**1、本章面试题**

     什么是语句块，有什么作用

**2、知识点**

**2.1、课程回顾**

表空间管理（oracle特有，为oracle优化起了很大作用）

用户管理和授权

常用函数

**2.2、本章重点**

1,plsql语句块的作用,分类和基本语法

2,%type,%rowtype,record,table用法

3, 事务控制语言TCL（transaction control language）

**3、具体内容**

**3.1 PL/SQL语言块简介**

**3.1.1 概念：**

多个语句放在一起执行，可以使用条件判断，循环，异常处理等等，为了完成一个完整的业务功能的语句集合。

**3.1.2作用：**

       如果不使用PL/SQL语言块，oracle一次只能处理一条SQL语句。每条SQL语句都导致客户（client）向服务器(server)调用，从而在性能上产生很大的开销，尤其是在网络操作中。如果使用 PL/SQL,一个块中的语句作为一个组，导致客户向服务器的一次调用，减少网络转输

jdbc(java database connection) 客户端192.168.1.110 网线 服务器192.168.1.120

执行10条语句，不适用语句块就会发10次请求，得到10次响应

使用语句块，发一次请求 得到1次响应

**3.1.3分类：**

     由基本的PL/SQL块组成PL/SQL程序，可组成不同的程序形式，它们的用途和适用性各不相同。程序形式大致有以下几种：

1、无名块：也就是没有命名的PL/SQL块，它可以是嵌入某一个应用之中的一个PL/SQL块。匿名块是出现在应用程序中的没有名字且不存储到数据库中的块。它们可以调用其他程序，却不能被其他程序调用。

类似 java 类写一个static{} 静态代码块

2、存储过程/函数：也就是命名了的PL/SQL块，它可以接收参数，并且可以重复地被调用。

有名字了，类似java中的自己写好的业务方法，编写一次可以被任何人在任何时候重复调用，不需要再去编写该业务，该方法就是一个接口，写完一个方法，写出该方法的接口文档，测试没问题的情况下，发给全组组员或者使用方法客户

3、包

命名了的PL/SQL模块，由一组相关的过程、函数和标识符组成。

类似于java中的包，当写的业务类很多时，方便分门别类，让代码模块更清晰。两个人开发的类名称重复时，可以放在不同包解决 com.aaa.a.test.TestClass

com.aaa.b.test.TestClass

4、数据库触发器：是一个与具体数据库表相关联的存储PL/SQL程序。每当一个SQL操作影响到该数据库表时，系统就自动执行相应的数据库触发器。每个表最多可以有12个触发器。

当要删除比较重要的表信息时，执行删除操作后，可以把该数据备份，该功能就可以使用触发器来实现

**3.1.4语法：**

一个基本的PL/SQL块由三部分组成：定义部分、可执行部分以及例外处理部分：

**定义部分（java中的定义变量）：**

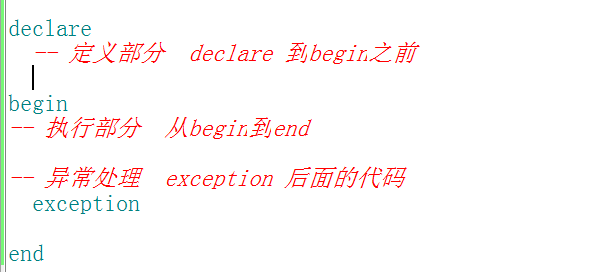
定义将在可执行部分中调用的所有变量、常量、游标和用户自定义的例外处理。这部分可以没有。

