

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的鞋类商品订单管理系统的数据库设计 | | |
| 课程 | Oracle数据库应用 | | |
| 学 院 | 信息科学与工程学院 | | |
| 专 业 | 软件工程 | 年级 | 2018级 |
| 学生姓名 | 王澳航 | 学号 | 201810414421 |
| 指导教师 | 赵卫东 | 职称 | 副教授 |

|  |  |  |  |
| --- | --- | --- | --- |
| **评分项** | **评分标准** | **满分** | **得分** |
| 文档整体 | 文档内容详实、规范，美观大方 | 10 |  |
| 表设计 | 表，表空间设计合理，数据合理 | 20 |  |
| 用户管理 | 权限及用户分配方案设计正确 | 20 |  |
| PL/SQL设计 | 存储过程和函数设计正确 | 30 |  |
| 备份方案 | 备份方案设计正确 | 20 |  |
| **得分合计** | | |  |

2021 年 6 月 1 日

# orcale数据库实验六-鞋类商品订单管理系统

[orcale数据库实验六-鞋类商品订单管理](#header-n0)   
 [一. 概述](#header-n4)  
 [二. 需求与功能分析](#header-n7)   
 [三. 数据库设计](#header-n9)  
 [1. 添加用户及权限管理](#header-n10)  
 [创建 WAH用户，并且给用户分配角色空间](#header-n24)  
 [2. 通过新创建的用户连接到 myorcl](#header-n28)   
 [4. 利用新创建的用户WAH创建了五个表](#header-n34)  
 [6. 创建视图](#header-n138)  
 [7. 将五个表的视图的SELECT对象权限授予WAH用户](#header-n151)   
 [8. 向数据库中写入数据共计50000多个数据](#header-n153)  
 [8.1 部门表](#header-n154)  
 [8.2 员工表](#header-n157)  
 [8.3 订单详情表](#header-n160)  
 [8.4 订单表](#header-n163)  
 [8.5 产品表](#header-n166)  
 [9. PL/SQL设计](#header-n169)  
 [9.2. 包](#header-n197)  
 [9.2.1. 查询部门销售额](#header-n198)  
 [10. 数据库备份](#header-n202)  
 [热备份和冷备份优缺点](#header-n205)  
 [热备份的优点是：](#header-n206)  
 [热备份的不足是：](#header-n212)  
 [冷备份的优点是：](#header-n216)  
 [冷备份不足是：](#header-n222)  
 [物理备份之冷备份：](#header-n227)  
 [10.1. 全数据库备份](#header-n236)  
 [10.2. 查询备份文件](#header-n241)  
 [10.3. 查看备份文件的内容](#header-n245)  
 [10.4. 备份后修改数据库](#header-n255)  
 [10.5. 删除数据库文件模拟数据损坏](#header-n259)  
 [10.6. 数据库完全恢复](#header-n268)  
 [10.6.1. 重启损坏的数据库到mount状态](#header-n269)  
 [10.6.2. 开始恢复数据库](#header-n274)  
 [10.6.3. 查询数据是否恢复](#header-n278)  
 [11. DataGuard实现数据库整体的异地备份](#header-n284)

## 一. 概述

此项目的主要目的是通过对各种品牌的鞋的市场调查，再通过整理其价格，市场目标，市场销量对其进行分类。通过该项目能对市场上的大量球鞋进行初步了解，对于普通消费者以及球鞋市场有着重要研究价值。鞋类商品订单管理服务于管理和发展两方面,目的是提高企业销售能力、各部门的工作效率、员工的职业技能,整个体系的有效性还对公司管理以及维护客户关系、协调并控制销售人员十分具有意义。

## 二. 需求与功能分析

鞋类商品订单管理系统是在信息时代化背景下发展的产物,依赖于计算机以及具体操作软件的运行环境。落后的书面时代已经过去，在本子上记下一笔又一笔的订单销售记录的方式已经被公司摒弃,这种人为的劳动力使用是应该被机械化数据取代。鞋类商品订单管理系统的主要功能是通过统一订单提供用户整合的一站式的供应链服务,订单管理以及订单跟踪管理能够使用户的物流服务得到全程的满足。订单管理系统是物流管理链条中的不可或缺的部分，通过对订单的管理和分配,使仓储管理和运输管理有机的结合,稳定有效地实现物流管理中各个环节充分发挥作用﹐使仓储、运输、订单成为一个有机整体，满足物流系统信息化的需求。

## 三. 数据库设计

## 1. 添加用户及权限管理

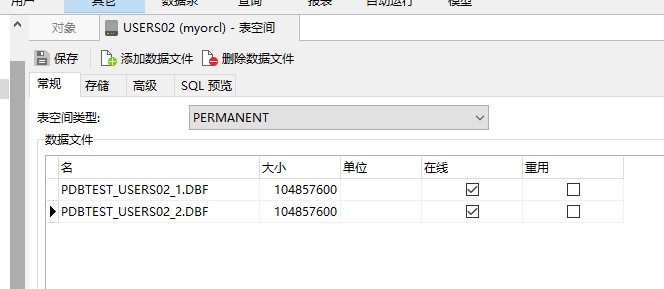
oracle中的表就是一张存储数据的表。表空间是逻辑上的划分。方便管理的。

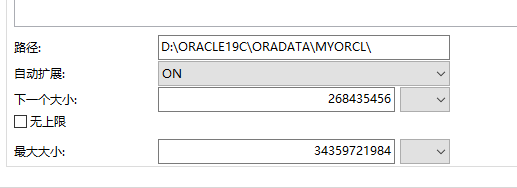
数据表空间，即存放数据总是需要空间， Oracle把一个数据库按功能划分若干空间来保存数据。当然数据存放在磁盘最终是以文件形式，所以一盘一个数据表空间包含一个以上的物理文件  
数据表。

