

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的商城商品销售系统的数据库设计 | | |
| 课程 | Oracle数据库应用 | | |
| 学 院 | 计算机学院 | | |
| 专 业 | 软件工程 | 年级 | 2018级 |
| 学生姓名 | 王鹏凯 | 学号 | 201810424327 |
| 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

目录

1. 背景
   1. 开发现状
   2. 研究目的
2. 开发环境
3. 系统设计

3.1系统结构

3.2系数据表设计

1. 详细设计及代码实现

4.1用户创建以及空间分配

4.2创建表、约束。所以

4.3创建触发器、序列、视图

4.4插入数据和数据库检测

1. 手工全备份和全恢复

5.1开始全备份

5.2开始全恢复

1. 总结

### 背景

* 1. 开发现状

随着如今社会市场竞争的越发激烈，为了提高企业的市场竞争力，减少不必要的各项支出，提高企业管理水平、采用先进高效的管理方法成为了企业发展的必要条件。企业经营产生的购入、支出数据量越发巨大，传统的管理经营方法已经无法满足日益增长的市场需要。因此很多具有较好的网络硬件建设基础的企业开始着手研发电子商品销售系统来逐步取代传统的管理方式。企业通过商品销售系统，可以保证信息处理的及时性与准确性，还可以对商品的购入与销出进行高效的管理。本项目将结合大多中小型企业的现状,以及现有的销售系统情况进行研发。

* 1. 研究目的

商品的集成销售管理是一家企业进行经营管理的核心内容，在企业管理中占据首要地位。一个企业的经济实力如何，很大程度取决于企业的销售管理效果。利用先进的计算机技术，对企业的销售及销售过程中所涉及的--系列账务关系进行全面细致的跟踪管理，很大程度上解决了人工管理过程中的管理效率低下、数据准确性不高、管理不及时等问题，从而大大提高了企业经营运转的速度，提升企业的管理水平、提高企业的经济效益，使企业的销售管理更加的科学、合理。管理方式的转变，使企业能够及时通过计算机管理系统，对整个销售过程进行数据的统计与分析，并根据分析结果做出必要的调整，使企业在经济浪潮中占据有利地位，更好的适应市场的发展变化。

### 开发环境

Oracle 12c

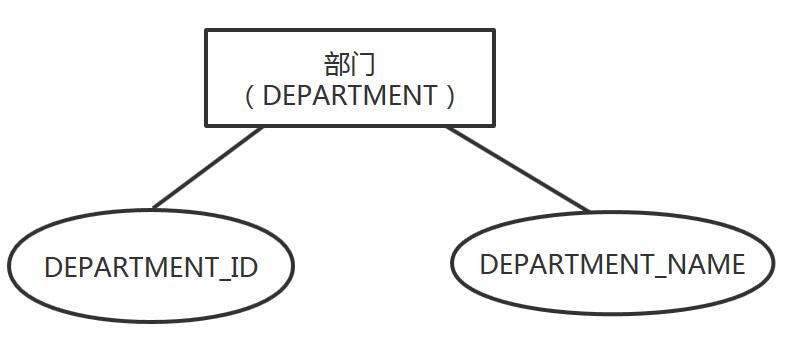
Oracle Database，又名Oracle RDBMS，或简称Oracle。是[甲骨文公司](https://baike.baidu.com/item/%E7%94%B2%E9%AA%A8%E6%96%87%E5%85%AC%E5%8F%B8/430115)的一款[关系数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/11032386)。它是在数据库领域一直处于领先地位的产品。可以说Oracle数据库系统是目前世界上流行的关系数据库管理系统，系统可移植性好、使用方便、功能强，适用于各类大、中、小、微机环境。它是一种高效率、可靠性好的、适应高吞吐量的数据库方案。

