 数据库的连接，假定服务器地址为 192.168.0.118，端口为 54321，数据库名为 DB-Student，用户名为 Info001，密码为 123456

```
1 String strURL ="jdbc:kingbase:// 192.168.0.118:54321/DB-Student";
2 Connection conn= DriverManager.getConnection(strURL," Info001","123456")
;
```



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

### ③ 步骤 3：执行 SQL 语句

- 静态语句执行类对象 (Statement)
- 派生执行类对象：
  - 动态语句执行类 (PreparedStatement): 执行动态的 SQL 语句
  - 存储过程执行类 (CallableStatement): 执行数据库存储过程
- 执行方法
  - ResultSet executeQuery(): 执行数据库查询语句
  - int executeUpdate(): 处理增、删、改以及定义语句
  - boolean execute(): 处理存储过程或动态 SQL 语句
- SQL 注入防御
  - 预编译
  - 正则过滤，输入验证，框架……



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

### □ 【8.10】使用 JDBC 向课堂评价表中插入一条记录

```
1 PreparedStatement stmt = conn.prepareStatement("INSERT INTO SC VALUES  
    (?, ?, ?, ?, ?, ?)");  
2 /* 生成PreparedStatement类对象中的动态参数，注意第六个字段Feedback，未设  
   置输入值 */  
3 stmt.setString(1, " 20180001 ");      /*设置学生学号*/  
4 stmt.setString(2, "19950018");       /*设置职工号*/  
5 stmt.setString(3, "81001-01");       /*设置教学班号*/  
6 stmt.setString(4, "老师讲得很出色"); /*设置学生评价意见*/  
7 stmt.setBoolean(5,true);           /*设置学生评价意见类型*/  
8 stmt.executeUpdate();
```



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

### ④ 步骤 4：处理结果集

#### □ ResultSet: 结果集合类对象

- .GetXXX(参数):

获取元组的属性值，XXX 代表某种数据类型

可以指定参数为列号 (JDBC 的列从 1 开始) 或列名

- 游标 (cursor): JDBC 处理结果集的机制

TYPE\_FORWARD\_ONLY: 只能向下滚动 (默认类型)

TYPE\_SCROLL\_INSENSITIVE 双向滚动, 区别为是否同步数据库更新操作



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

- 【8.12】遍历教师职工号为 19950018，教学班号为 81001-01 的学生课程评价详情

```
1 String SQL = "SELECT Sno, Assess, CAtype, Feedback FROM ClassAssess  
2   WHERE Tno='19950018' AND TCo = '81001-01'";  
3 ResultSet rs = stmt.executeQuery(SQL);  
4 while(rs.next()){  
5     String Sno = rs.getString("Sno");          /*等价于rs.getString(1)*/  
6     String strAssess = rs.getString("Assess");    /*等价于rs.getString  
7       (2) */  
8     Boolean bCAtype = rs.getBoolean ("CAtype");    /*等价rs.getBoolean  
9       (3) */  
10    String strFeedback = rs.getString("Feedback"); /*等价于rs.getString  
11      (4) */  
12    System.out.printf(" [%s,%b,%s]%n", strAssess, bCAtype, strFeedback);  
13 }
```



# 使用 JDBC 操纵数据库的工作流程

## 3 JDBC 编程

### ⑤ 步骤 5：释放资源

- 执行结束后，将与数据库进行交互的对象释放
- 释放资源有标准的顺序：
  - 关闭结果集：ResultSet.close()
  - 关闭语句执行类对象：Statement.close()
  - 释放数据库连接对象：Connection.close()
- 释放资源的 Coding Tips
  - Connection 对象在应用程序中较为稀有，尽量做到晚创建，早释放
  - 推荐在 finally 代码块中释放资源，以满足异常处理机制（finally 语句在什么情况下都会执行，确保一定会释放资源）