在仓库，我们可能有多间房子，每个房子又有多个货架，每架又有多层。 我们在数据库中存放数据，最终是数据表的单元来存储与管理的。  
数据文件。

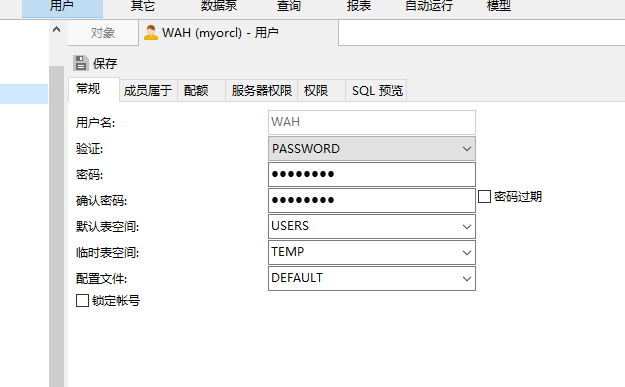
以上几个概念都是逻辑上的， 而数据文件则是物理上的。就是说，数据文件是真正“看得着的东西”，它在磁盘上以一个真实的文件体现。

创建表空间 USERS02：

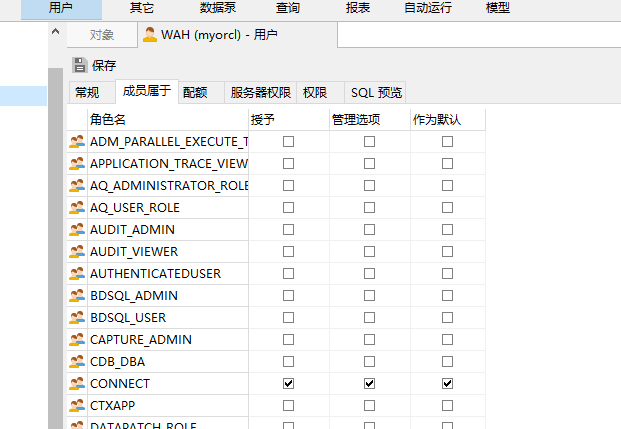




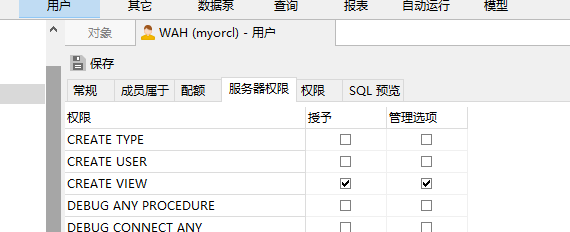
建好tablespace, 就可以建用户



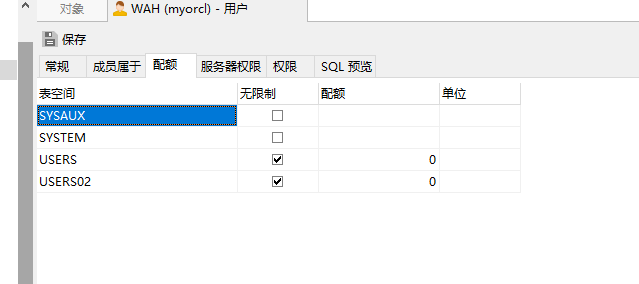
成员属于选择Connect和Recourse：



授权 create view：



配额：



### 使用WAH用户登录本地Oracle数据库



## 4. 利用新创建的用户WAH创建了五个表

DECLARE num NUMBER;

BEGIN

SELECT

count( 1 ) INTO num

FROM

user\_tables

WHERE

TABLE\_NAME = 'DEPARTMENTS';

IF

num = 1 THEN

execute IMMEDIATE 'drop table DEPARTMENTS cascade constraints PURGE';

END IF;

SELECT

count( 1 ) INTO num

FROM

user\_tables

WHERE

TABLE\_NAME = 'EMPLOYEES';

IF

num = 1 THEN

execute IMMEDIATE 'drop table EMPLOYEES cascade constraints PURGE';

END IF;

SELECT

count( 1 ) INTO num

FROM

user\_tables

WHERE

TABLE\_NAME = 'ORDER\_ID\_TEMP';

IF

num = 1 THEN

execute IMMEDIATE 'drop table ORDER\_ID\_TEMP cascade constraints PURGE';

END IF;

SELECT

count( 1 ) INTO num

FROM

user\_tables

WHERE

TABLE\_NAME = 'ORDER\_DETAILS';

IF

num = 1 THEN

execute IMMEDIATE 'drop table ORDER\_DETAILS cascade constraints PURGE';

END IF;

SELECT

count( 1 ) INTO num

FROM

user\_tables

WHERE

TABLE\_NAME = 'ORDERS';

IF

num = 1 THEN

execute IMMEDIATE 'drop table ORDERS cascade constraints PURGE';

END IF;

SELECT

count( 1 ) INTO num

FROM

user\_tables

WHERE

TABLE\_NAME = 'PRODUCTS';

IF

num = 1 THEN

execute IMMEDIATE 'drop table PRODUCTS cascade constraints PURGE';

END IF;

SELECT

count( 1 ) INTO num

FROM

user\_sequences

WHERE

SEQUENCE\_NAME = 'SEQ\_ORDER\_DETAILS\_ID';