### 3.系统设计

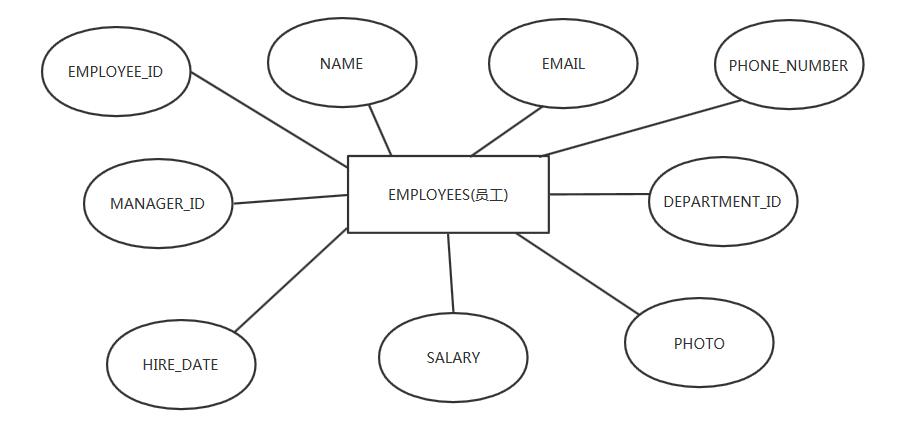
3.1.1实体模型

根据应用场最分析，共有3个原始的实体(ntit,它们是部门、员工和产品。

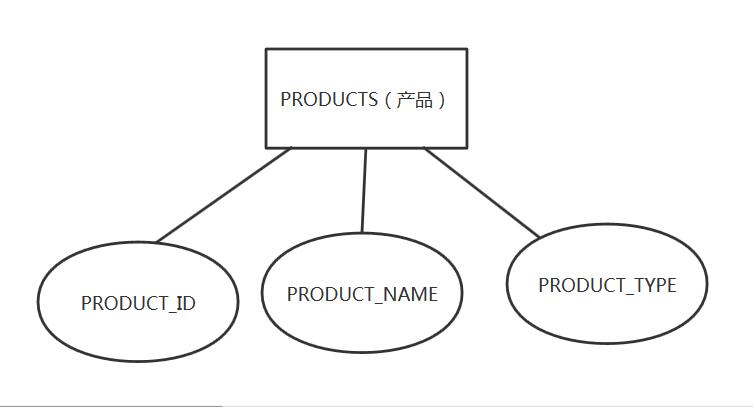
a.部门（DEPARTMENT）：部门包括部门ID(DEPARTMENT JID) 和部门名称(DEPARTMENT NAME)。部门的实体如图2



b.员工（EMPLOYEES）：员工包括员工ID(EMPLOYEE ID)，姓名(NAME)，照片(PHOTO)，工资(SALARY)等。员工的属性中还应包括员工所属的部门ID(DEPARTMENT ID),部门ID不能为空，表示员工必须属于某一个部门。 员工的属性中还有员工的上司(MANAGER ID),该属性可以为空，表示没有上司。员工的实体如图2。

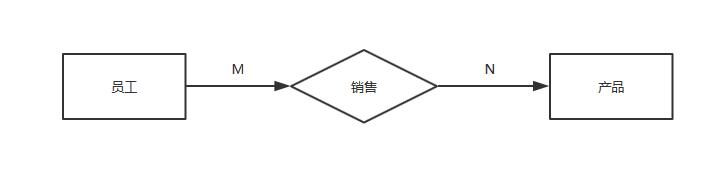


c.产品（PRODUCTS）：产品包括3个属性:产品ID（PRODUCT\_ID）产品名称(PRODUCT NAME)和产品类型(PRODUCT TYPE).见图

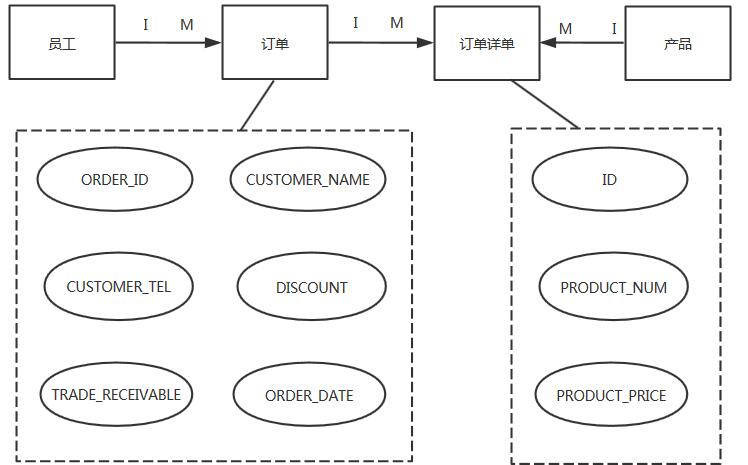


3.1.2 实体联系模型

企业员工的工作是销售产品，因此员工和产品之间就有一个“销售”的联系，如图所示，员工与产品之间的关系是多对多的关系



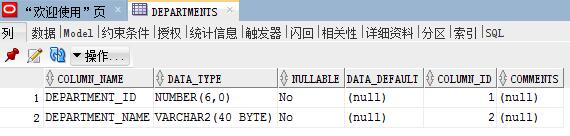
考虑到销售活动中有一些重要属性， 比如折扣，客户信息，销售时间，产品的销售数量和销售价格，我们把销售关系细分为订单和订单详单两个实体，订单中存储:订单ID(ORDER\_ ID)、折扣(DISCOUNT)、客户信息(CUSTOMER NAME, CUSTOMER TEL)、订单时间(ORDER DATE)以及应收货款总额(TRADE RECEIVABLE)，订单详单中存储订单详单的ID,以及订单中的全部产品信息，包括:销售数量(PRODUCT NUM)和销售价格(PRODUCT\_ PRICE)



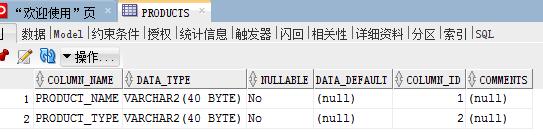
3.2数据库表设计

 E-R模型建立好以后，就可以设计Oracle的关系表了。在独立实体中找出主要属性设置为主键，比如在产品表中，产品名称(PRODUCT\_ NAME)是主键。由关系派生出的实体中要加入外键关系，比如在图14-5 中有两个一对多的关系，需要增加外键属性，即在订单表中增加员工ID属性(EMPLOYEE ID)，在订单详单中增加产品ID属性(PRODUCT\_ ID)。

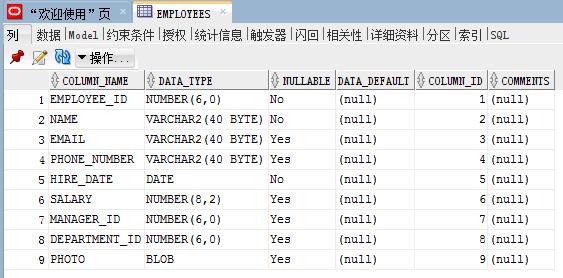
部门表（DEPARTMENT）包括DEPARTMENT\_ID和DEPARTMENT\_NAME两个属性。