# 【例 8.13】基于 JDBC 实现【任务 4】

## 3 JDBC 编程

### ■ 选课关系模式:

- SC(Sno, TCno, Grade)

### ■ 教师关系模式:

- Teacher(Tno, Tname, Ttitle, Tbirthdate, Dno)

### ■ 教学班关系模式:

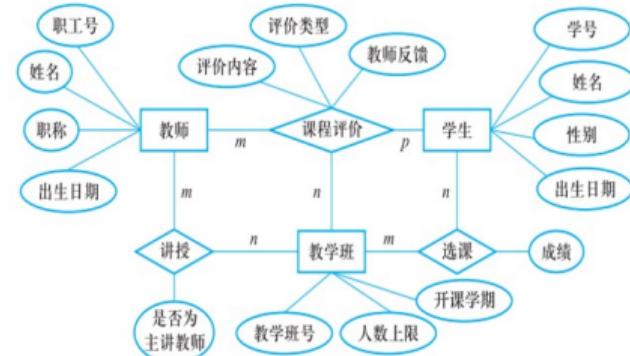
- TeachingClass(TCno, Capacity, Semester, Cno)

### ■ 讲授关系模式:

- Teaching(TCno, Tno, isLeading)

### ■ 课堂评价关系模式:

- ClassAssess(Sno, Tno, TCno, Assess, CAtype, CRfeedback)





# 【例 8.13】基于 JDBC 实现【任务 4】

## 3 JDBC 编程

### ■ 数据库连接的管理

- 设置连接数据库的 URL
- 创建连接数据库的静态函数
- 创建释放数据库连接对象的静态函数

### ■ 封装 (步骤 1-步骤 2-步骤 6) 到数据库连接管理类中

```
1 public class ConnectionManager{  
2     static final String jdbcUrl="jdbc:kingdb  
3         ://192.168.0.118:54321/DB-Student";  
4     static final String jdbcUsername = "Info001";  
5     static final String jdbcPassword = " 123456";  
6     private static Connection connection = null;  
7     /* 静态代码段， 加载Kingbase数据库驱动程序 */  
8     static {  
9         Class.forName("com.kingbase.Driver");  
10    }  
11    public static Connnection createConnection(){  
12        connection = DriverManager.getConnection(jdbcUrl  
13            , jdbcUsername, jdbcPassword);  
14        System.out.println("数据库连接成功");  
15        return connection;  
16    }  
17    /* 释放资源 */  
18    public static void release(){  
19        if (connection != null) connection.close();  
20        System.out.println("释放资源成功");  
21    }  
}
```



## 【例 8.13】基于 JDBC 实现【任务 4】

### 3 JDBC 编程

- 学生浏览指定的教学班和授课教师，并输入课程评价
- 完成步骤 3-4
- 步骤 3：创建语句对象
- 步骤 4：执行语句对象

```
1 Connection conn = ConnectionManager.createConnection();
2 Statement stmt = conn.createStatement();
3 /* 设置如下任务的SQL语句；获取学生所选的教学班及授课老师
   */
4 /* 假设strSno为学生学号 */
5 String SQL4ClassAssess = "SELECT TCno, Tno FROM Student,
                           Teacher, Teaching, SC WHERE Teacher.Tno=Teaching.
                           Tno AND Teaching.TCno=SC.TCno AND SC.sno='"+strSno;
```



# 【例 8.13】基于 JDBC 实现【任务 4】

## 3 JDBC 编程

### ■ 执行步骤 5

- 根据结果集，执行用户的逻辑
- 构建学生课程评价的动态 SQL 语句
- 遍历学生选课的每一条记录，在动态 SQL 语句中设置学生对该门课程的评价，并更新该条记录

