# 注意事项：

曾天文老师

陈贵明老哥

陈金边老师

去除重复行 思想有问题

空值不能运算 NULL

对数据库的DML基本操作参考：Test.sql和SQL的作业

局部变量以”l\_”打头，全局变量以”g\_”打头，常量以”c\_”打头，定义游标以”csr\_”打头，游标记录变量以”rec\_”打头。

# ACID

⑴ 原子性（Atomicity）

　　原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，这和前面两篇博客介绍事务的功能是一样的概念，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。

⑵ 一致性（Consistency）

　　一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。

　　拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A和B之间如何转账，转几次账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性。

⑶ 隔离性（Isolation）

　　隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

　　即要达到这么一种效果：对于任意两个并发的事务T1和T2，在事务T1看来，T2要么在T1开始之前就已经结束，要么在T1结束之后才开始，这样每个事务都感觉不到有其他事务在并发地执行。

　　关于事务的隔离性数据库提供了多种隔离级别，稍后会介绍到。

⑷ 持久性（Durability）

　　持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

　　例如我们在使用JDBC操作数据库时，在提交事务方法后，提示用户事务操作完成，当我们程序执行完成直到看到提示后，就可以认定事务以及正确提交，即使这时候数据库出现了问题，也必须要将我们的事务完全执行完成，否则就会造成我们看到提示事务处理完毕，但是数据库因为故障而没有执行事务的重大错误。

以上介绍完事务的四大特性(简称ACID)，现在重点来说明下事务的隔离性，当多个线程都开启事务操作数据库中的数据时，数据库系统要能进行隔离操作，以保证各个线程获取数据的准确性，在介绍数据库提供的各种隔离级别之前，我们先看看如果不考虑事务的隔离性，会发生的几种问题：

1. **函数的使用**

1.大小写转换函数：

LOWER('SQL Course') sql course

UPPER('SQL Course') SQL COURSE

INITCAP('SQL course') Sql Course

2.字符串操作函数：

函数 结果

CONCAT('Hello', 'World') HelloWorld

SUBSTR('HelloWorld',1,5) Hello

LENGTH('HelloWorld') 10

INSTR('HelloWorld', 'W') 6

LPAD(salary,10,'\*') \*\*\*\*\*24000

RPAD(salary, 10, '\*') 24000\*\*\*\*\*

TRIM('H' FROM 'HelloWorld') elloWorld

TRIM(' HelloWorld') HelloWorld

TRIM('Hello World') Hello World

## 3.数字操作函数：

函数 结果

ROUND(45.926, 2) 45.93

TRUNC(45.926, 2) 45.92

MOD(1600, 300) 100

## 4.日期操作函数：

MONTHS\_BETWEEN ('01-SEP-95','11-JAN-94') 19.6774194

ADD\_MONTHS ('11-JAN-94',6) 11-Jul-94

NEXT\_DAY ('01-SEP-95','FRIDAY') 8-Sep-95

NEXT\_DAY ('01-SEP-95',1) 3-Sep-95

NEXT\_DAY ('1995-09-01',1)

NEXT\_DAY (to\_date('1995-09-01','YYYY-MM-DD'),1) 3-Sep-95

LAST\_DAY('01-FEB-95') 28-Feb-95

ROUND('25-JUL-95','MONTH') 1-Aug-95

ROUND('25-JUL-95' ,'YEAR') 1-Jan-96

TRUNC('25-JUL-95' ,'MONTH') 1-Jul-95

TRUNC('25-JUL-95','YEAR') 1-Jan-95

## 5.其他常用单行函数：

NVL (expr1, expr2) 如果expr1为空，这返回expr2

NVL2 (expr1, expr2, expr3) 如果expr1为空，这返回expr3（第2个结果）否则返回expr2

NULLIF (expr1, expr2) 如果expr1和expr2相等，则返回空

COALESCE (expr1, expr2, ..., exprn) 如果expr1不为空，则返回expr1,结束；否则计算expr2,直到找到 一个不为NULL的值 或者如果全部为NULL，也只能返回NULL 了

## 6.典型使用

   1.SELECT COUNT(1)  
      INTO l\_customer\_flag  
      FROM cux\_om\_customers\_21080 coc  
     WHERE coc.customer\_id = g\_om\_headers\_rec.customer\_id  
       AND trunc(SYSDATE) BETWEEN coc.start\_date\_active AND nvl(coc.end\_date\_active,trunc(SYSDATE));

