数 据 库 系 统 原 理

党德鹏

ddepeng@bnu.edu.cn

数据库系统原理

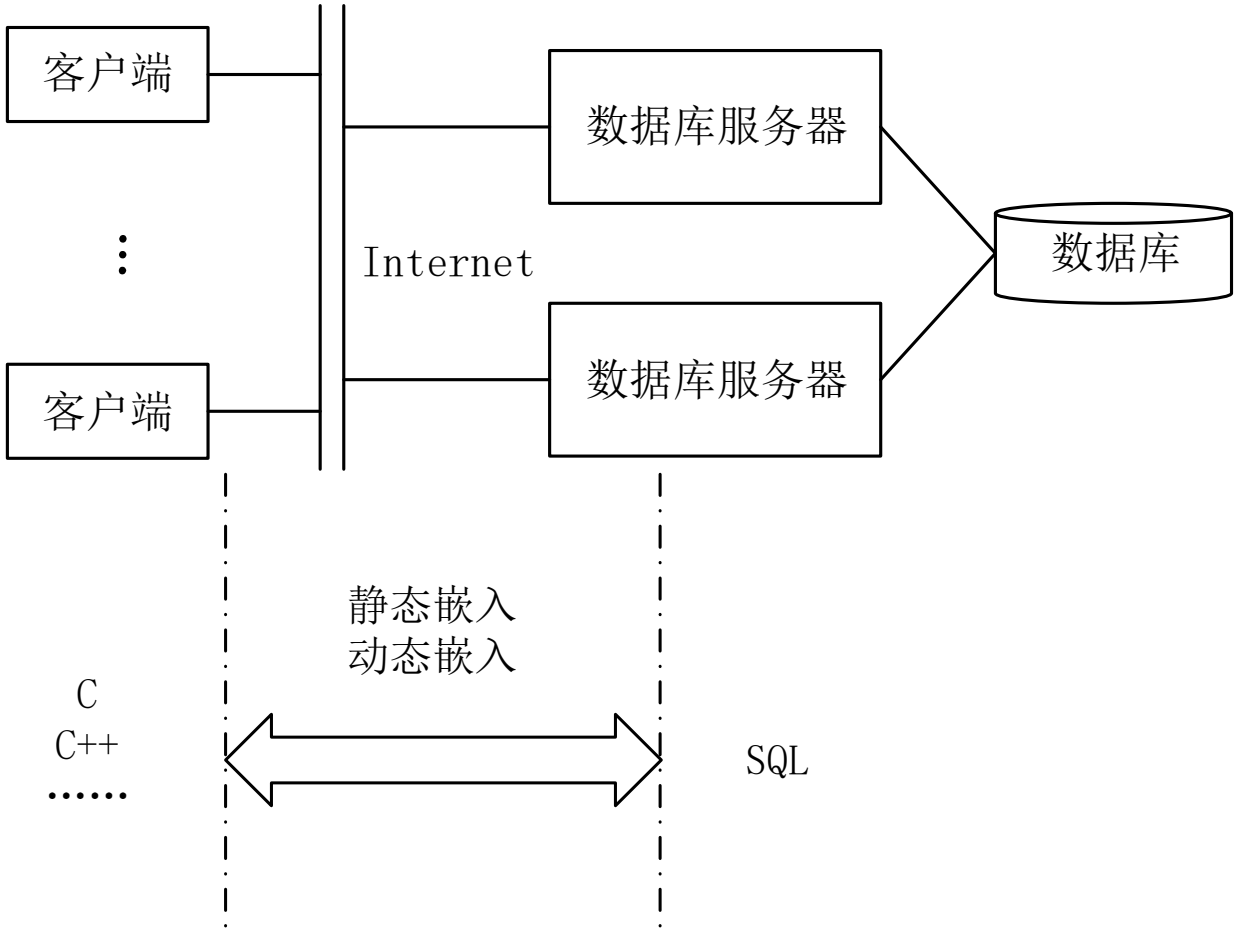
第四章 PG应用

第四章：PG应用

- 应用体系结构
- 嵌入式pgSQL
- JDBC编程
- PL/pgSQL

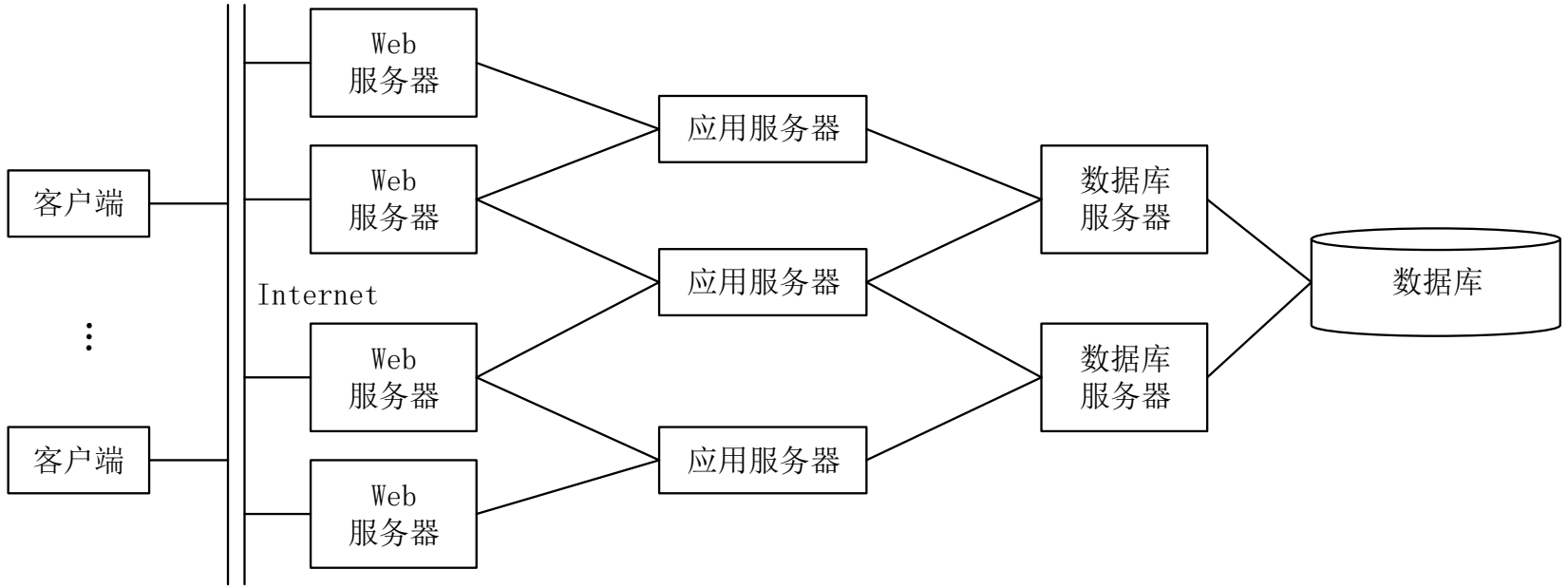
- C/S
- B/S