IF

num = 1 THEN

execute IMMEDIATE 'drop SEQUENCE SEQ\_ORDER\_DETAILS\_ID';

END IF;

SELECT

count( 1 ) INTO num

FROM

user\_sequences

WHERE

SEQUENCE\_NAME = 'SEQ\_ORDER\_ID';

IF

num = 1 THEN

execute IMMEDIATE 'drop SEQUENCE SEQ\_ORDER\_ID';

END IF;

SELECT

count( 1 ) INTO num

FROM

user\_views

WHERE

VIEW\_NAME = 'VIEW\_ORDER\_DETAILS';

IF

num = 1 THEN

execute IMMEDIATE 'drop VIEW VIEW\_ORDER\_DETAILS';

END IF;

SELECT

count( object\_name ) INTO num

FROM

user\_objects\_ae

WHERE

object\_type = 'PACKAGE'

AND OBJECT\_NAME = 'MYPACK';

IF

num = 1 THEN

execute IMMEDIATE 'DROP PACKAGE MYPACK';

END IF;

END;

/

CREATE TABLE DEPARTMENTS (

DEPARTMENT\_ID NUMBER ( 6, 0 ) NOT NULL,

DEPARTMENT\_NAME VARCHAR2 ( 40 BYTE ) NOT NULL,

CONSTRAINT DEPARTMENTS\_PK PRIMARY KEY ( DEPARTMENT\_ID )

);

CREATE TABLE EMPLOYEES (

EMPLOYEE\_ID NUMBER ( 6, 0 ) NOT NULL,

NAME VARCHAR2 ( 40 BYTE ) NOT NULL,

EMAIL VARCHAR2 ( 40 BYTE ),

PHONE\_NUMBER VARCHAR2 ( 40 BYTE ),

HIRE\_DATE DATE NOT NULL,

SALARY NUMBER ( 8, 2 ),

MANAGER\_ID NUMBER ( 6, 0 ),

DEPARTMENT\_ID NUMBER ( 6, 0 ),

PHOTO BLOB,

CONSTRAINT EMPLOYEES\_PK PRIMARY KEY ( EMPLOYEE\_ID )

);

CREATE TABLE PRODUCTS (

PRODUCT\_NAME VARCHAR2 ( 40 BYTE ) NOT NULL,

PRODUCT\_TYPE VARCHAR2 ( 40 BYTE ) NOT NULL,

CONSTRAINT PRODUCTS\_PK PRIMARY KEY ( PRODUCT\_NAME )

);

CREATE GLOBAL TEMPORARY TABLE "ORDER\_ID\_TEMP" ( "ORDER\_ID" NUMBER ( 10, 0 ) NOT NULL ENABLE, CONSTRAINT "ORDER\_ID\_TEMP\_PK" PRIMARY KEY ( "ORDER\_ID" ) ENABLE ) ON COMMIT DELETE ROWS;

COMMENT ON TABLE "ORDER\_ID\_TEMP" IS '用于触发器存储临时ORDER\_ID';

CREATE TABLE ORDERS (

ORDER\_ID NUMBER ( 10, 0 ) NOT NULL,

ORDER\_DATE DATE NOT NULL,

DEPARMENT\_ID NUMBER ( 6, 0 ) NOT NULL,

TRADE\_RECEIVABLE NUMBER ( 8, 2 ) DEFAULT 0

) TABLESPACE USERS PCTFREE 10 INITRANS 1 STORAGE ( BUFFER\_POOL DEFAULT ) NOCOMPRESS NOPARALLEL PARTITION BY RANGE ( ORDER\_DATE ) (

PARTITION PARTITION\_BEFORE\_2016

VALUES

LESS THAN (

TO\_DATE( ' 2016-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN' )) NOLOGGING TABLESPACE USERS PCTFREE 10 INITRANS 1 STORAGE ( INITIAL 8388608 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED BUFFER\_POOL DEFAULT ) NOCOMPRESS NO INMEMORY,

PARTITION PARTITION\_BEFORE\_2017

VALUES

LESS THAN (

TO\_DATE( ' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN' )) NOLOGGING TABLESPACE USERS02 PCTFREE 10 INITRANS 1 STORAGE ( INITIAL 8388608 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED BUFFER\_POOL DEFAULT ) NOCOMPRESS NO INMEMORY

);

CREATE INDEX ORDERS\_INDEX\_DATE ON ORDERS ( ORDER\_DATE ASC ) LOCAL (

PARTITION PARTITION\_BEFORE\_2016 TABLESPACE USERS PCTFREE 10 INITRANS 2 STORAGE ( INITIAL 8388608 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED BUFFER\_POOL DEFAULT ) NOCOMPRESS,

PARTITION PARTITION\_BEFORE\_2017 TABLESPACE USERS02 PCTFREE 10 INITRANS 2 STORAGE ( INITIAL 8388608 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED BUFFER\_POOL DEFAULT ) NOCOMPRESS

) STORAGE ( BUFFER\_POOL DEFAULT ) NOPARALLEL;

CREATE UNIQUE INDEX ORDERS\_PK ON ORDERS ( ORDER\_ID ASC ) GLOBAL PARTITION BY HASH ( ORDER\_ID ) ( PARTITION INDEX\_PARTITION1 TABLESPACE USERS NOCOMPRESS, PARTITION INDEX\_PARTITION2 TABLESPACE USERS02 NOCOMPRESS ) NOLOGGING TABLESPACE USERS PCTFREE 10 INITRANS 2 STORAGE ( INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED BUFFER\_POOL DEFAULT ) NOPARALLEL;