**可执行部分（java中的方法执行）：**

包括对数据库中进行操作的SQL语句，以及对块中进行组织、控制的PL/SQL语句。这部分必须存在。

**例外处理部分（java中的 try catch finally）：**

对可执行部分中的语句，在执行过程中出错或出现非正常现象时所做的相应处理。这部分可以没有。



**3.2 语句块的数据类型**

等同于oracle 的数据类型（其实多于oracle的数据类型）

字符类型 char varchar2

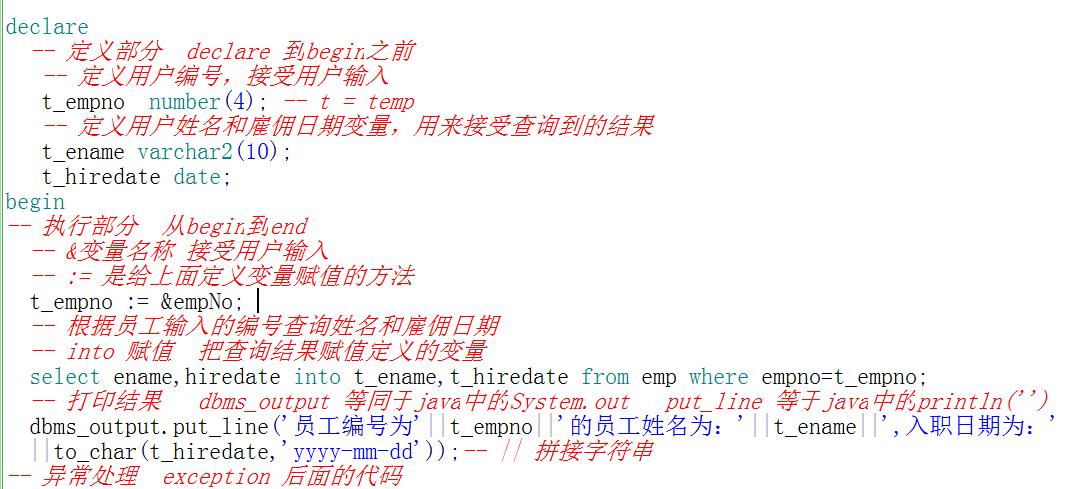
数值类型 number (长度) number(m,n)