### ■ 执行步骤 6

- 释放结果集、语句、连接对象

```
1 ResultSet rs=stmtnt.executeQuery(SQL4ClassAssess);
2 /* 设置插入学生课程评价的动态SQL语句 */
3 String insertSQL = "INSERT INTO ClassAssess(Sno, Tno,
4 PreparedStatement ps = conn.prepareStatement(insertSQL);
5 while(rs.next){
6     /* 阅读该学生所选的教学班及授课老师 */
7     String strTno = rs.getString("Tno");
8     String strTCno = rs.getString("TeachingClassNo");
9     /* 对该授课教师讲授的教学班进行评价 */
10    ps.setString(1, strSno);
11    ps.setString(2, strTno);
12    ps.setString(3, strTeachingClassNo);
13    ps.setString(4, "老师讲得很好");
14    /* 课程评价根据用户输入 */
15    ps.setBoolean(5, true);
16    ps.executeUpdate();
17 }
18 rs.close();
19 ps.close();
20 stmtnt.close();
21 ConnectionManager.release();
```



# Table of Contents

## 4 基于 MVC 框架的数据库应用开发

- ▶ 概述
- ▶ 过程化 SQL
- ▶ JDBC 编程
- ▶ 基于 MVC 框架的数据库应用开发
- ▶ 本章小结



## 4 基于 MVC 框架的数据库应用开发

### Section 4.1

# 基于 MVC 框架的数据库应用开发

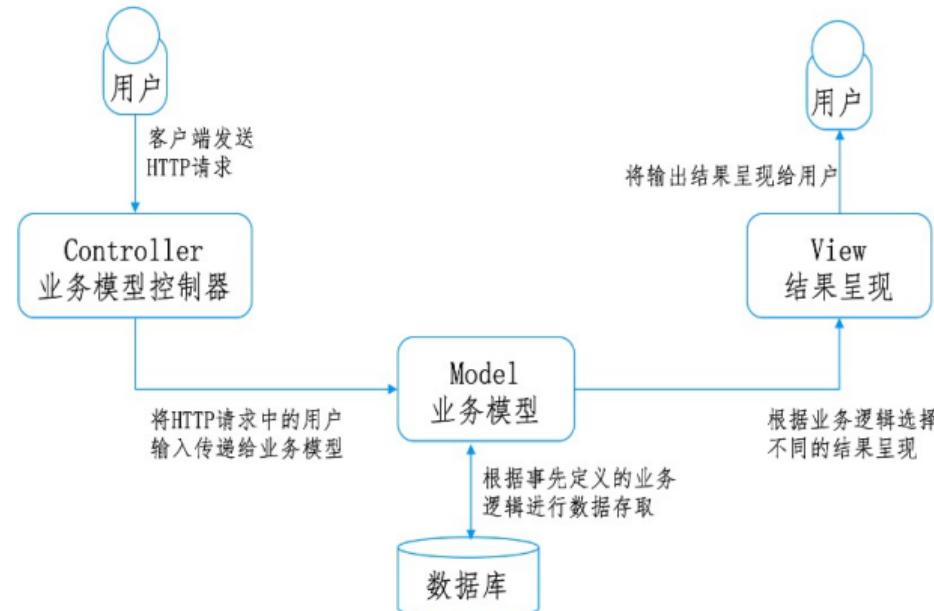


# 什么是 MVC

## 4 基于 MVC 框架的数据库应用开发

- MVC 框架包含三个部分每个部分各自处理自己的工作，互不干扰

- 业务模型控制器 (Controller)
- 业务模型 (Model)
- 用户结果呈现 (View)