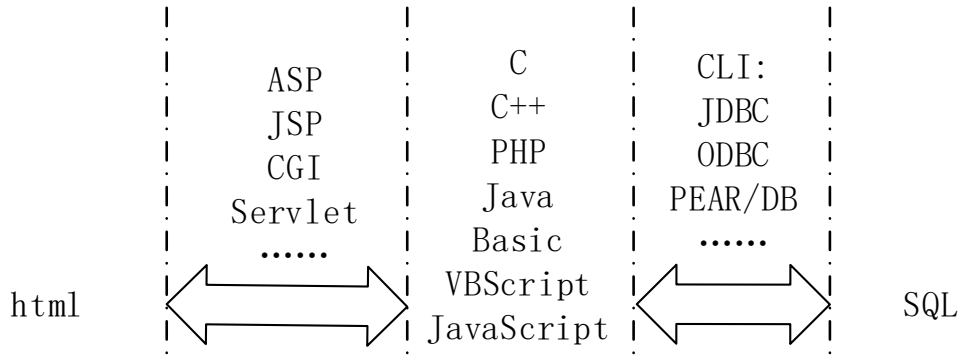
C/S



高级语言与SQL混合

B/S





html与高级语言混合　　高级语言与SQL混合

第四章：PG应用

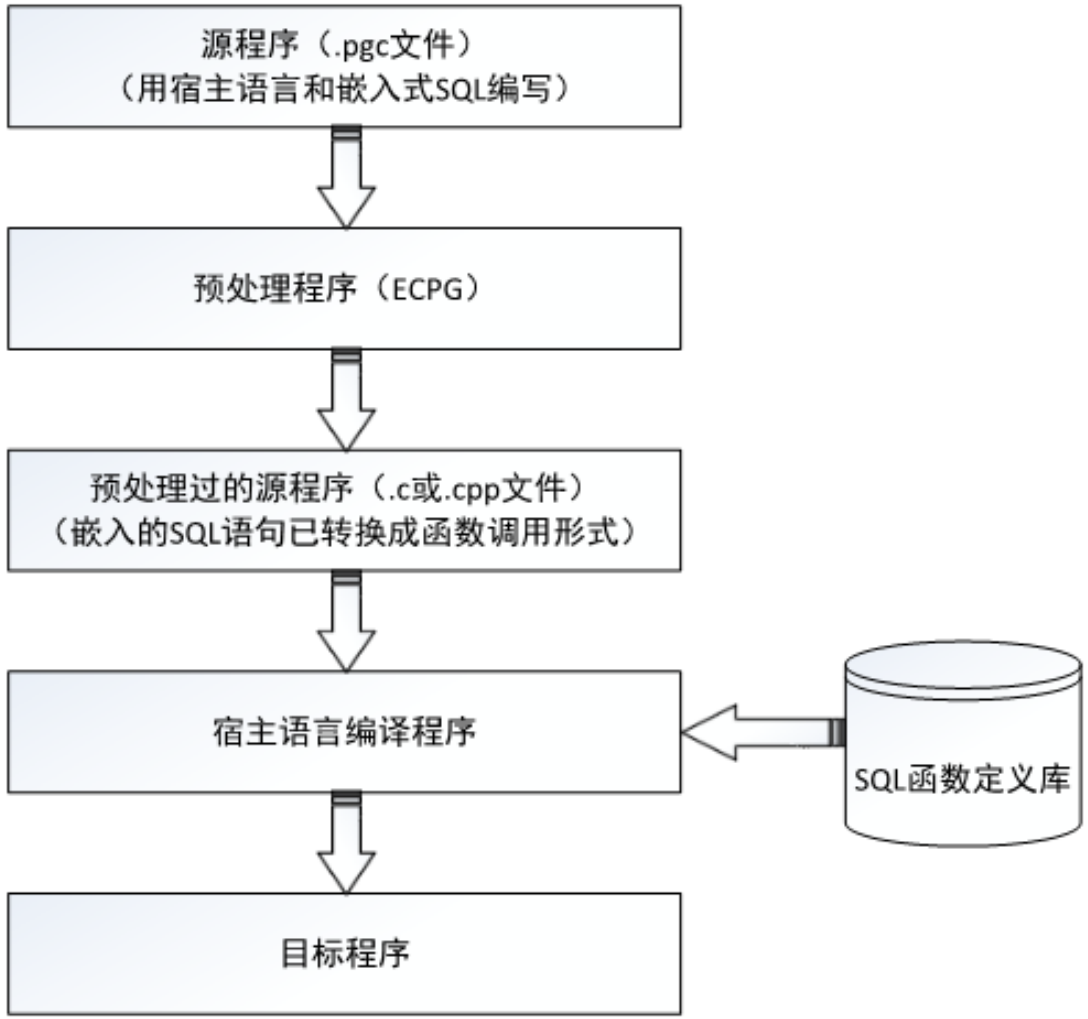
4.1 应用体系结构

4.2 嵌入式pgSQL

4.3 JDBC编程

4.4 PL/pgSQL

- pgSQL不仅可以作为独立的数据语言直接以交互的方式使用;
- pgSQL还可以作为子语言嵌入在宿主语言中使用，这里所说的宿主语言就是指我们常见的高级程序设计语言，如C、C++语言等。


源程序（.pgc文件）
（用宿主语言和嵌入式SQL编写）
预处理程序（ECPG）
预处理过的源程序（.c或.cpp文件）
（嵌入的SQL语句已转换成函数调用形式）
宿主语言编译程序
SQL函数定义库
目标程序