ALTER TABLE ORDERS ADD CONSTRAINT ORDERS\_PK PRIMARY KEY ( ORDER\_ID ) USING INDEX ORDERS\_PK ENABLE;

CREATE TABLE ORDER\_DETAILS (

ID NUMBER ( 10, 0 ) NOT NULL,

ORDER\_ID NUMBER ( 10, 0 ) NOT NULL,

PRODUCT\_NAME VARCHAR2 ( 40 BYTE ) NOT NULL,

PRODUCT\_NUM NUMBER ( 8, 2 ) NOT NULL,

PRODUCT\_PRICE NUMBER ( 8, 2 ) NOT NULL,

CONSTRAINT ORDER\_DETAILS\_FK1 FOREIGN KEY ( ORDER\_ID ) REFERENCES ORDERS ( ORDER\_ID ) ENABLE

) TABLESPACE USERS PCTFREE 10 INITRANS 1 STORAGE ( BUFFER\_POOL DEFAULT ) NOCOMPRESS NOPARALLEL PARTITION BY REFERENCE ( ORDER\_DETAILS\_FK1 ) (

PARTITION PARTITION\_BEFORE\_2016 NOLOGGING TABLESPACE USERS --必须指定表空间，否则会将分区存储在用户的默认表空间中

PCTFREE 10 INITRANS 1 STORAGE ( INITIAL 8388608 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED BUFFER\_POOL DEFAULT ) NOCOMPRESS NO INMEMORY,

PARTITION PARTITION\_BEFORE\_2017 NOLOGGING TABLESPACE USERS02 PCTFREE 10 INITRANS 1 STORAGE ( INITIAL 8388608 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED BUFFER\_POOL DEFAULT ) NOCOMPRESS NO INMEMORY

);

CREATE UNIQUE INDEX ORDER\_DETAILS\_PK ON ORDER\_DETAILS ( ID ASC ) NOLOGGING TABLESPACE USERS PCTFREE 10 INITRANS 2 STORAGE ( INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS UNLIMITED BUFFER\_POOL DEFAULT ) NOPARALLEL;

ALTER TABLE ORDER\_DETAILS ADD CONSTRAINT ORDER\_DETAILS\_PK PRIMARY KEY ( ID ) USING INDEX ORDER\_DETAILS\_PK ENABLE;

CREATE SEQUENCE SEQ\_ORDER\_ID INCREMENT BY 1 START WITH 1 MAXVALUE 999999999 MINVALUE 1 CACHE 20 ORDER;

CREATE SEQUENCE SEQ\_ORDER\_DETAILS\_ID INCREMENT BY 1 START WITH 1 MAXVALUE 999999999 MINVALUE 1 CACHE 20 ORDER;

部门信息表

|  |  |  |
| --- | --- | --- |
| 属性 | 字段 | 注解 |
| 部门ID | DEPARTMENT\_ID | 部门ID |
| 部门名称 | DEPARTMENT\_NAME | 部门名称 |

产品表

|  |  |  |
| --- | --- | --- |
| 属性 | 字段 | 注解 |
| 产品编号 | PRODUCT\_ID | 产品编号 |
| 产品名称 | PRODUCT\_NAME | 产品名称 |
| 产品类型 | PRODUCT\_TYPE | 产品类型 |

员工表

|  |  |  |
| --- | --- | --- |
| 属性 | 字段 | 注解 |
| 编号 | EMPLOYEE-ID | 员工ID |
| 姓名 | NAME | 姓名 |
| 电子邮件 | EMAIL | 电子邮件 |
| 薪资 | SALARY | 薪资 |
| 照片 | PHOTO | 照片 |

订单表

|  |  |  |
| --- | --- | --- |
| 属性 | 字段 | 注解 |
| 订单编号 | ORDER\_ID | 订单编号 |
| 日期  部门ID | ORDER\_DATE  DEPARTMENT\_ID | 下单日期  部门id |

订单详细表

|  |  |  |
| --- | --- | --- |
| 属性 | 字段 | 注解 |
| 编号 | ID | 编号 |
| 编号 | ORDER\_ID | 订单编号 |
| 产品名称  产品数量  产品价格 | PRODUCT\_NAME  PRODUCT\_NUM  PRODUCT\_PRICE | 信息介绍  产品数量  产品价格 |

## 6. 创建视图

* 视图(view)，也称虚表, 不占用物理空间，这个也是相对概念，因为视图本身的定义语句还是要存储在数据字典里的。视图只有逻辑定义。每次使用的时候,只是重新执行SQL。
* 视图是从一个或多个实际表中获得的，这些表的数据存放在数据库中。那些用于产生视图的表叫做该视图的基表。一个视图也可以从另一个视图中产生。
* 视图的定义存在数据库中，与此定义相关的数据并没有再存一份于数据库中。通过视图看到的数据存放在基表中。
* 视图看上去非常象数据库的物理表，对它的操作同任何其它的表一样。当通过视图修改数据时，实际上是在改变基表中的数据；相反地，基表数据的改变也会自动反映在由基表产生的视图中。由于逻辑上的原因，有些Oracle视图可以修改对应的基表，有些则不能（仅仅能查询）。
* 还有一种视图：物化视图（MATERIALIZED VIEW ），也称实体化视图，快照 （8i 以前的说法） ，它是含有数据的，占用存储空间。