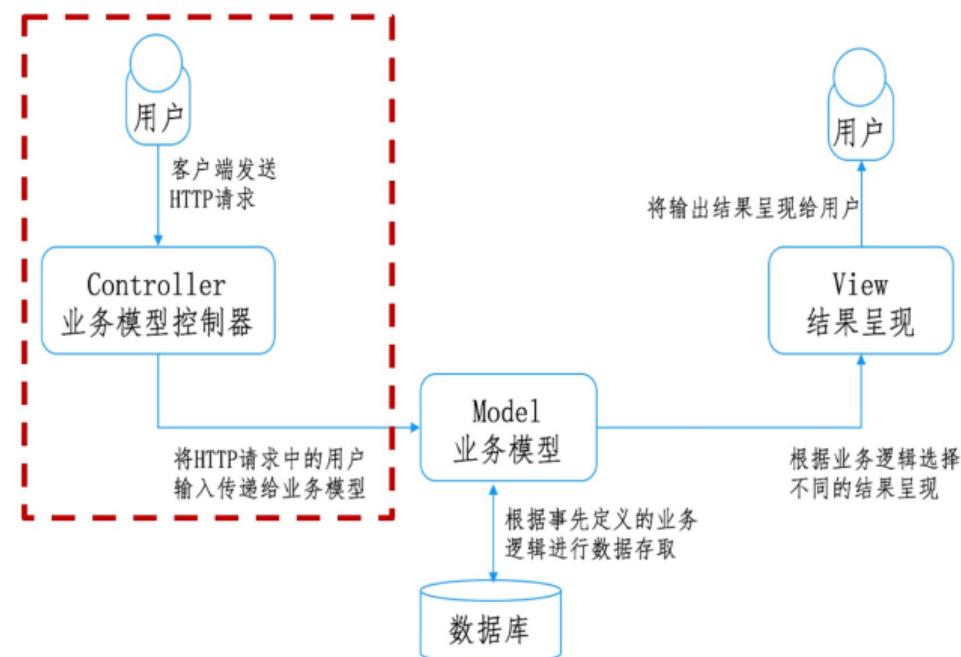




# 什么是 MVC

## 4 基于 MVC 框架的数据库应用开发

- **业务模型控制器**: 接收用户通过浏览器发送的 HTTP 服务请求
  - 包括用户需要选择的业务模型, 以及其它可选的输入参数
  - 可以是用户在文本框中输入的文本、下拉列表框中的数据项等
  - 也可以是用户的操作类型