# 2.条件表达式：

实现方法： CASE 语句 或者DECODE函数，两者均可实现 IF-THEN-ELSE 的逻辑，相比较 而言，DECODE 更加简洁。

CASE expr WHEN comparison\_expr1 THEN return\_expr1 [WHEN comparison\_expr2 THEN return\_expr2 WHEN comparison\_exprn THEN return\_exprn ELSE else\_expr] END

DECODE语句：

DECODE(col|expression, search1, result1 [, search2, result2,...,] [, default])

## 1.典型使用：

SELECT (SUM(sums.sum)) / COUNT(sums.customer\_id) AS avgscore

INTO g\_avg

FROM (SELECT coh.customer\_id,

SUM(decode(coh.order\_status,

'NEW',

'10',

'UNAPPROVED',

'5',

'APPROVED',

'5',

'CANCELLED',

'-20',

'CLOSED',

'10',

'SUBMIT',

'5',

'REJECTED')) AS SUM

FROM cux\_om\_headers\_21080 coh

GROUP BY coh.customer\_id) sums;

# 3.游标的使用和遍历

DECLARE

CURSOR cur\_cursor IS

SELECT dts.department\_id,

dts.department\_name,

dts.manager\_id,

els.last\_name,

dts.location\_id,

lts.street\_address

FROM departments\_21080 dts,

employees\_21084 els,

locations\_21084 lts

WHERE dts.manager\_id = els.employee\_id

AND dts.location\_id = lts.location\_id

and dts.department\_id = &l\_department\_id

ORDER BY dts.department\_id;

BEGIN

dbms\_output.put\_line('DEPARTMENT\_ID' || chr(9) || 'DEPARTMENT\_NAME' || chr(9) || 'MANAGER\_ID' || chr(9) || 'MANAGER\_NAME' || 'LOCATION\_ID' ||

'STREET\_ADDRESS');

FOR l\_dept\_line IN cur\_cursor

LOOP

dbms\_output.put\_line(l\_dept\_line.department\_id || chr(9) || l\_dept\_line.department\_name || chr(9) || l\_dept\_line.manager\_id || chr(9) ||

l\_dept\_line.last\_name || chr(9) || l\_dept\_line.location\_id || chr(9) || l\_dept\_line.street\_address || chr(9));

END LOOP;

END;

*-- NO.04*

*/\**

*分别编写函数与存储过程，实现*

*输入参数p\_department\_id,输出参数为cux\_ departments\_xxxx*

*记录变量，当出现找不到值时，报错：*

*输入参数p\_department\_id有误，找不到相应记录，请检查*

*\*/*

*-- 创建函数*

CREATE OR REPLACE FUNCTION cux\_get\_dept\_infor\_21080(p\_department\_id NUMBER)

RETURN departments\_21080%ROWTYPE IS

l\_departments\_rec departments\_21080%ROWTYPE;

BEGIN

SELECT \*

INTO l\_departments\_rec

FROM departments\_21080 cds

WHERE cds.department\_id = p\_department\_id;

RETURN l\_departments\_rec;

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('输入参数'||p\_department\_id||'有误，找不到相应记录，请检查');

RETURN NULL;

WHEN OTHERS THEN

dbms\_output.put\_line('erroy' || SQLERRM);

RETURN NULL;

END;

*-- 使用函数*

DECLARE

l\_departments\_rec departments\_21080%ROWTYPE;

BEGIN

l\_departments\_rec := cux\_get\_dept\_infor\_21080(p\_department\_id => 10);

dbms\_output.put\_line('Department Name:' || l\_departments\_rec.department\_name);

END;

## 最简单的运用，遍历所有数据输出

DECLARE

--l\_dept\_line departments\_21084%ROWTYPE;

CURSOR dept\_cursor IS

SELECT \*

FROM departments\_21084 d;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('DEPARTMENT\_ID' || CHR(9) || 'DEPARTMENT\_NAME' || CHR(9) || 'MANAGER\_ID' || CHR(9) || 'LOCATION\_ID');

FOR l\_dept\_line IN dept\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE(l\_dept\_line.department\_id || CHR(9) || l\_dept\_line.department\_name || CHR(9) || l\_dept\_line.manager\_id || CHR(9) || l\_dept\_line.location\_id);

END LOOP;

END;