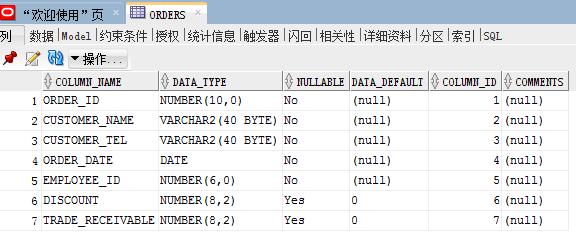


产品表PRODUCTS包括产品ID(PRODUCT\_ ID)、产品名称(PRODUCT\_ NAME)和产品类型(PRODUCT TYPE)，见表14-2。注意产品表中的产品类型只能取值:耗材，手机，电脑。

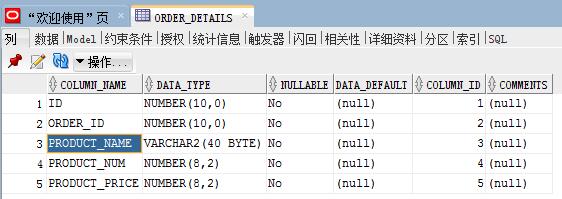


员工表EMPLOYEES包括员工的属性，要注意MANAGER ID 是员工的上司，是员工表EMPOLYEE ID的外键，MANAGER ID不能等于EMPLOYEE\_ID，即员工的领导不能是自己。主键删除时MANAGER ID设置为空值。见表14-3。

  
订单表ORDERS是订货信息，见表14-4。表中TRADE\_RECEIVABLE是订单的应收货款，是从订单详单表中自动汇总过来的。计算公式是: Trade Receivable= sum(订单详单表.Product Num\*订单详单表Product\_Price)-Discount。

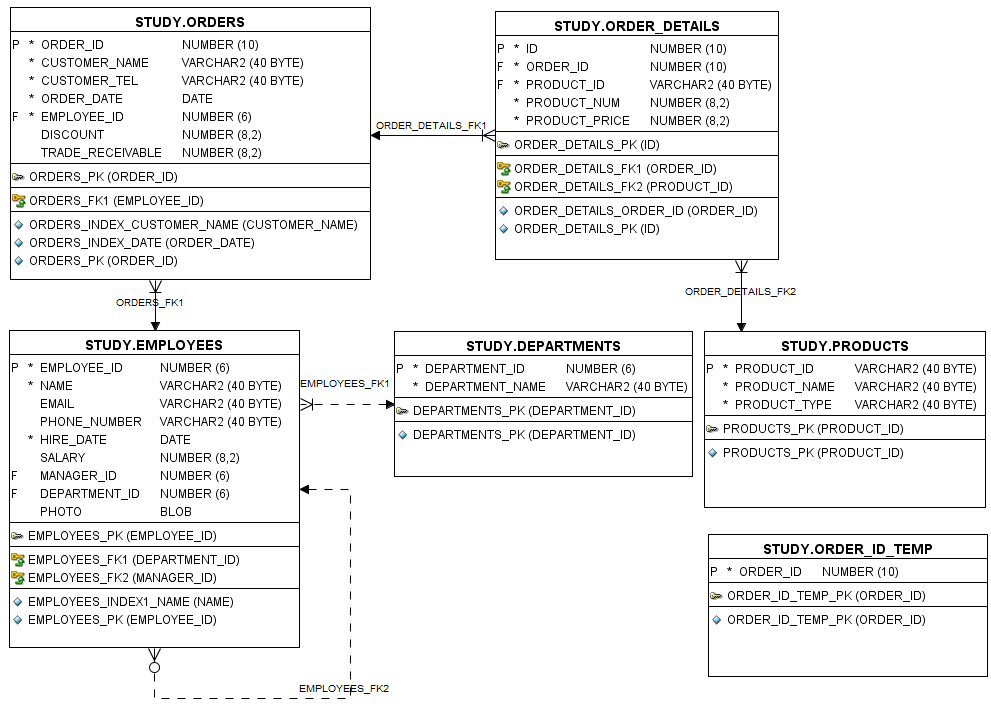


订单详单表ORDER\_ DETAILS包含订单中全部产品的信息，见表14-5。要注意，软件系统要通过触发器将产品的金额汇总到订单表中的TRADE\_RECEIVABLE属性。



为了使用触发器计算订单的应收货款，需要创建一个临时表ORDER\_ID\_TEMP,这个表的目的是暂时存储ORDER\_ID



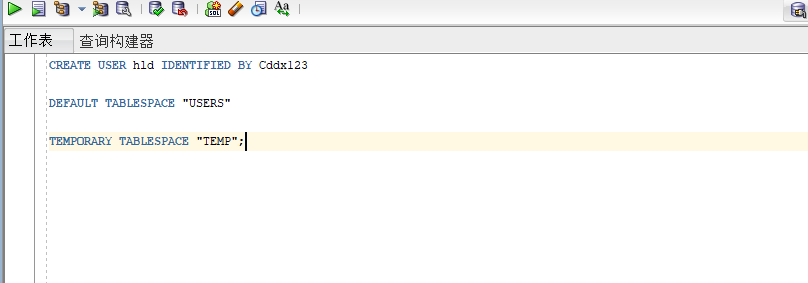


### 4. 详细设计及代码实现

4.1用户创建与空间分配：

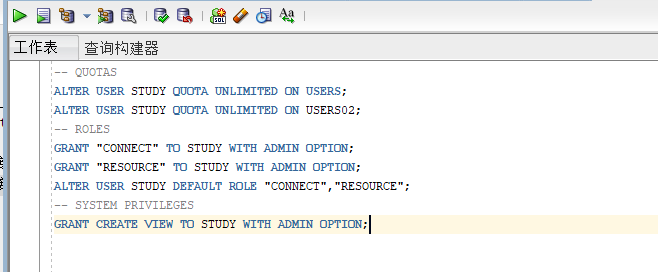
表的结构设计好之后，还要考虑用户和空间的分配，我们需要为系统新建一个用户（STUDY），在磁盘空间方面，考虑数据存储在PDB上，这里选择Oracle 12c安装时默认的PDBORCL数据库，另外，还需为销售系统创建一个新的表空间USERS02用于存储订单记录。

