本章内容：

·了解存储过程的优点

·学会如何创建存储过程

·学会如何调用存储过程

·学会如何创建函数

·学会如何调用函数

·学会如何创建和使用触发器

·存储过程：在Oracle中，可以在数据库中定义一个PLSQL的子程序，这种程序被保存在数据库中，存储在数据字典中，也可以进行不同用户之间的共享。

·存储过程的优点：

·存储过程执行在数据库服务器中，执行速度非常快

·执行一次之后，代码会保留在高速缓存区，以后执行只要调用高速缓存区的编译好的代码，系统的性能可以得到很大的提高。

·安全性很高

·可以自动的完成预先设置的任务。

编写一个存储过程：向一个表中添加数据

|  |
| --- |
| create or replace procedure ins\_dh02t\_person *--给存储过程命名*  (pid in number,pname in varchar2,psex in number,pbir in date)*--在存储过程中，传入的参数需要使用in 返回的参数使用out*  as  begin  insert into dh02t\_person (p\_id,p\_name,p\_sex,p\_bir) values (pid,pname,psex,pbir);  commit;  exception  when others then  rollback;*--如果出现异常进行回滚*  end ins\_dh02t\_person;*--结束此存储过程* |

参数的命名必须符合规范，在一个存储过程中，可以不传递参数，也可以传递一个或多个参数，和函数一样，参数的类型有三种，in 、out、in out

in：表示传入参数

out：表示返回的参数

in out：表示即使传入的参数，也是返回的参数（一般情况下，这种一般不会使用）

存储过程结束时要跟上存储过程的名称。

一般情况下，如果有返回参数，我们一般是在如果没有异常的情况下返回1，否则返回-1。

|  |
| --- |
| create or replace procedure ins\_dh02t\_person01 *--给存储过程命名*  (pid in number,pname in varchar2,psex in number,pbir in date,bool out number)*--在存储过程中，传入的参数需要使用in 返回的参数使用out*  as  begin  insert into dh02t\_person (p\_id,p\_name,p\_sex,p\_bir) values (pid,pname,psex,pbir);  bool := 1;  commit;  exception  when others then  bool := -1;  rollback;*--如果出现异常进行回滚*  end ins\_dh02t\_person01;*--结束此存储过程* |

如何去调用存储过程的问题：

JDBC调用存储过程：

|  |
| --- |
| package com.wanczy.jdbcDemo03;  import java.sql.CallableStatement;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.SQLException;  import com.wanczy.demo02.GetIdk;  import com.wanczy.demo02.PublicMethod;  public class JDBCDemo01 {  //配置驱动程序  public static final String DRIVER = "oracle.jdbc.driver.OracleDriver";  //配置数据库连接池  public static final String URL = "jdbc:oracle:thin:@172.26.64.44:1521:orcl";  //数据库用户名  public static final String USERNAME = "jjm";  //数据库密码  public static final String PASSWORD = "jjm";  public static void main(String[] args) {  Connection con = null;  CallableStatement cstat = null;//专门用于调用存储过程和函数的操作接口  String sql = "{call ins\_dh02t\_person01(?,?,?,?,?)}";  try {  Class.forName(DRIVER);//本程序中加载驱动  con = DriverManager.getConnection(URL,USERNAME,PASSWORD);//取得数据库连接对象  cstat = con.prepareCall(sql);//参数为的{call 存储过程名称+(?,?,?,?,?)}  cstat.setInt(1, GetIdk.getId(con, "dh02t\_person", "p\_id"));  cstat.setString(2, "王二");  cstat.setInt(3, 1);  cstat.setDate(4, new java.sql.Date(PublicMethod.getDate("1990-09-09").getTime()));  cstat.registerOutParameter(5, java.sql.Types.INTEGER);//设置存储过程返回类型  cstat.execute();  int x = cstat.getInt(5);  System.out.println("结果为："+x);    } catch (Exception e) {  e.printStackTrace();  }finally{  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  } |

删除存储过程：drop procedure 存储过程名称

·函数：

·具有返回值的子程序

·里面存在return语句

·存储在数据库中，可以由任意的PLSQL程序区调用函数

函数的两个组成部分：

·说明

以function开始，以return结束

·主体

以is 开始，以end 或者end 函数名结束

范例：新增数据时用于获取表的主键值

|  |
| --- |
| create or replace function getIDk(params varchar2)  return number  is  *--定义变量*  next\_idk number;  begin  select max(p\_id)+1 into next\_idk from dh02t\_person;  return next\_idk;  exception  when others then  return 0;  end getIDk; |