# 4.异常处理

## 1.自定义异常：

DECLARE  
  p\_department\_id   NUMBER;  
  x\_departments\_rec departments\_21089%ROWTYPE;  
  
  CURSOR csr\_departments(p\_department\_id IN NUMBER) IS  
    SELECT \*  
      FROM departments\_21089 dts  
     WHERE dts.department\_id = p\_department\_id;  
  
  e\_noreason\_exception EXCEPTION;  
  l\_error\_message VARCHAR2(300);  
BEGIN  
 p\_department\_id := 10;  
  
  BEGIN  
    SELECT \*  
      INTO x\_departments\_rec  
      FROM departments\_21089 dts  
     WHERE dts.department\_id = p\_department\_id;  
    
    raise\_application\_error(-20000,  
                            'XXXXXXXXXXX错误');  
  EXCEPTION  
    WHEN OTHERS THEN  
      l\_error\_message := 'XXX错误';  
      dbms\_output.put\_line('1' || l\_error\_message);  
      RAISE e\_noreason\_exception;  
  END;  
  
  BEGIN  
    SELECT \*  
      INTO x\_departments\_rec  
      FROM departments\_21089 dts  
     WHERE dts.department\_id = p\_department\_id + 9999;  
  EXCEPTION  
    WHEN OTHERS THEN  
      l\_error\_message := l\_error\_message || 'XXX错误';  
      dbms\_output.put\_line('2' || l\_error\_message);  
  END;  
  
  IF l\_error\_message IS NOT NULL THEN  
    RAISE e\_noreason\_exception;  
  END IF;  
EXCEPTION  
  WHEN no\_data\_found THEN  
    dbms\_output.put\_line('error' || SQLERRM);  
  WHEN e\_noreason\_exception THEN  
    dbms\_output.put\_line('3' || l\_error\_message);  
    dbms\_output.put\_line('No Reason Error' || l\_error\_message);  
  WHEN OTHERS THEN  
    dbms\_output.put\_line('error' || SQLERRM);  
END;

# 5．函数

## 1.最简单的案例（无传值）：

CREATE OR REPLACE FUNCTION cux\_get\_dept\_name1001\_func RETURN VARCHAR2 AS  
  l\_department\_id NUMBER;  
  l\_dept\_name     VARCHAR2(30);  
  l\_manager\_id NUMBER;  
BEGIN  
  l\_department\_id := 20;  
    
  SELECT cds.department\_name,cds.manager\_id  
    INTO l\_dept\_name,l\_manager\_id  
    FROM cux\_departments\_21083 cds  
   WHERE cds.department\_id = l\_department\_id;  
  
  dbms\_output.put\_line('Dept Name:' || l\_dept\_name);  
  dbms\_output.put\_line('Manager Id:' || l\_manager\_id);  
    
  RETURN l\_dept\_name;  
EXCEPTION  
  WHEN OTHERS THEN  
    dbms\_output.put\_line('Exception:' || SQLERRM);  
END;

1. 进一点点阶：

CREATE OR REPLACE FUNCTION cux\_get\_dept\_infor\_21080(p\_department\_id NUMBER)

RETURN departments\_21080%ROWTYPE IS

l\_departments\_rec departments\_21080%ROWTYPE;

BEGIN

SELECT \*

INTO l\_departments\_rec

FROM departments\_21080 cds

WHERE cds.department\_id = p\_department\_id;

RETURN l\_departments\_rec;

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('输入参数'||p\_department\_id||'有误，找不到相应记录，请检查');

RETURN NULL;

WHEN OTHERS THEN

dbms\_output.put\_line('erroy' || SQLERRM);

RETURN NULL;

END;

*-- 使用函数*

DECLARE

l\_departments\_rec departments\_21080%ROWTYPE;

BEGIN

l\_departments\_rec := cux\_get\_dept\_infor\_21080(p\_department\_id => 10);

dbms\_output.put\_line('Department Name:' || l\_departments\_rec.department\_name);

END;

# 6.存储过程

## 1.最简单的案例：

CREATE OR REPLACE PROCEDURE cux\_get\_dept\_name1001 AS

l\_department\_id NUMBER;

l\_dept\_name VARCHAR2(30);

l\_manager\_id NUMBER;

BEGIN

l\_department\_id := 20;

SELECT cds.department\_name,cds.manager\_id

INTO l\_dept\_name,l\_manager\_id

FROM departments\_21080 cds

WHERE cds.department\_id = l\_department\_id;

dbms\_output.put\_line('Dept Name:' || l\_dept\_name);

dbms\_output.put\_line('Manager Id:' || l\_manager\_id);