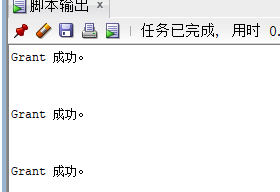
用户创建与空间分配，创建hld用户



用户分配表空间配额和授权

|  |
| --- |
| -- QUOTAS  ALTER USER STUDY QUOTA UNLIMITED ON USERS;  ALTER USER STUDY QUOTA UNLIMITED ON USERS02;  -- ROLES  GRANT "CONNECT" TO STUDY WITH ADMIN OPTION;  GRANT "RESOURCE" TO STUDY WITH ADMIN OPTION;  ALTER USER STUDY DEFAULT ROLE "CONNECT","RESOURCE";  -- SYSTEM PRIVILEGES  GRANT CREATE VIEW TO STUDY WITH ADMIN OPTION; |

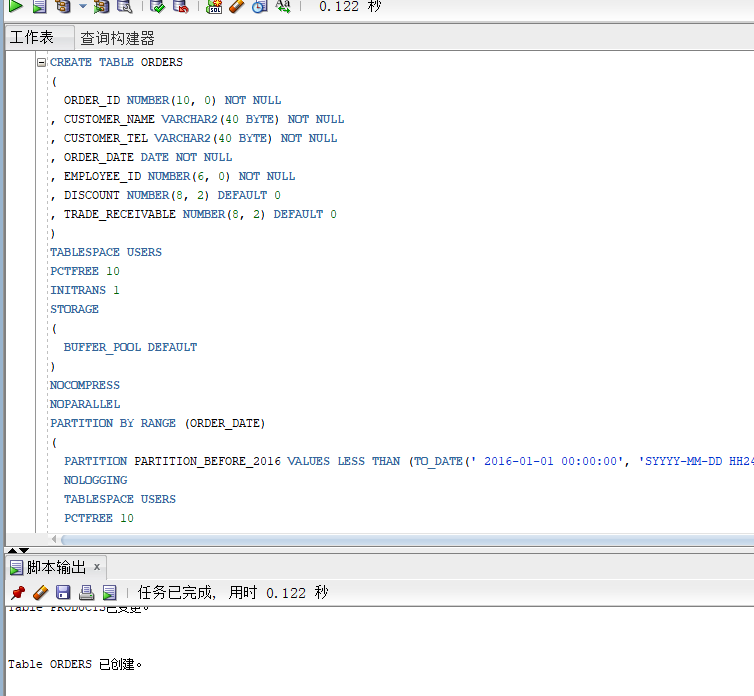


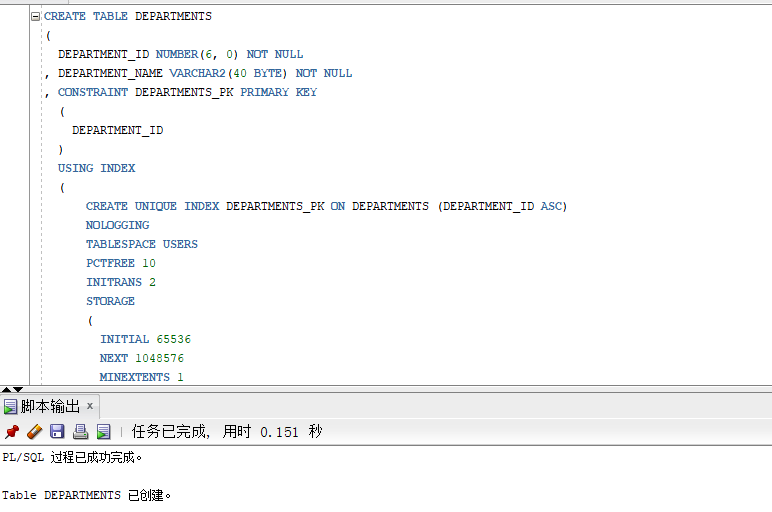
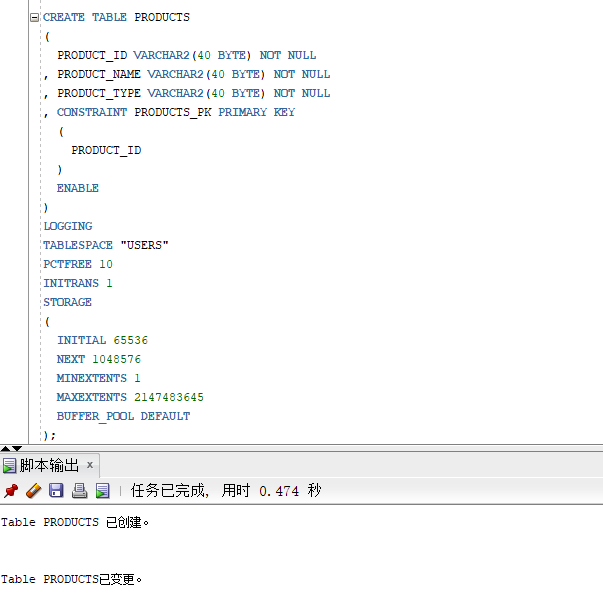
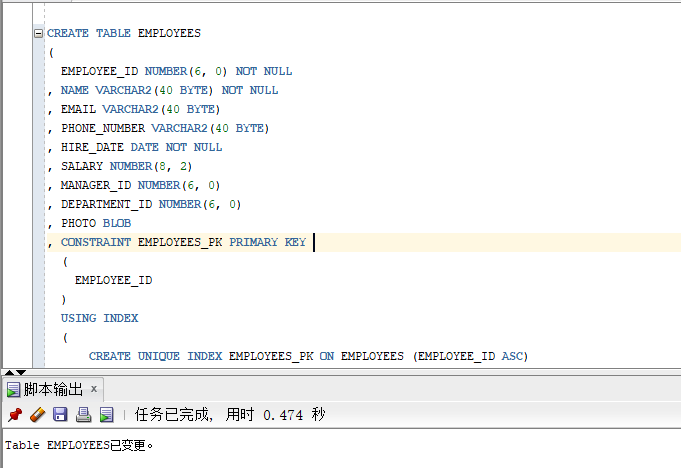