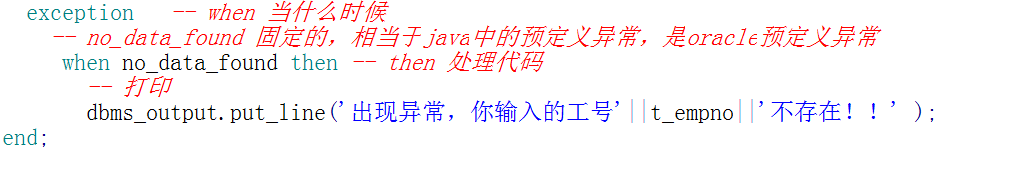
时间类型 date timstamp

大数据类型 clob blob

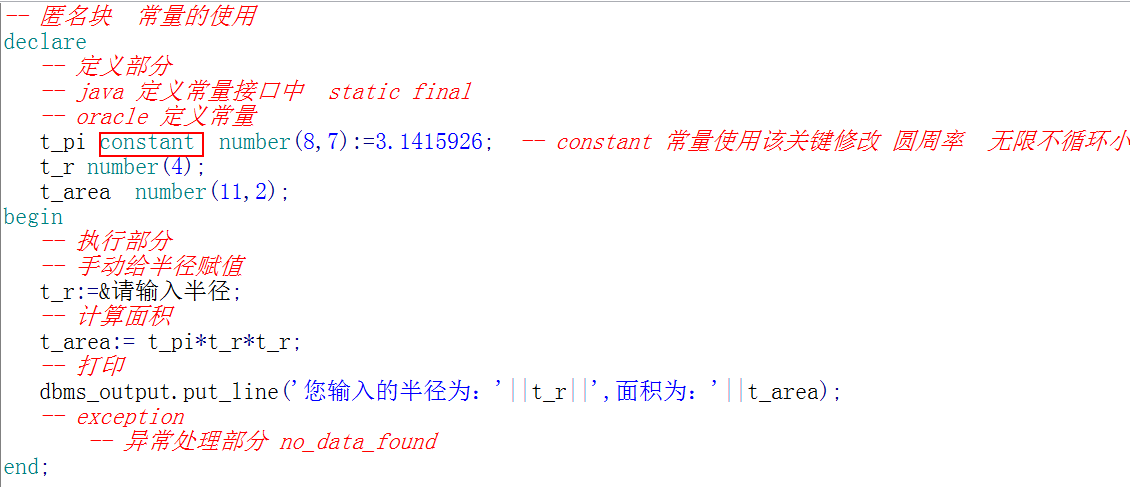
**3.3 匿名块示例（是后面课程一个铺垫）**

接收用户输入的员工编号，查询并输出该员工的姓名和雇佣日期，处理用户输入的员工编号不存在的异常（按上面所讲语法写出示例，注释每行作用，讲解注释，赋值）



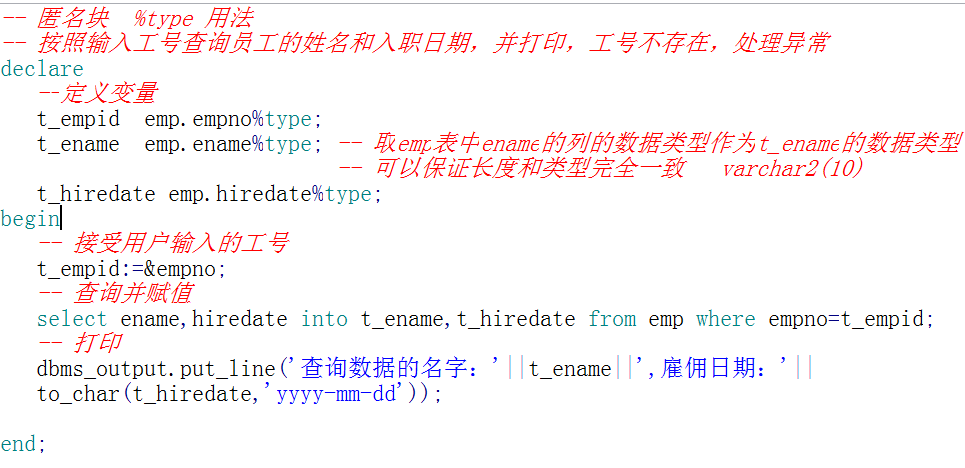


**3.4 常量的使用**



**3.5 %type %rowtype record table用法**

**%type 定义一个变量的数据类型与已经定义的变量数据类型（已经存在的表）一致；**



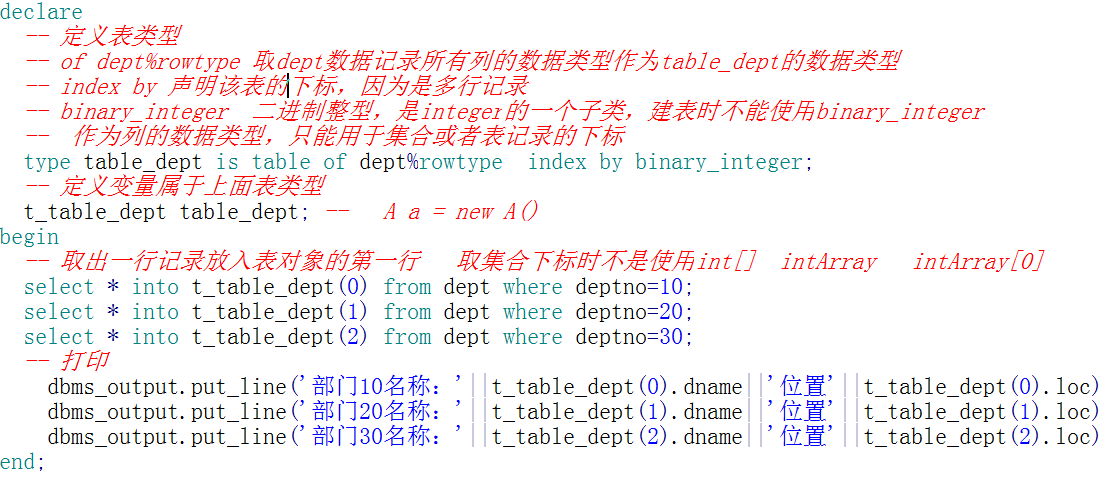
**%rowtype 可以使一个变量的数据类型和一个表中的数据记录的各个列的数据类型一致；**