EXCEPTION

WHEN OTHERS THEN

dbms\_output.put\_line('Exception:' || SQLERRM);

END;

调用：

DECLARE

BEGIN

cux\_get\_dept\_name1001;

END;

## 2进一点点阶：

*-- 创建存储过程*

CREATE OR REPLACE PROCEDURE cux\_prod\_dept\_infor\_21080(p\_department\_id IN NUMBER,

x\_departments\_rec OUT departments\_21080%ROWTYPE) IS

BEGIN

SELECT \*

INTO x\_departments\_rec

FROM departments\_21080 cds

WHERE cds.department\_id = p\_department\_id;

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('输入参数'||p\_department\_id||'有误，找不到相应记录，请检查');

WHEN OTHERS THEN

dbms\_output.put\_line('erroy' || SQLERRM);

END;

*--prcedure*

DECLARE

x\_departments\_rec departments\_21080%ROWTYPE;

BEGIN

cux\_prod\_dept\_infor\_21080( 20,x\_departments\_rec);

dbms\_output.put\_line('Department Name:' || x\_departments\_rec.department\_name);

END;

# 7.匿名块

## *1.案例*

*--1.使用匿名块，实现输入department\_id 或location\_id，通过dbms\_output输出如下部门信息*

DECLARE

l\_department\_name VARCHAR(30);

BEGIN

SELECT dts.department\_name

INTO l\_department\_name

FROM departments\_21080 dts

WHERE dts.department\_id = &l\_department\_id;

dbms\_output.put\_line('Department Name:' || l\_department\_name);

END;

*--2.使用匿名块，实现输入department\_id\_from与department\_id\_to的值，通过dbms\_output输出批量部门信息*

DECLARE

l\_department\_name VARCHAR(30);

CURSOR cur\_cursor IS

SELECT dts.department\_name

INTO l\_department\_name

FROM departments\_21080 dts

WHERE dts.department\_id BETWEEN &l\_department\_id\_from AND &l\_department\_id\_to;

BEGIN

FOR l\_dept\_line IN cur\_cursor

LOOP

dbms\_output.put\_line('Department Name:' || l\_dept\_line.department\_name);

END LOOP;

END;

# 8.触发器

## 1.创建触发器案例：

CREATE OR REPLACE TRIGGER cux\_om\_headers\_rafiud\_21080

after INSERT OR UPDATE OR DELETE ON cux\_om\_headers\_21080

--还有before

FOR EACH ROW

BEGIN

IF inserting THEN

INSERT INTO cux\_fnd\_logs\_21080

(log\_id,

action\_code,

object\_name,

object\_id,

action\_by,

action\_date)

VALUES

(cux\_fnd\_logs\_21080\_s.nextval,

'INSERT',

'CUX\_OM\_HEADERS\_21080',

:new.HEADER\_ID,

FND\_GLOBAL.USER\_ID,

SYSDATE);

ELSIF deleting THEN

INSERT INTO cux\_fnd\_logs\_21080

(log\_id,

action\_code,

object\_name,

object\_id,

action\_by,

action\_date)

VALUES

(cux\_fnd\_logs\_21080\_s.nextval,

'DELETE',

'CUX\_OM\_HEADERS\_21080',

:old.HEADER\_ID,

FND\_GLOBAL.USER\_ID,

SYSDATE);

ELSIF updating THEN

IF :new.order\_status = 'APPROVED'

OR :new.order\_status = 'CLOSED' THEN

INSERT INTO cux\_fnd\_logs\_21080

(log\_id,

action\_code,

object\_name,

object\_id,

action\_by,

action\_date,

comments)

VALUES

(cux\_fnd\_logs\_21080\_s.nextval,

'UPDATE',

'CUX\_OM\_HEADERS\_21080',

:new.HEADER\_ID,

FND\_GLOBAL.USER\_ID,

SYSDATE,

'ORDER\_STATUS='||:new.ORDER\_STATUS);

END IF;

END IF;

END;

## 2.使用OLD 和NEW修饰词

CREATE OR REPLACE TRIGGER audit\_emp\_values

AFTER DELETE OR INSERT OR UPDATE ON employees

FOR EACH ROW

BEGIN

INSERT INTO audit\_emp\_table(user\_name, timestamp, id, old\_last\_name, new\_last\_name, old\_title, new\_title, old\_salary, new\_salary)