- 连接数据库：高级语言需要与数据库服务器建立连接。
- 嵌入识别问题：宿主语言的编译程序不能识别pgSQL语句，所以首要的问题就是要解决如何区分宿主语言的语句和pgSQL语句;
- 宿主语言与pgSQL语言的数据交互问题：pgSQL语句的查询结果必须能够交给宿主语言处理,宿主语言的数据也要能够交给pgSQL语句使用;
- 宿主语言的单元组与pgSQL的多元组的协调问题：宿主语言一般一次处理一条元组，而pgSQL常常处理的是行的集合，这个矛盾必须解决。

- 加载驱动程序
- 给出数据库服务器地址、端口、数据库名、用户名、口令，以及连接协议。

- CONNECT TO dbname[@hostname][:port] [USER user-name USING password];

- 前缀 “EXEC SQL”， 和结束标志分号“;”。
- EXEC SQL INSERT INTO exmaminee

  VALUES('218811011028','赵丽颖','女','18','文学院');

```
EXEC SQL BEGIN  DECLARE SECTION;
                …
                共享变量说明
                …
EXEC SQL END  DECLARE  SECTION;
```

```
EXEC SQL BEGIN DECLARE SECTION;
        char bs_eeid[12];
        char bs_same[20];
        int bs_eeage;
EXEC SQL END DECLARE SECTION;
```

```
UPDATE  examinee

SET  eeid = : bs_eeid

WHERE  eeage = : bs_eeage;
```

- DECLARE CURSOR
- OPEN
- FETCH
- CLOSE

- 如果是INSERT、DELETE、UPDATE、DDL/DPL语句，只要加上前缀标识“EXEC SQL”和结束标志“;”，就能嵌入在宿主语言程序中使用。

- SELECT语句，如果可以确定查询结果肯定至多包含一个元组

```
EXEC SQL SELECT eid, ename, etype
        INTO  :id,:name,:type
        FROM exampaper
        WHERE eid = :gid;
```

```
#define NO_MORE_TUPLES !(strcmp(SQLSTATE, ″ 02000 ″ ))
void queryachieve( )
  { EXEC SQL BEGIN DECLARE SECTION;
                char seeid[12],   seid[10];
                int sachieve;
                char SQLSTATE[6];
     EXEC SQL END DECLARE SECTION;
     EXEC SQL DECLARE quecur CURSOR FOR SELECT eeid, eid, achieve    FROM eeexam;
     EXEC SQL OPEN quecur;
     while (1)
         {  EXEC SQL FETCH FROM quescr
                        INTO :seeid, :seid, :sachieve;
            if （NO_MORE_TUPLES） break;
            printf (”%s, %s, %d”, seeid, seid, sachieve);
          }
     EXEC SQL CLOSE quecur;
  }
```

EXEC SQL DECLARE〈游标名〉SCROLL CURSOR FOR
〈SELECT语句〉;

EXEC SQL FETCH { NEXT | PRIOR | FIRST | LAST | RELATIVE <整数> | ABSOLUTE <整数> } FROM〈游标名〉 INTO〈共享变量表〉;

EXEC SQL PREPARE〈动态pgSQL语句名〉FROM〈共享变量或字符串〉；

EXEC SQL EXECUTE〈动态pgSQL语句名〉；

EXEC SQL EXECUTE IMMEDIATE〈共享变量或字符串〉

EXEC SQL EXECUTE〈 动态pgSQL语句名 〉USING〈共享变量〉

```
#include<stdio.h>
int main()
{
  EXEC SQL CONNECT TO postgres@192.168.1.101:5432 USER postgres USING '123456';
  EXEC  SQL  BEGIN  DECLARE  SECTION;
      char  *tt="update exam set eeyear=eeyear+? where eeid=?;";
      char  *ttc="commit;";
      int ii=199;
      int jj=20000;
  EXEC  SQL  END  DECLARE  SECTION;
  EXEC SQL PREPARE mmtt from :tt;
  EXEC SQL EXECUTE mmtt USing :jj,:ii;
  EXEC SQL EXECUTE IMMEDIATE :ttc;
  EXEC SQL DISCONNECT;
  return 0;
  }
```

	eeid [PK] integer	eename character(10)	eeyear integer
1	166	aaaa	1996
2	189	north	2001
3	198	south	1998
4	199	west	2000
5	200	bbbb	1999
6	222	east	1999
7	269	cccc	1992
8	369	dddd	1994
*			

	eeid [PK] integer	eename character(10)	eeyear integer
1	166	aaaa	1996
2	189	north	2001
3	198	south	1998
4	199	west	22000
5	200	bbbb	1999
6	222	east	1999
7	269	cccc	1992
8	369	dddd	1994
*			