**record 自定义记录类型（既不是取单个列也不是取整行）**



**table 定义记录表类型(record只能取一行，table可以取多行记录，相当于数据副本)**



**3.6 异常处理**

**3.6.1 预定义异常**

像java中一样，pl/sql预先定义了一些错误异常，当编写语句时违反了oracle规定或者超出了系统规定的限制，就会出现事先定好的错误。

ArithmeticException NullPointException CassCaseException SqlException

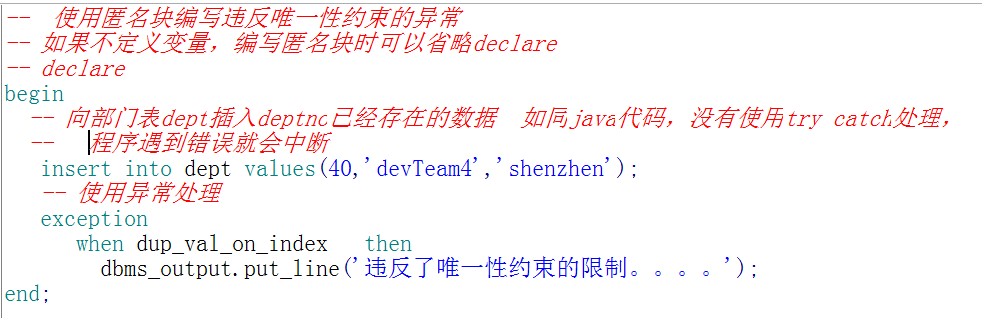
IOException FileNotFoundException ClassNotFoundException.....







**使用匿名块编写违反唯一性约束的异常**



**3.6.2 自定义异常**

类似于java的自定义异常，在oracle中语句块编写过程中，为了具体的业务规则，编程需要自己定义一些异常，来帮助你完成你的具体业务。

java 自定义异常 try catch finally throw throws

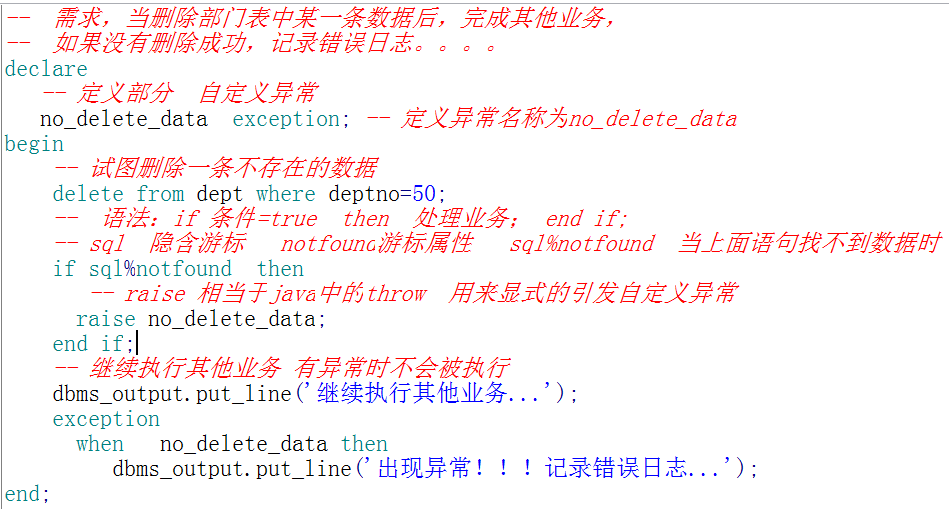
throw new Exception('自定义异常的描述')； 自己编写列继承Exception

throw new RuntimeException('自定义异常的描述'); 方法声明 throws

需求，当删除部门表中某一条数据后，完成其他业务，如果没有删除成功，记录错误日志。。。。

（oracle删除不存在的数据时，没有预定义异常，所以要使用自定义异常）

delete from dept where deptno=60;