## **CREATE OR REPLACE FORCE EDITIONABLE VIEW "VIEW\_ORDER\_DETAILS" ("ID", "ORDER\_ID", "ORDER\_DATE", "PRODUCT\_TYPE", "PRODUCT\_NAME", "PRODUCT\_NUM", "PRODUCT\_PRICE") AS**

## **SELECT**

## **d.ID,**

## **o.ORDER\_ID,**

## **o.ORDER\_DATE,**

## **p.PRODUCT\_TYPE,**

## **d.PRODUCT\_NAME,**

## **d.PRODUCT\_NUM,**

## **d.PRODUCT\_PRICE**

## **FROM ORDERS o,ORDER\_DETAILS d,PRODUCTS p where d.ORDER\_ID=o.ORDER\_ID and d.PRODUCT\_NAME=p.PRODUCT\_NAME;**

## 8. 向数据库中写入数据共计50000多个数据

declare

dt date;

m number(8,2);

V\_department\_ID NUMBER(6);

v\_order\_id number(10);

v\_name VARCHAR2 ( 100 );

v number(10,2);

begin

for i in 1..10000

loop

if i mod 2 =0 then

dt:=to\_date('2015-3-2','yyyy-mm-dd')+(i mod 60);

else

dt:=to\_date('2016-3-2','yyyy-mm-dd')+(i mod 60);

end if;

V\_department\_ID:=CASE I MOD 3 + 1 WHEN 1 THEN 1 WHEN 2 THEN 2 ELSE 3 END;

--插入订单

v\_order\_id:=SEQ\_ORDER\_ID.nextval; --应该将SEQ\_ORDER\_ID.nextval保存到变量中。

insert /\*+append\*/ into ORDERS (ORDER\_ID,ORDER\_DATE,DEPARMENT\_ID)

values (v\_order\_id,dt,V\_department\_ID);

--插入订单y一个订单包括3个产品

v:=dbms\_random.value(100,40);

v\_name:='巴黎世家'|| (i mod 3 + 1);

insert /\*+append\*/ into ORDER\_DETAILS(ID,ORDER\_ID,PRODUCT\_NAME,PRODUCT\_NUM,PRODUCT\_PRICE)

values (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,1,v);

v:=dbms\_random.value(20,10);

v\_name:='莆田鞋'|| (i mod 3 + 1);

insert /\*+append\*/ into ORDER\_DETAILS(ID,ORDER\_ID,PRODUCT\_NAME,PRODUCT\_NUM,PRODUCT\_PRICE)

values (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,1,v);

v:=dbms\_random.value(30,15);

v\_name:='耐克'|| (i mod 3 + 1);

insert /\*+append\*/ into ORDER\_DETAILS(ID,ORDER\_ID,PRODUCT\_NAME,PRODUCT\_NUM,PRODUCT\_PRICE)

values (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,1,v);

--在触发器关闭的情况下，需要手工计算每个订单的应收金额：

select sum(PRODUCT\_NUM\*PRODUCT\_PRICE) into m from ORDER\_DETAILS where ORDER\_ID=v\_order\_id;

if m is null then

m:=0;

end if;

UPDATE ORDERS SET TRADE\_RECEIVABLE = m WHERE ORDER\_ID=v\_order\_id;

IF I MOD 1000 =0 THEN

commit; --每次提交会加快插入数据的速度

END IF;

end loop;

--统计用户的所有表，所需时间很长：2千万行数据，需要1600秒，该语句可选

--dbms\_stats.gather\_schema\_stats(User,estimate\_percent=>100,cascade=> TRUE); --estimate\_percent采样行的百分比

end;

/

--最后动态增加一个PARTITION\_BEFORE\_2018分区：

ALTER TABLE ORDERS

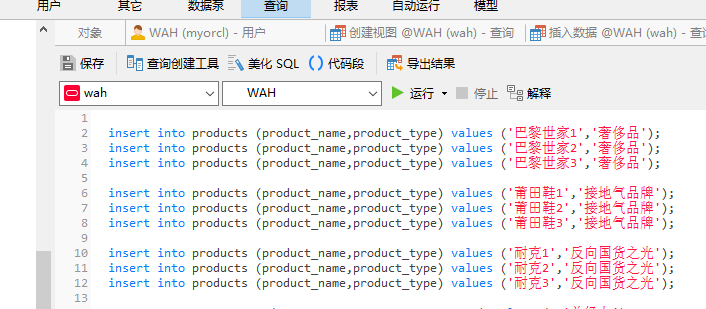
ADD PARTITION PARTITION\_BEFORE\_2018 VALUES LESS THAN (TO\_DATE(' 2018-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'));

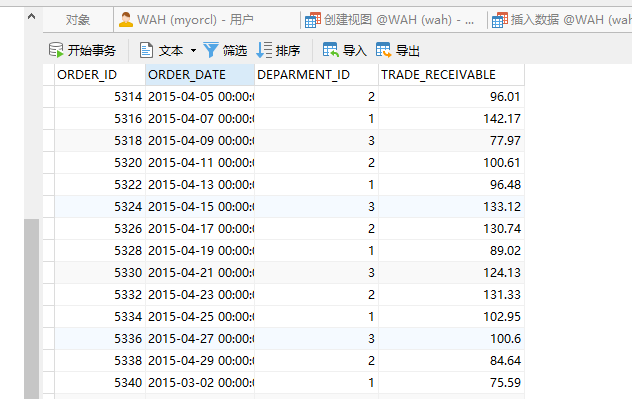
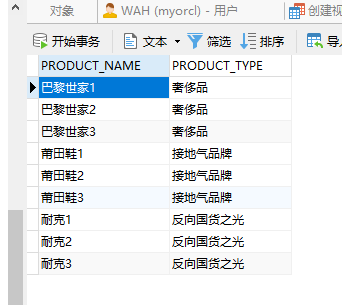
ALTER INDEX ORDERS\_INDEX\_DATE