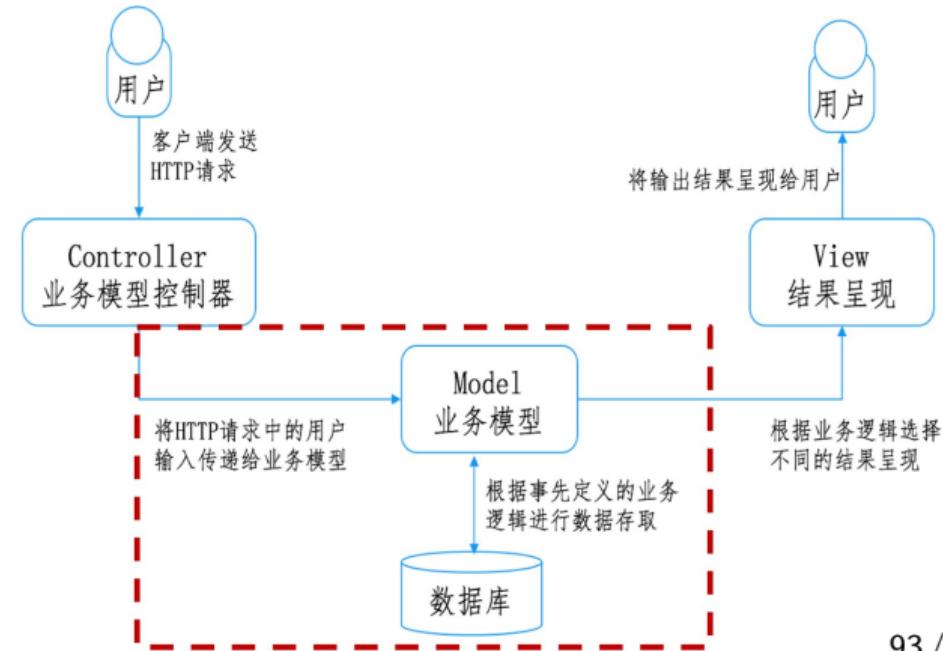


# 什么是 MVC

## 4 基于 MVC 框架的数据库应用开发

### ■ 业务模型：建立用户的业务逻辑

- 解析出来的用户数据，执行事先定义的业务逻辑，并操纵数据库存取数据
- 输出独立于具体的数据格式，可为多个用户结果呈现提供展示所需要的数据，减少了代码的重复性

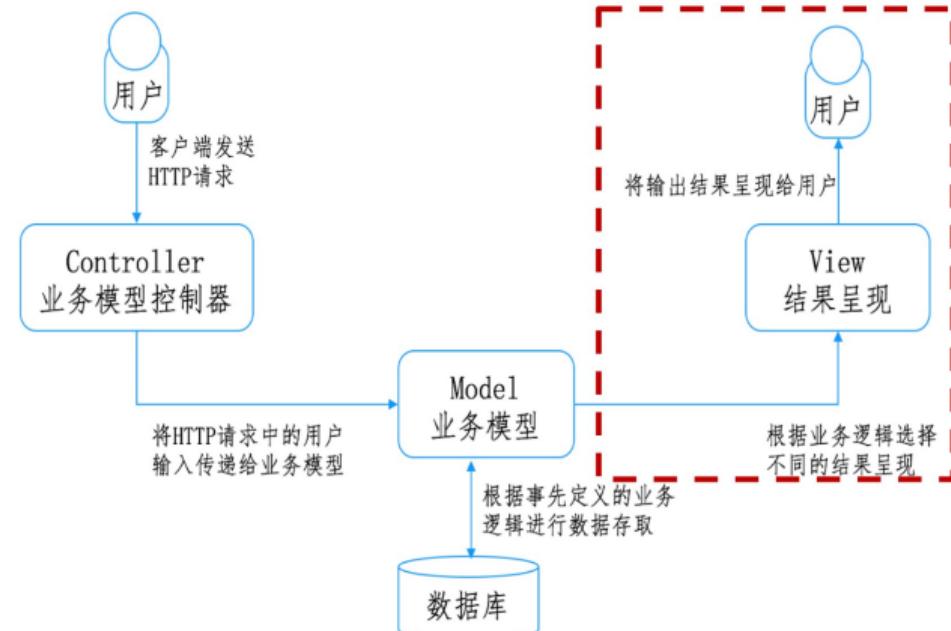




# 什么是 MVC

## 4 基于 MVC 框架的数据库应用开发

- **用户结果呈现**: 用户看到并与之交互获得结果展示的界面
  - 根据业务模型输出的结果反馈给客户端进行呈现





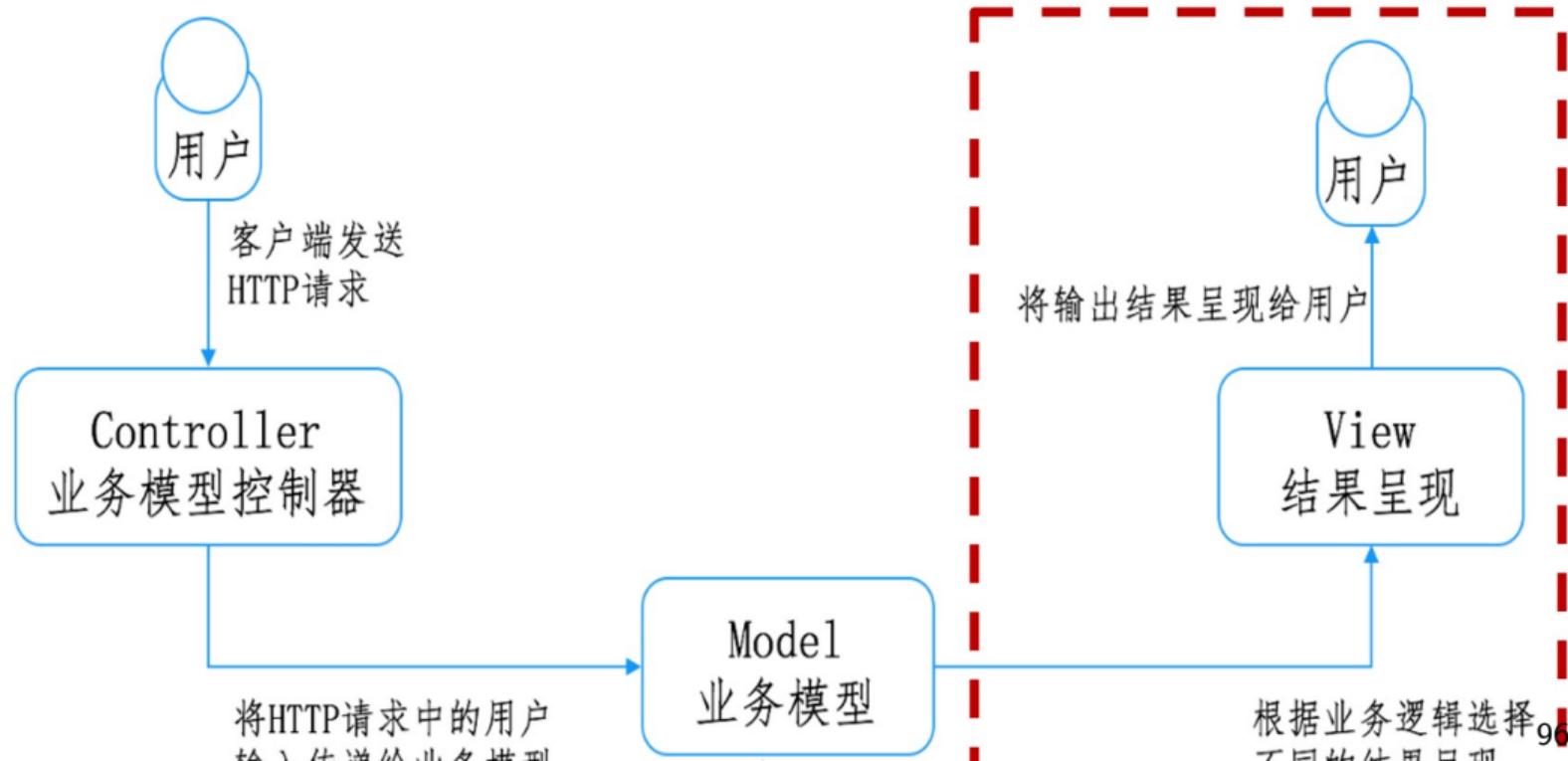
# 使用 MVC 框架实现任务 4 的系统流程图

4 基于 MVC 框架的数据库应用开发



# 使用 MVC 框架实现任务 4 的系统流程图

4 基于 MVC 框架的数据库应用开发





# Table of Contents

## 5 本章小结

- ▶ 概述
- ▶ 过程化 SQL
- ▶ JDBC 编程
- ▶ 基于 MVC 框架的数据库应用开发
- ▶ 本章小结



## 本章小结

### 5 本章小结

#### ■ 扩展 SQL 语言的功能

- 引入新的 SQL 子句、引入新的内置函数、引入 PL/SQL 以及存储过程和存储函数等技术
- SQL 与高级语言具有不同的数据处理方式。SQL 是面向集合的，而高级语言是面向记录的。游标就是用来协调这两种不同的处理方式的机制

#### ■ 通过高级语言实现复杂应用

- JDBC 的工作原理和工作流程
- 基于 MVC 框架的开发方式



# Q&A



## Reference 参考资料

王珊、杜小勇、陈红. 数据库系统概论 (第 6 版), 2023.