第12讲：过程化SQL

（本讲包括2个视频，对应教科书8.2，8.3节）

过程化SQL

突破SQL语言局限性的三种技术方案：

1 利用高级语言的表达能力：嵌入式SQL

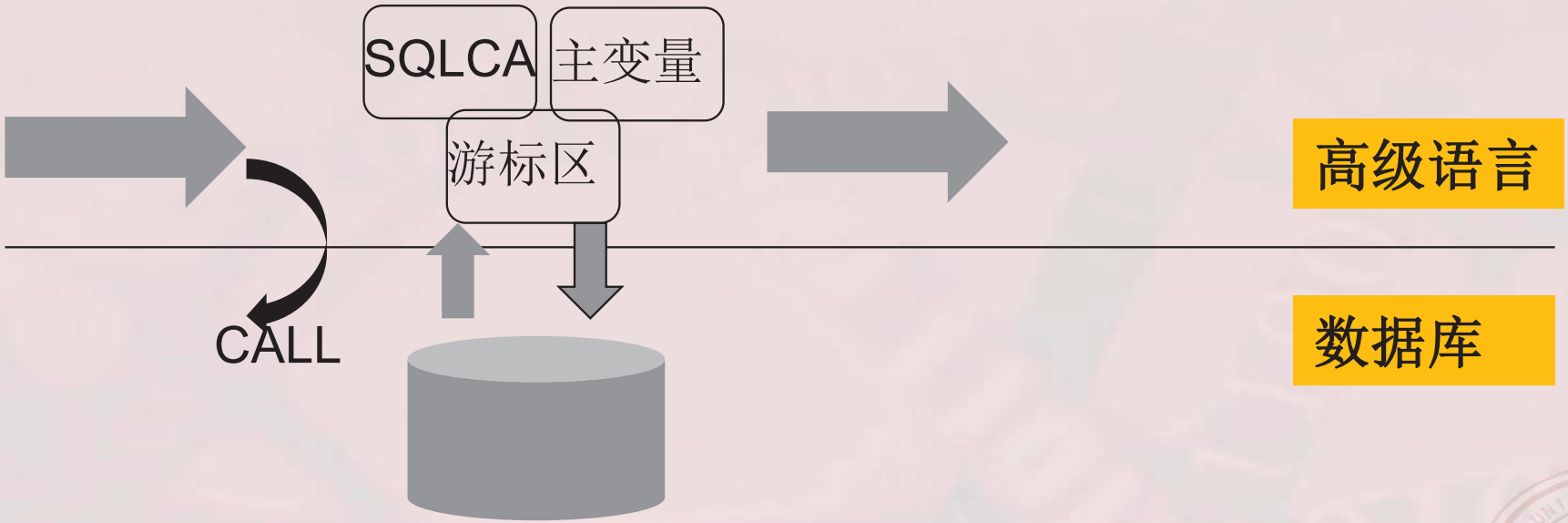
2 扩展SQL语言对于过程控制的表达能力：过程化SQL

3 在一个更大的视野上，将数据库看做是一类数据源：ODBC编程

Video 12-1：过程化SQL

嵌入式SQL技术路线的问题

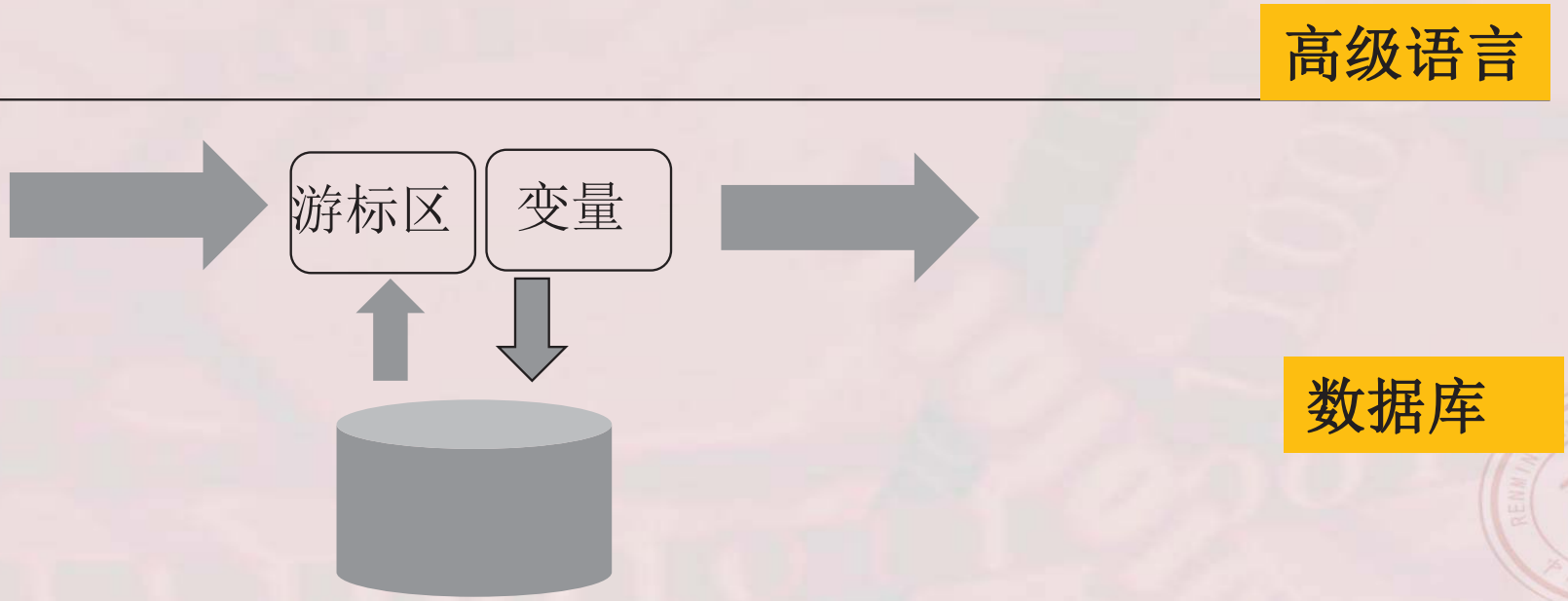
在数据库空间和高级语言空间之间的通讯，需要额外的代价



过程化SQL技术路线的优势

尽量减少系统之间的通讯和协同！

如果把嵌入式SQL中的数据库看作是“仆人”的话，那么这个仆人现在要自己做“主人”了



过程化SQL的基本结构

- 过程化SQL （也称PL/SQL）是对SQL的扩展
  - 基本结构是块（Block）
    - 块可以嵌套 （调用）
    - 每个块完成一个逻辑操作
  - 增加了变量、常量等的定义语句
  - 增加了变量赋值语句
  - 增加了流程控制语句

过程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. 执行部分

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

变量的定义

1. 变量定义
   - 变量名 数据类型 [[NOT NULL]:=初值表达式]或
   - 变量名 数据类型 [[NOT NULL] 初值表达式]
2. 举例

```
DECLARE                    /*定义变量*/
    totalDepositOut Float  := 0;
    totalDepositIn Float := 0;
    inAccountnum INT NOT NULL;
```

常量的定义

1. 常量定义
   - 常量名 数据类型 CONSTANT :=常量表达式
   - 常量必须要给一个值，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
2. 举例：
   - errorMsg string CONSTSNT := “Nested SQL error”

赋值语句

1. 赋值语句
   - 变量名称 :=表达式
   - SET 变量名称 = 表达式

2 举例：
   - SET Sno=Sno+1;

流程控制语句

1. 条件控制语句
2. 循环控制语句
3. 错误处理

流程控制（续）

1. 条件控制语句

IF-THEN，IF-THEN-ELSE和嵌套的IF语句

（1）IF condition THEN
Sequence_of_statements;
END IF;

（2）IF condition THEN
Sequence_of_statements1;
ELSE
Sequence_of_statements2;
END IF;

（3）在THEN和ELSE子句中还可以再包含IF语句，即IF语句可以嵌套

流程控制（续）

2. 循环控制语句