4.2创建表、约束和索引

4.2.1创建表

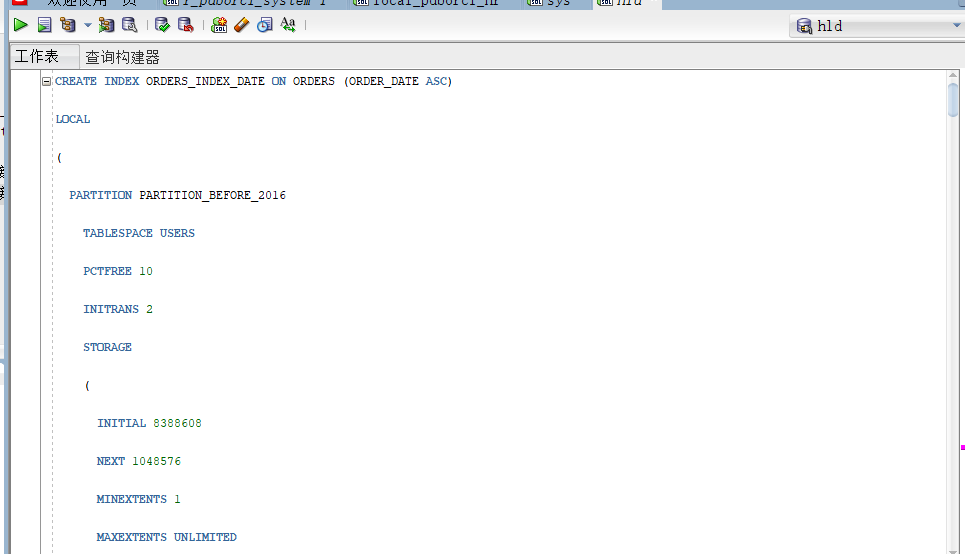
|  |
| --- |
| CREATE TABLE ORDERS  (  ORDER\_ID NUMBER(10, 0) NOT NULL  , CUSTOMER\_NAME VARCHAR2(40 BYTE) NOT NULL  , CUSTOMER\_TEL VARCHAR2(40 BYTE) NOT NULL  , ORDER\_DATE DATE NOT NULL  , EMPLOYEE\_ID NUMBER(6, 0) NOT NULL  , DISCOUNT NUMBER(8, 2) DEFAULT 0  , TRADE\_RECEIVABLE NUMBER(8, 2) DEFAULT 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  ); |