VALUES(USER,SYSDATE, :OLD.employee\_id, :OLD.last\_name, :NEW.last\_name, :OLD.job\_id, :NEW.job\_id, :OLD.salary, :NEW.salary);

END;

# 快捷操作：

## *1.快速建对表建立增删改锁方法的工具包*

BEGIN

hand\_plsql\_autocreate.table\_handle\_pkg(p\_table\_name => 'CUX\_OM\_HEADERS',

p\_owner => 'CUX',

p\_primary\_key => 'HEADER\_ID');

END;

## 2.复制表

CREATE TABLE CUX\_PO\_QUOTES\_21080\_SYNC

AS SELECT \* FROM CUX\_PO\_QUOTES\_21080;

## 3，复制一行数据

1.特制方法复制一行数据

DECLARE

cux\_om\_headers\_rec cux\_om\_headers\_21080%ROWTYPE;

BEGIN

select \*

into cux\_om\_headers\_rec

from cux\_om\_headers\_21080 coh

where coh.header\_id=10001;

cux\_om\_headers\_rec.header\_id:=10050;

cux\_om\_headers\_rec.order\_number:=100000232;

CUX\_OM\_HEADERS\_21080\_PKG.insert\_row(cux\_om\_headers\_rec);

END;

2.快速复制一行数据

INSERT INTO cux\_om\_headers\_21080

SELECT cux\_om\_headers\_21080\_s.nextval,

org\_id,

1000003434,

customer\_id,

order\_date,

order\_status,

description,

created\_by,

creation\_date,

last\_updated\_by,

last\_update\_date,

last\_update\_login,

attribute\_category,

attribute1,

attribute2,

attribute3,

attribute4,

attribute5,

attribute6,

attribute7,

attribute8,

attribute9,

attribute10,

attribute11,

attribute12,

attribute13,

attribute14,

attribute15

FROM cux\_om\_headers\_21080 coh

WHERE coh.header\_id = 10001;

## 4.快码的使用

例：一般作为子查询

(SELECT flv.meaning

FROM fnd\_lookup\_values\_vl flv

WHERE flv.lookup\_type = 'POS\_ORDER\_STATUS'----按情况修改

AND flv.lookup\_code = pha.document\_status---按情况修改

AND flv.enabled\_flag = 'Y'

AND SYSDATE BETWEEN nvl(flv.start\_date\_active,

SYSDATE - 1) AND nvl(flv.end\_date\_active,SYSDATE)) AS header\_status,

# 问题集中营：

创建数组？：

DECLARE  
  TYPE g\_om\_headers\_rec\_type IS RECORD(  
    order\_status cux\_om\_headers\_21082.order\_status%TYPE);  
--自动拓展空间插入  
  TYPE g\_om\_headers\_tbl\_type IS TABLE OF g\_om\_headers\_rec\_type INDEX BY BINARY\_INTEGER;  
  
  g\_om\_headers\_tbl g\_om\_headers\_tbl\_type;  
  
BEGIN  
  FOR i IN 1..30  
    LOOP  
      g\_om\_headers\_tbl(i).order\_status := 'NEW';      
    END LOOP;  
  dbms\_output.put\_line(g\_om\_headers\_tbl.count);  
END;

答：自动拓展空间插入，不用extend。

# 表单情况:

*-countries\_21080 城市信息表*

*-departments\_21080 部门信息表*

*-employees\_21080 员工信息表*

*-jobs\_21080 职位信息表*

*-job\_grades\_21080 职位等级信息表*

*-job\_history\_21080 职位历史信息表*

*-locations\_21080 地址信息表*

*-CUX\_OM\_CUSTOMERS\_21080 用于存放客户信息*

*-CUX\_OM\_ITEMS\_21080 用于存放产品信息*

*-CUX\_OM\_HEADERS\_21080 用于存放销售订单头信息*

*-CUX\_OM\_LINES\_21080 用于存放销售订单行信息*

*CUX\_PO\_CONTACTS\_21080 供应商联系人*

*CUX\_PO\_VENDORS\_21080 供应商*

*CUX\_PO\_ITEMS\_21080 产品*

*CUX\_PO\_QUOTES\_21080 供应商产品报价*

*CUX\_PO\_HEADERS\_21080 采购订单头*

*CUX\_PO\_LINES\_21080 采购订单行*