以上的程序可能有问题，如果存在并发操作的时候，可能会取得相同的值。一般情况下， 去取得主键值的方法应该要新建一个表格，用于存储下一主键值。在一个系统中，所有表格都可以共用一个主键的获取方法。

|  |
| --- |
| create or replace function get\_idk\_new (params varchar2)  return number;  is  next\_idk number(18);  begin;  *--跟上forupdate 表示此表已经锁住，这次回话已经在执行，只有当此次会话结束，其他的连接才可以操作此表*  select idk into next\_idk from table\_idk for update;    update table\_idk set idk = (next\_idk+1);  commit ;  return next\_idk;  exception  when others then  rollback;  return 0;  end; |

使用JDBC去调用此函数。

|  |
| --- |
| package com.wanczy.jdbcDemo03;  import java.sql.CallableStatement;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;  import com.wanczy.demo02.GetIdk;  import com.wanczy.demo02.PublicMethod;  public class JDBCDemo02 {  //配置驱动程序  public static final String DRIVER = "oracle.jdbc.driver.OracleDriver";  //配置数据库连接池  public static final String URL = "jdbc:oracle:thin:@172.26.64.44:1521:orcl";  //数据库用户名  public static final String USERNAME = "jjm";  //数据库密码  public static final String PASSWORD = "jjm";  public static void main(String[] args) {  Connection con = null;  CallableStatement cstat = null;//专门用于调用存储过程和函数的操作接口  String sql = "{?=call get\_idk\_new(?)}";  try {  Class.forName(DRIVER);//本程序中加载驱动  con = DriverManager.getConnection(URL,USERNAME,PASSWORD);//取得数据库连接对象  cstat = con.prepareCall(sql);//  cstat.setString(2, "");  cstat.registerOutParameter(1, java.sql.Types.INTEGER);//设置函数返回类型  cstat.execute();  int x = cstat.getInt(1);  System.out.println("结果为："+x);  } catch (Exception e) {  e.printStackTrace();  }finally{  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  } |

在函数的定义过程中，参数应该尽量避免使用in 、out、in out

触发器：触发器其实就是一些过程，只是用于表数据的保护，当进行一些增删改的时候，触发器会自动的执行，实现数据的完整性，触发器和应用程序无关。

触发器可以分为三类12种（一个表最多可以拥有12个触发器）

·insert

·update

·delete

比如 insert操作：

·before insert

·after insert

·before insert for each row

·after insert for each row

触发器的语法格式：

create or replace trigger 触发器名称{before | after } {insert|update |delete } on 表名 （for each row）

begin

你想要干的事情

end 触发器名称 ；

触发器编写规则：

·代码量必须小于32K

·触发器中可以使用DML语句，但是不能使用DDL语句

·会对一些long类型的限制，不能再触发器中声明此种类型的变量，也不能给此种变量赋值。

·执行规则：先执行before 再执行after

现有一个学生需要插入到数据表中，但是此学生对应的班级没有在数据表中，这个时候我们需要在插入这个学生之前先将班级插入到数据库中。这个时候在学生表中写一个触发器，在插入学生之前，先判断此学生对应的班级是否存在，如果存在则啥都不干，如果不存在，则先插入班级。

|  |
| --- |
| create or replace trigger tri\_stuinfo\_insert\_before  before insert on studentinfo for each row  declare  v\_count number(1) := 0;  begin  select count(c\_id) into v\_count from classinfo where c\_id = :new.c\_id;*--new 表示即将要插入到studentinfo表中的的数据的集合*  if v\_count=0 then  insert into classinfo(c\_id) values(:new.c\_id);  end if;  end tri\_stuinfo\_insert\_before; |

现在关于：new 的话只有在新增和修改的触发器中才会有

关于：old 的话只有在删除和修改的时候才会有

比如说我现在要删除班级，就意味着这个班级不再存在学生了，所以说在删除班级之前需要将次班级的学生全部删除。

|  |
| --- |
| create or replace trigger classinfo\_delete\_before before delete on classinfo for each row  begin  delete studentinfo where c\_id = :old.c\_id;  end classinfo\_delete\_before; |

触发器的删除：drop trigger 触发器名称。