**3.7 事务控制语言（TCL transaction control language）**

**事务四个特性：**

原子性(A): 数据操作时的一个执行单元（包含多条更新操作），要么全做，要么全不做。

例子：假如转账操作A->B转钱，1:从A账号中扣钱，2:给B账号加钱 1和2要么全做，要么全不做

一致性(C)：事务的执行结果必须从一个一致性转移为另一个一致性。

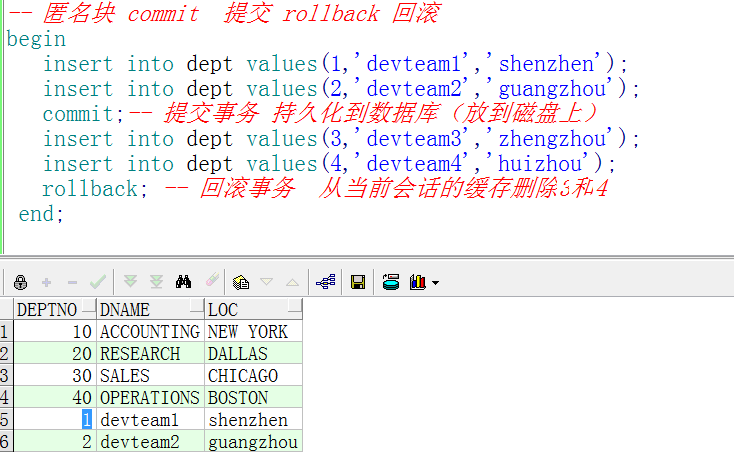
例子：假如转账操作A->B转钱 A账号有10W B账号有10W A和B的账号钱数总和为20W，A给B转钱1W A账号为9W B账号为11W 转账后总钱数依然是20W。

隔离性(I):一个事务的执行不能被其他事务干扰。

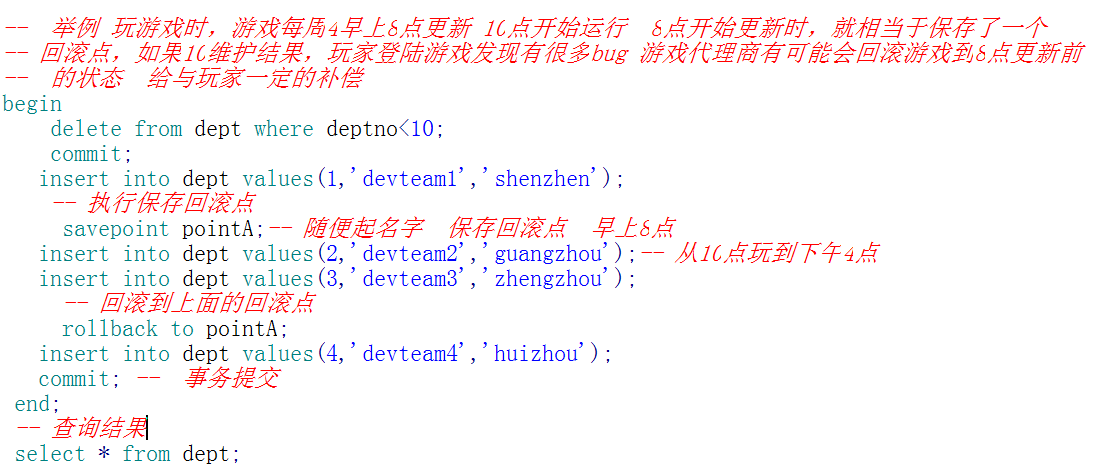
例子：A->B转钱 C->D转钱 不能从A账号扣钱后加到D账号，或者其他不合逻辑的操作