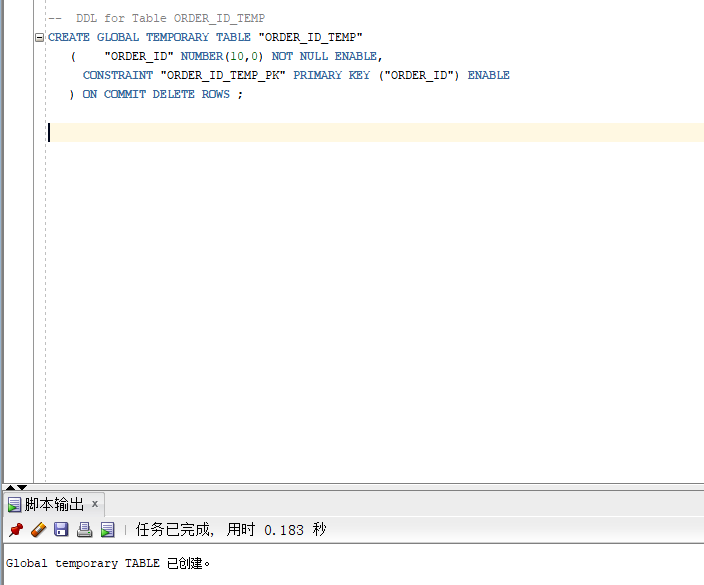


4.2.2创建本地分区索引ORDERS\_INDEX\_DATE

|  |
| --- |
| 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; |



4.2.3创建临时表空间



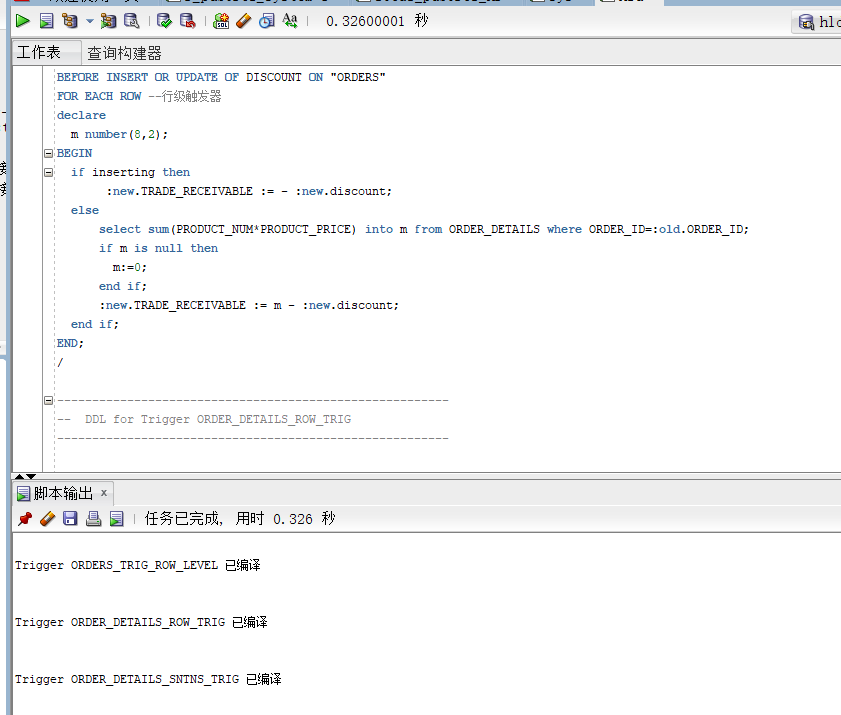
4.3创建触发器、序列和视图

4.3.1创建3个触发器

|  |
| --- |
| --------------------------------------------------------  -- DDL for Trigger ORDERS\_TRIG\_ROW\_LEVEL  --------------------------------------------------------  CREATE OR REPLACE EDITIONABLE TRIGGER "ORDERS\_TRIG\_ROW\_LEVEL"  BEFORE INSERT OR UPDATE OF DISCOUNT ON "ORDERS"  FOR EACH ROW --行级触发器  declare  m number(8,2);  BEGIN  if inserting then  :new.TRADE\_RECEIVABLE := - :new.discount;  else  select sum(PRODUCT\_NUM\*PRODUCT\_PRICE) into m from ORDER\_DETAILS where ORDER\_ID=:old.ORDER\_ID;  if m is null then  m:=0;  end if;  :new.TRADE\_RECEIVABLE := m - :new.discount;  end if;  END; |
|  |

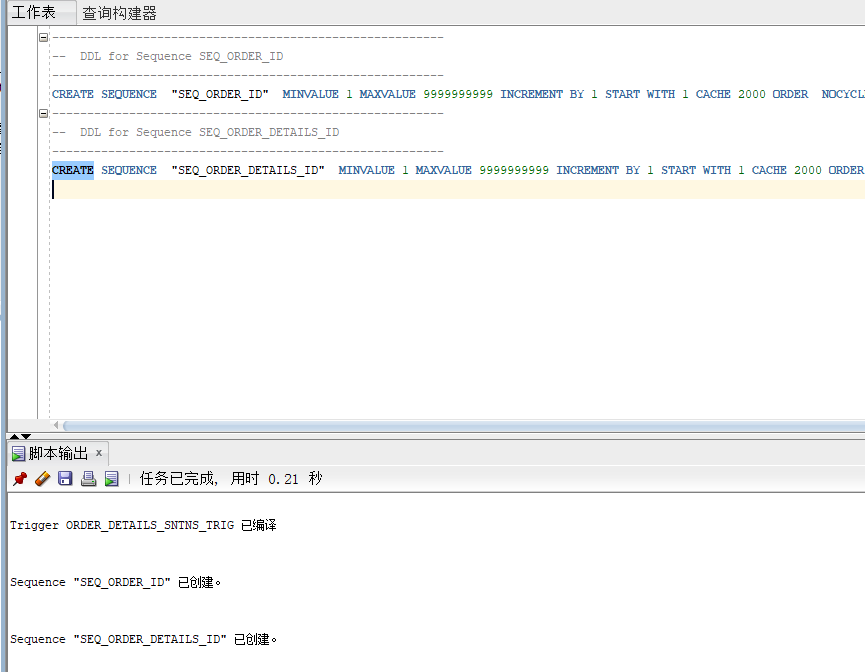
|  |
| --- |
| -------------------------------------------------------  -- DDL for Trigger ORDER\_DETAILS\_ROW\_TRIG  --------------------------------------------------------  CREATE OR REPLACE EDITIONABLE TRIGGER "ORDER\_DETAILS\_ROW\_TRIG"  AFTER DELETE OR INSERT OR UPDATE ON ORDER\_DETAILS  FOR EACH ROW  BEGIN  --DBMS\_OUTPUT.PUT\_LINE(:NEW.ORDER\_ID);  IF :NEW.ORDER\_ID IS NOT NULL THEN  MERGE INTO ORDER\_ID\_TEMP A  USING (SELECT 1 FROM DUAL) B  ON (A.ORDER\_ID=:NEW.ORDER\_ID)  WHEN NOT MATCHED THEN  INSERT (ORDER\_ID) VALUES(:NEW.ORDER\_ID);  END IF;  IF :OLD.ORDER\_ID IS NOT NULL THEN  MERGE INTO ORDER\_ID\_TEMP A  USING (SELECT 1 FROM DUAL) B  ON (A.ORDER\_ID=:OLD.ORDER\_ID)  WHEN NOT MATCHED THEN  INSERT (ORDER\_ID) VALUES(:OLD.ORDER\_ID);  END IF;  END; |
|  |