第四章：PG应用

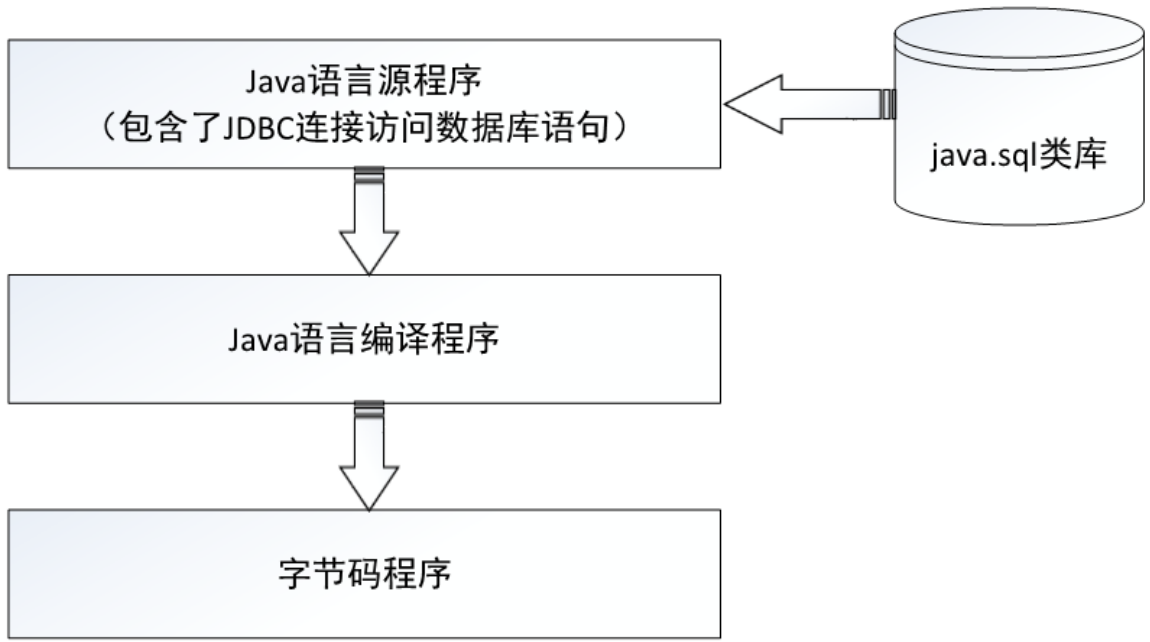
4.1 应用体系结构

4.2 嵌入式SQL

4.3 JDBC编程

4.4 PL/pgSQL

- JDBC的目标是使应用程序开发人员使用JDBC可以连接任何提供了JDBC驱动程序的数据库系统。
- JDBC为访问不同的数据库提供了一种统一的途径。
- JDBC定义了用来访问数据库的标准Java类库，在java.sql类包中，使用这个类库可以以一种标准的方法、方便地访问各种关系数据库。


Java语言源程序
（包含了JDBC连接访问数据库语句）
java.sql类库
Java语言编译程序
字节码程序

- 四个步骤:
  - 加载JDBC驱动程序
  - 建立与数据库的连接
  - 进行数据库操作
  - 关闭相关连接

```
public static void JDBCexample(String username, String password)
{
      try {
         Class.forName (“org.postgresql.Driver”);
         Connection myjconn =
    DriverManager.getConnection(“jdbc:postgresql://localhost:5432/postgres”, username, password);
         Statement mystmt =myjconn.createStatement();
         …… Do Actual Work  访问数据库……
         mystmt.close();
         myjconn.close();
      }
      catch (SQLException sqle) {
           System.out.println("SQLException : " + sqle);
       }
}
```