持久性(D):事务一旦提交，数据一旦持久化，就不会再发生变化，不受外界干扰。

**commit, rollback,savepoint, rollback to 实现**



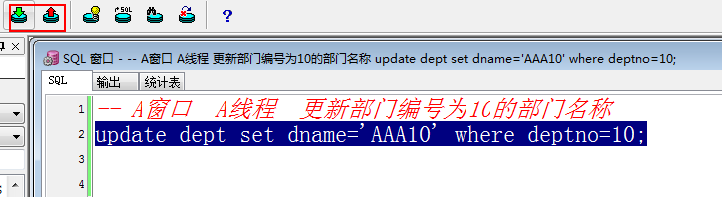
**savepoint, rollback to**



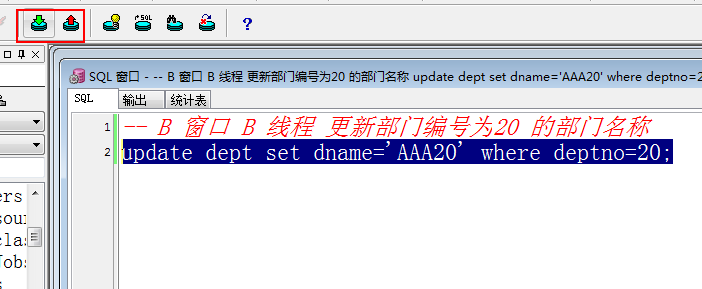
**3.8 死锁的原理及示例**

一般会出现在写业务过程中，A线程拥有A1资源的锁，B线程拥有B1资源的锁，A在拥有A1的同时想要B1资源的锁（不叫死锁），B在拥有B1的同时想要A1资源的锁，构成了死锁。

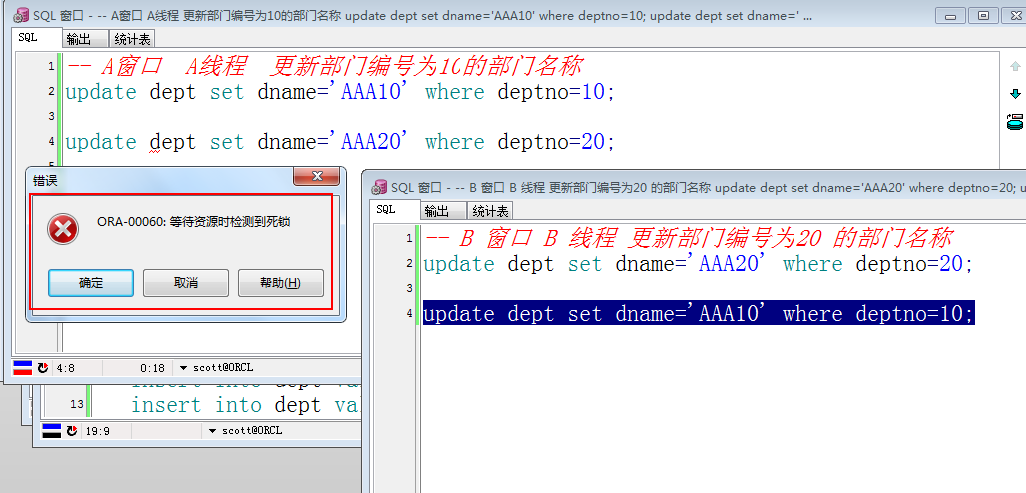
**在oracle中死锁的实例：**



不提交也不回滚说明A窗口获取更新部门为编号10的锁



不提交也不回滚说明B窗口获取更新部门为编号20的锁



**如何避免死锁(应用中)**

1，避免同一个线程锁定多个资源

2， 不同线程使用相同的加锁顺序（可以使用zookeeper解决）

3， 使用定时锁（一种方案springcloud分布式锁的，用redis的带过时锁）

4， 使用算法检查死锁（拿锁前做检查，如果有线程拿着没有释放，不再去拿资源的锁）

**4、本章总结**

**4.1 总结本章知识点**

**4.2 面试题答案**

**4.3 预习下一章重点**