|  |
| --- |
| -------------------------------------------------------  -- DDL for Trigger ORDER\_DETAILS\_SNTNS\_TRIG  --------------------------------------------------------  CREATE OR REPLACE EDITIONABLE TRIGGER "ORDER\_DETAILS\_SNTNS\_TRIG"  AFTER DELETE OR INSERT OR UPDATE ON ORDER\_DETAILS  declare  m number(8,2);  BEGIN  FOR R IN (SELECT ORDER\_ID FROM ORDER\_ID\_TEMP)  LOOP  --DBMS\_OUTPUT.PUT\_LINE(R.ORDER\_ID);  select sum(PRODUCT\_NUM\*PRODUCT\_PRICE) into m from ORDER\_DETAILS  where ORDER\_ID=R.ORDER\_ID;  if m is null then  m:=0;  end if;  UPDATE ORDERS SET TRADE\_RECEIVABLE = m - discount  WHERE ORDER\_ID=R.ORDER\_ID;  END LOOP;  delete from ORDER\_ID\_TEMP; --这句话很重要，否则可能一直不释放空间，后继插入会非常慢。  END; |



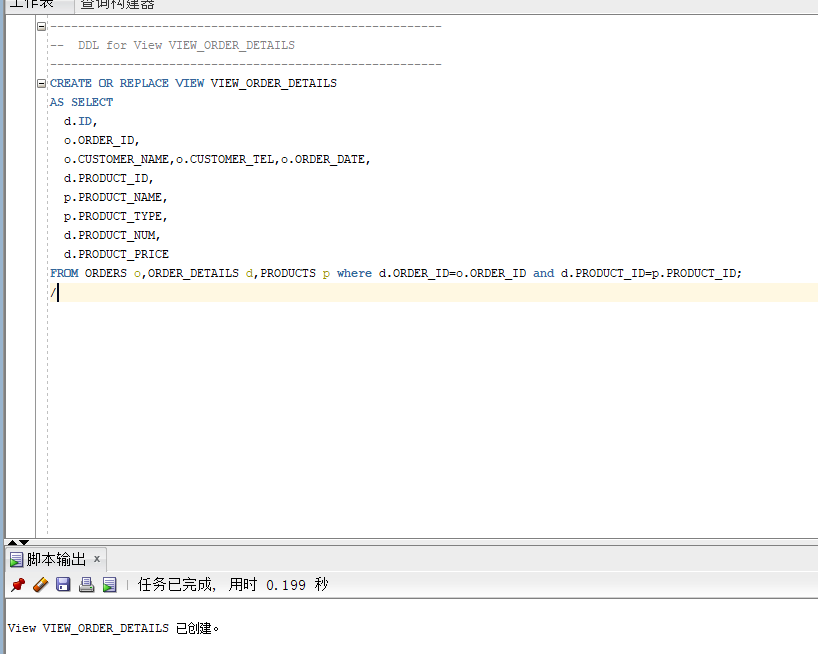
4.3.2创建序列

|  |
| --- |
| --------------------------------------------------------  -- DDL for Sequence SEQ\_ORDER\_ID  --------------------------------------------------------  CREATE SEQUENCE "SEQ\_ORDER\_ID" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 1 CACHE 2000 ORDER NOCYCLE NOPARTITION ;  --------------------------------------------------------  -- DDL for Sequence SEQ\_ORDER\_DETAILS\_ID  --------------------------------------------------------  CREATE SEQUENCE "SEQ\_ORDER\_DETAILS\_ID" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 1 CACHE 2000 ORDER NOCYCLE NOPARTITION ; |



4.3.3创建视图

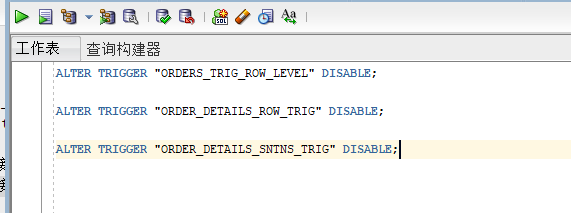
|  |
| --- |
| --------------------------------------------------------  -- DDL for View VIEW\_ORDER\_DETAILS  --------------------------------------------------------  CREATE OR REPLACE VIEW VIEW\_ORDER\_DETAILS  AS SELECT  d.ID,  o.ORDER\_ID,  o.CUSTOMER\_NAME,o.CUSTOMER\_TEL,o.ORDER\_DATE,  d.PRODUCT\_ID,  p.PRODUCT\_NAME,  p.PRODUCT\_TYPE,  d.PRODUCT\_NUM,  d.PRODUCT\_PRICE  FROM ORDERS o,ORDER\_DETAILS d,PRODUCTS p where d.ORDER\_ID=o.ORDER\_ID and d.PRODUCT\_ID=p.PRODUCT\_ID; |