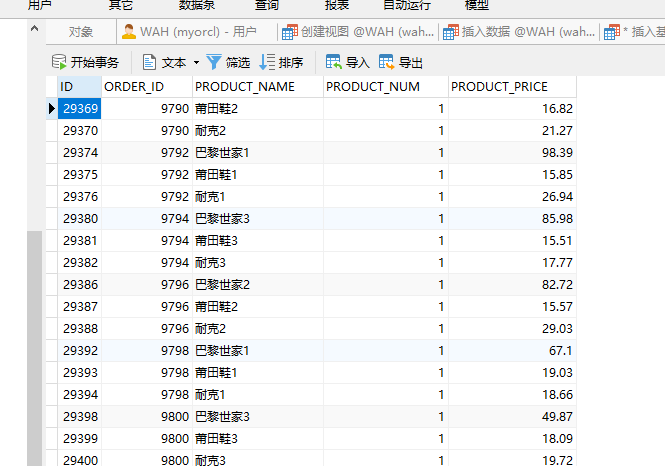
MODIFY PARTITION PARTITION\_BEFORE\_2018

NOCOMPRESS;/

代码截图：







## 9. PL/SQL设计

过程和函数由以下4部分：

* 签名或头
* 关键字IS或AS
* 局部声明（可选）
* BEGIN和END之间的过程体（包括异常处理程序）

简单示例：

create or replace procedure show\_line(ip\_line\_length in number, ip\_separator in varchar2)  
is   
actual\_line varchar2(150);  
begin  
 insert into t\_user(id,name,sex)values(ip\_line\_length,ip\_separator,ip\_line\_length);  
 for idx in 1..ip\_line\_length loop  
 actual\_line := actual\_line||ip\_separator;  
 end loop;  
 dbms\_output.put\_line(actual\_line);  
exception when others then  
 dbms\_output.put\_line(SQLERRM);  
end;

如下调用：

begin show\_line(50,'=');end;/

在SQLPLUS里面调用：

SQL> BEGIN2 show\_line(50,'=');3 END;

几点说明：

1、参数没有指定长度，当有实际数据传递进来的时候，参数的长度才被确定。

2、局部声明为：actual\_line varchar2(150);

3、使用命令SQL> show errors在SQLPLUS里面查看错误。

### 9.2. 创建函数

#### 9.2.1. 查询部门所属总销售额

CREATE OR REPLACE

PACKAGE BODY "MYPACK" AS FUNCTION Get\_SaleAmount ( DEPARTMENT\_ID NUMBER ) RETURN NUMBER AS N NUMBER ( 20, 2 );

BEGIN

SELECT SUM(O.TRADE\_RECEIVABLE) into N FROM ORDERS O,DEPARTMENTS D

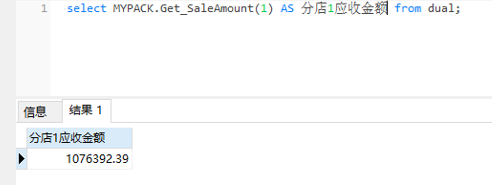
WHERE O.DEPARTMENT\_ID=D.DEPARTMENT\_ID AND D.DEPARTMENT\_ID ="门店号";

RETURN N;

END;

END;

#### 9.2.2. 测试



## 10. 数据库备份

**ORACLE数据库备份与恢复详解**

Oracle的备份与恢复有三种标准的模式，大致分为两 大类，备份恢复(物理上的)以及导入导出(逻辑上的)，而备份恢复又可以根据数据库的工作模式分为非归档模式(Nonarchivelog-style) 和归档模式(Archivelog-style),通常，我们把非归档模式称为冷备份，而相应的把归档模式称为热备份。

### 热备份和冷备份优缺点

#### 热备份的优点是：

1．可在表空间或数据文件级备份，备份时间短。

2．备份时数据库仍可使用。

3．可达到秒级恢复（恢复到某一时间点上）。

4．可对几乎所有数据库实体作恢复。

5．恢复是快速的，在大多数情况下在数据库仍工作时恢复。

#### 热备份的不足是：

1．不能出错，否则后果严重。

2．若热备份不成功，所得结果不可用于时间点的恢复。

3．因难维护，所以要特别仔细小心，不允许“以失败而告终”。

#### 冷备份的优点是：

1．是非常快速的备份方法（只需拷贝文件）

2．容易归档（简单拷贝即可）

3．容易恢复到某个时间点上（只需将文件再拷贝回去）

4．能与归档方法相结合，作数据库“最新状态”的恢复。

5．低度维护，高度安全。

#### 冷备份不足是：

1．单独使用时，只能提供到“某一时间点上”的恢复。

2．在实施备份的全过程中，数据库必须要作备份而不能作其它工作。也就是说，数据库必须是关闭状态。

3．若磁盘空间有限，只能拷贝到磁带等其它外部存储设备上，速度会很慢。

4．不能按表或按用户恢复。

### 物理备份之冷备份：

当数据库可以暂时处于关闭状态时，我们需要将它在这一稳定时刻的数据相关文件转移到安全的区域，当数据库遭到破坏，再从安全区域将备份的数据库相关文件拷 贝回原来的位置，这样，就完成了一次快捷安全等数据转移。由于是在数据库不提供服务的关闭状态，所以称为冷备份。冷备份具有很多优良特性，比如上面图中我 们提到的，快速，方便，以及高效。一次完整的冷备份步骤应该是：