（1）简单的循环语句LOOP

```
LOOP
  Sequence_of_statements;
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语句块能够结束

流程控制（续）

2. 循环控制语句（续）

（2）WHILE-LOOP

```
WHILE condition LOOP
        Sequence_of_statements;
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

流程控制（续）

2. 循环控制语句（续）

（3）FOR-LOOP

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

流程控制（续）

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

```
1  /*判断某个学生是否满足毕业要求 并修改student表中的graduate属性*/
2  /*创建PROCEDURE
3  输入: Fail_LIMIT 挂科次数限制action
4  AVG_GRADE_LIMIT 平均GRADE限制
5  */
6  DELIMITER //
7  CREATE PROCEDURE GRADUATE (IN Fail_LIMIT int, IN AVG_GRADE_LIMIT int)
8  BEGIN
9      DECLARE FAIL int;
10     DECLARE AVG_GRADE int;
11     DECLARE Sno int DEFAULT 0;
12     loop_label: LOOP
13         SELECT COUNT(*) INTO FAIL FROM SC WHERE SC.Sno = Sno AND SC.Grade < 60;
14         SELECT avg(Grade) INTO AVG_GRADE FROM SC WHERE SC.Sno = Sno AND SC.Grade >= 60;
15         IF FAIL <= Fail_LIMIT AND AVG_GRADE >= AVG_GRADE_LIMIT THEN
16                 UPDATE Student SET graduate = 'success' WHERE Student.Sno = Sno;
17         ELSE
18                 UPDATE Student SET graduate = 'fail' WHERE Student.Sno = Sno;
19         END IF;
20         SET Sno=Sno+1;
21         IF Sno>=100 THEN
22             LEAVE loop_label;
23         END IF;
24     END LOOP;
25 END;//
26
27 CALL GRADUATE(1, 70); //
28 SELECT * FROM student;
29
```

定义部分

执行部分