- **修改**

```
try {
      mystmt.executeUpdate("insert into eeexam values('218811011116', '0201020001', 80)");
}
catch (SQLException sqle) {
   System.out.println("Could not insert tuple. " +sqle);
}
```

- **查询**

```
ResultSet myrset = mystmt.executeQuery("select eeid, avg(achieve)  from eeexam  group by eeid");
while (myrset.next()) {
        System.out.println(myrset.getString("eeid") + "  " + myrset.getFloat(2));          }
```

```
PreparedStatement pmyStmt=myjconn.prepareStatement("insert into eeexam  values(?,?,?) " );
        pmyStmt.setString(1, "2188110111116");
        pmyStmt.setString(2, "0201020001");
        pmyStmt.setInt(3, 80);
        pmyStmt.executeUpdate();
        pmyStmt.setString(1, "2188110111117");
        pmyStmt.executeUpdate();
```

```
ResultSetMetaData myrsmd = myrset.getMetaData();
For(int i=1;i<=myrsmd.getColumnCount();i++)
{
        System.out.println(myrsmd.getColumnName(i));
        System.out.println(myrsmd.getColumnTypeName(i));
}
```

```
DatabaseMetaData mydbmd = myjconn.getMetaData();

ResultSet mydmrset = mydbmd.getCollumns(null, null, “eeexam”, “%”);

While (mydmrset.next())

{

 System.out.println(mydmrset.getString(“COLUMN_NAME”), mydmrset.getString(“TYPE_NAME”));

}
```

```
<html>
<head>
<title>This is a test of Servlet JDBC.</title>
</head>
<body>
<h3><font size=16>
```

This is a test of JSP JDBC.

```
</font> </h3>
</body>
</html>
```

第四章 PG应用

4.1 应用体系结构

4.2 嵌入式SQL

4.3 JDBC编程

4.4 PL/pgSQL

- PL/pgSQL
- 存储函数
  - 由于存储函数不需要额外的语法分析步骤，因而运行效率高。
  - 客户端不需要的中间结果无需在服务器端和客户端来回传递，降低了客户机和服务器之间的通信量。客户机上的应用程序只需向服务器发出存储函数的名字和参数，就可以让调用执行存储函数，只有最终处理结果才返回客户端。
  - 便于实施业务规则。通常把业务规则的计算程序写成存储函数，由数据库管理系统集中管理，方便进行维护。

- PL/pgSQL：
  - 过程
  - 块结构

- 变量:

  变量名 数据类型[:=初始表达式]

- 常量:

  常量名 数据类型 CONSTANT :=常量表达式

- 3. 赋值语句

  变量名称: =表达式

IF *逻辑表达式* THEN
    语句;
END IF;

IF *逻辑表达式*1 THEN
    语句;
ELSE
    语句;
END IF;

```
IF 逻辑表达式1 THEN
    语句序列1;
ELSE
    IF 逻辑表达式2 THEN
        语句序列2;
    END IF;
END IF;
```

IF *逻辑表达式1* THEN

语句序列1;

[ELSIF *逻辑表达式2* THEN

语句序列2;

[ELSIF *逻辑表达式3* THEN

语句序列3;

...]]

[ELSE

语句序列n]

END IF;

```
LOOP
    循环体;
END LOOP;
```

WHILE *逻辑表达式* LOOP

循环体;

END LOOP;

```
FOR count IN [REVERSE] bound1 .. bound2 [BY expression] LOOP
    循环体;
END LOOP;
```

- EXIT [label] [WHEN expression];

- CONTINUE [label] [WHEN expression];

```
CREATE OR REPLACE FUNCTION  add(INT, INT)
    RETURNS INT
    AS
    $ $
    DECLARE intsum INT;
    BEGIN
            SELECT $1 + $2 INTO intsum;
            RETURN intsum;
    END;
    $ $
    LANGUAGE plpgsql
```

- RETURN expression;

执行存储函数

SELECT 函数名 ([参数1, 参数2……]);

PERFORM 函数名 ([参数1, 参数2……]);

删除存储函数

DROP FUNCTION 函数名 ([参数1, 参数2……]);