1，首先关闭数据库（shutdown normal）

2，拷贝相关文件到安全区域（利用操作系统命令拷贝数据库的所有的数据文件、日志文件、控制文件、参数文件、口令文件等（包括路径））

3，重新启动数据库（startup）

这样，我们就完成了一次冷备份，请确定你对这些相应的目录（包括写入的目标文件夹）有相应的权限。

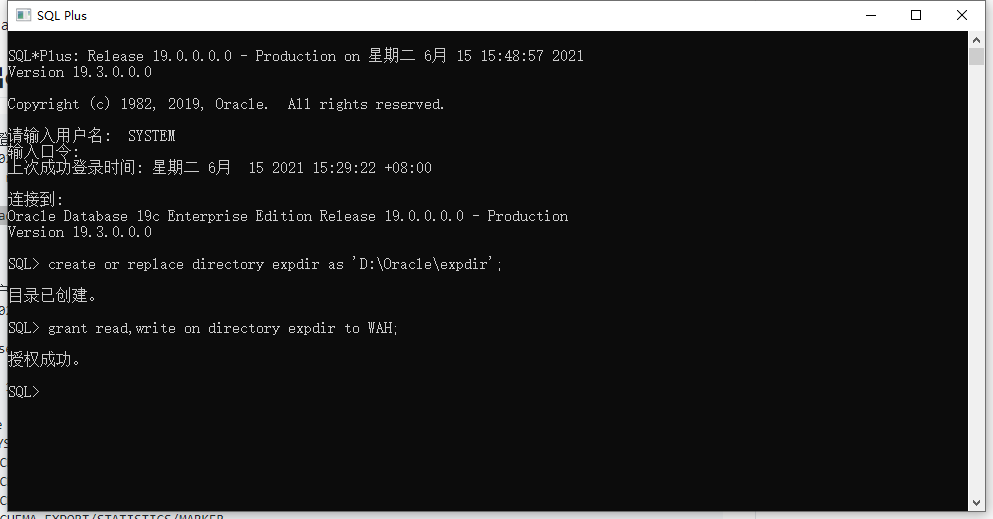
恢复的时候，相对比较简单了，我们停掉数据库，将文件拷贝回相应位置，重启数据库就可以了，当然也可以用脚本来完成。

## 10.1. 手动数据库备份之数据库导出expdp

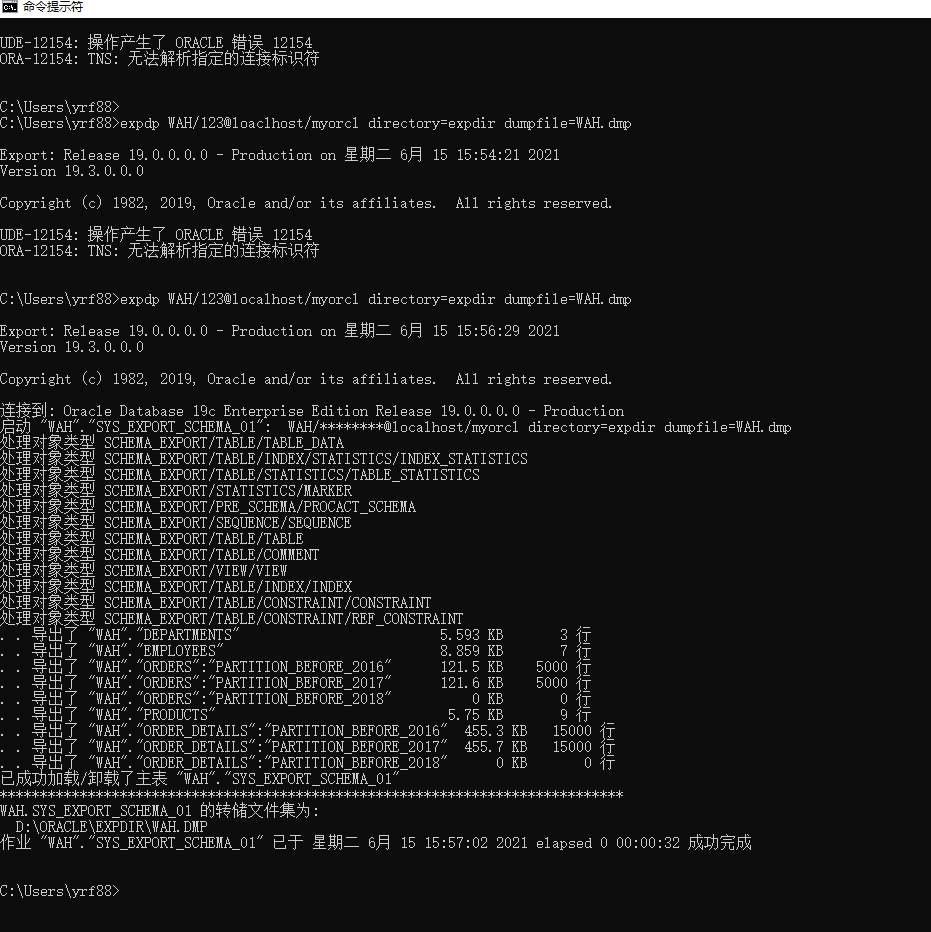
### 步骤：

SQL plus 已SYSTEM 用户登录

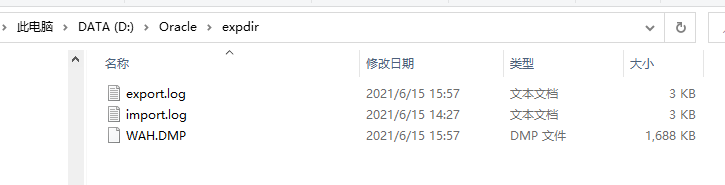
创建expdir目录并给WAH用户授权读写



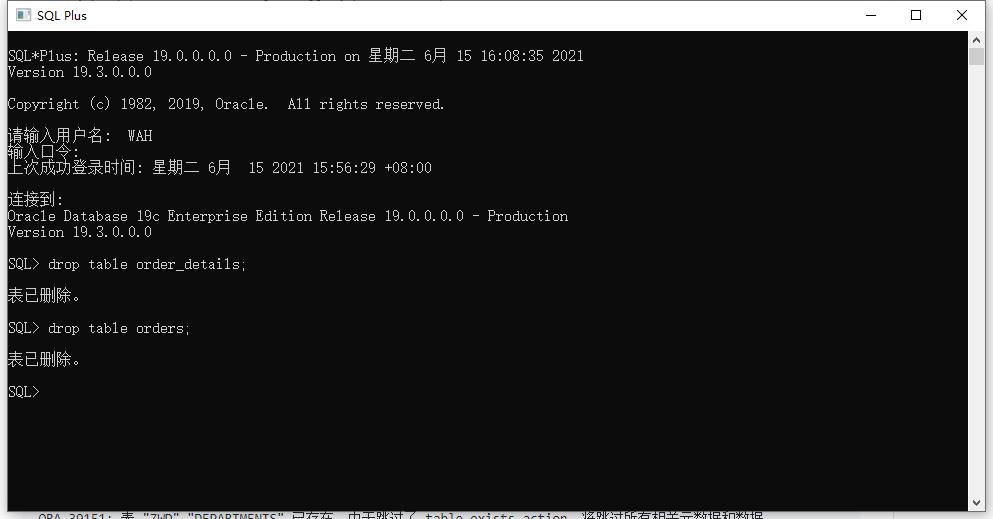
用户自己备份



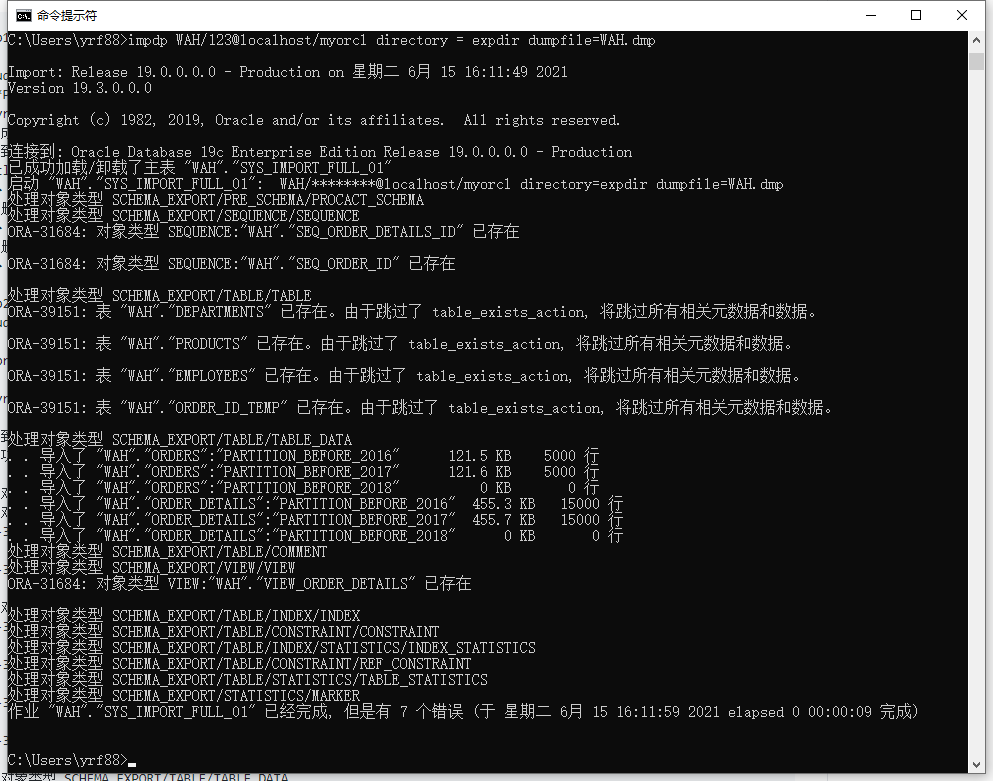
前往本地目录查看：



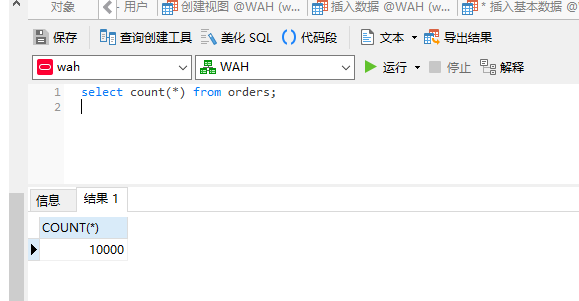
### 10.5. 删除数据库文件模拟数据损坏



### 10.6. 数据库完全恢复



用户WAH登录数据库查看恢复情况;



恢复成功。