4.4插入数据与数据库测试

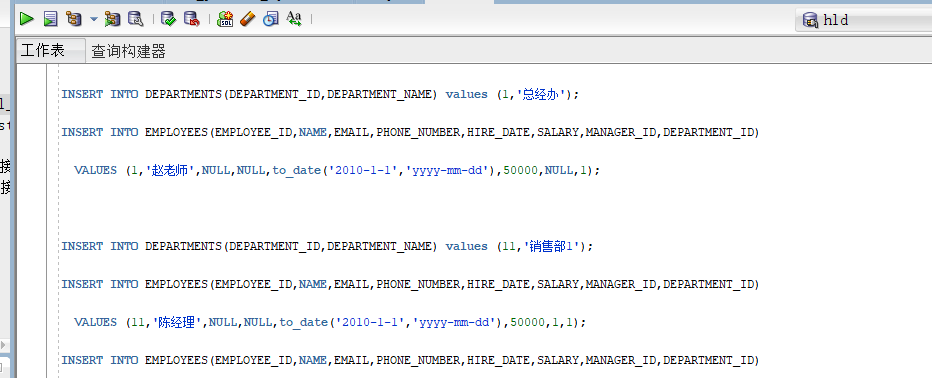
4.4.1批量插入订单数据之前，禁用触发器

|  |
| --- |
| ALTER TRIGGER "ORDERS\_TRIG\_ROW\_LEVEL" DISABLE;  ALTER TRIGGER "ORDER\_DETAILS\_ROW\_TRIG" DISABLE;  ALTER TRIGGER "ORDER\_DETAILS\_SNTNS\_TRIG" DISABLE; |



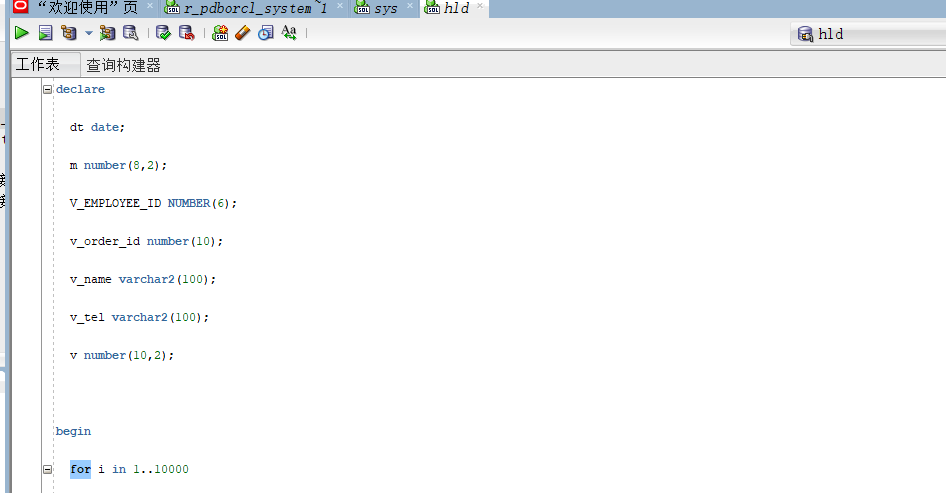
插入DEPARTMENTS，EMPLOYEES数据

|  |
| --- |
| INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (1,'总经办');  INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)  VALUES (1,'赵老师',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,NULL,1);  INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (11,'销售部1');  INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)  VALUES (11,'陈经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,1,1);  INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)  VALUES (111,'吴经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,11,11);  INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)  VALUES (112,'白经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,11,11);  INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (12,'销售部2');  INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)  VALUES (12,'王总',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,1,1);  INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)  VALUES (121,'赵经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,12,12);  INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)  VALUES (122,'刘经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,12,12); |
|  |



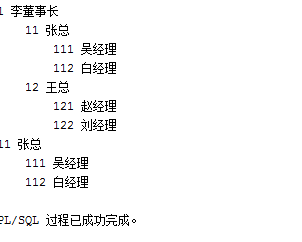
批量插入订单数据

|  |
| --- |
| declare  dt date;  m number(8,2);  V\_EMPLOYEE\_ID NUMBER(6);  v\_order\_id number(10);  v\_name varchar2(100);  v\_tel 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\_EMPLOYEE\_ID:=CASE I MOD 6 WHEN 0 THEN 11 WHEN 1 THEN 111 WHEN 2 THEN 112  WHEN 3 THEN 12 WHEN 4 THEN 121 ELSE 122 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。  v\_name := 'aa'|| 'aa';  v\_name := 'zhang' || i;  v\_tel := '139888883' || i;  insert /\*+append\*/ into ORDERS (ORDER\_ID,CUSTOMER\_NAME,CUSTOMER\_TEL,ORDER\_DATE,EMPLOYEE\_ID,DISCOUNT)  values (v\_order\_id,v\_name,v\_tel,dt,V\_EMPLOYEE\_ID,dbms\_random.value(100,0));  --插入订单y一个订单包括3个产品  v:=dbms\_random.value(10000,4000);  v\_name:='computer'|| (i mod 3 + 1);  insert /\*+append\*/ into ORDER\_DETAILS(ID,ORDER\_ID,PRODUCT\_ID,PRODUCT\_NUM,PRODUCT\_PRICE)  values (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,2,v);  v:=dbms\_random.value(1000,50);  v\_name:='paper'|| (i mod 3 + 1);  insert /\*+append\*/ into ORDER\_DETAILS(ID,ORDER\_ID,PRODUCT\_ID,PRODUCT\_NUM,PRODUCT\_PRICE)  values (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,3,v);  v:=dbms\_random.value(9000,2000);  v\_name:='phone'|| (i mod 3 + 1);  insert /\*+append\*/ into ORDER\_DETAILS(ID,ORDER\_ID,PRODUCT\_ID,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 - discount 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; |



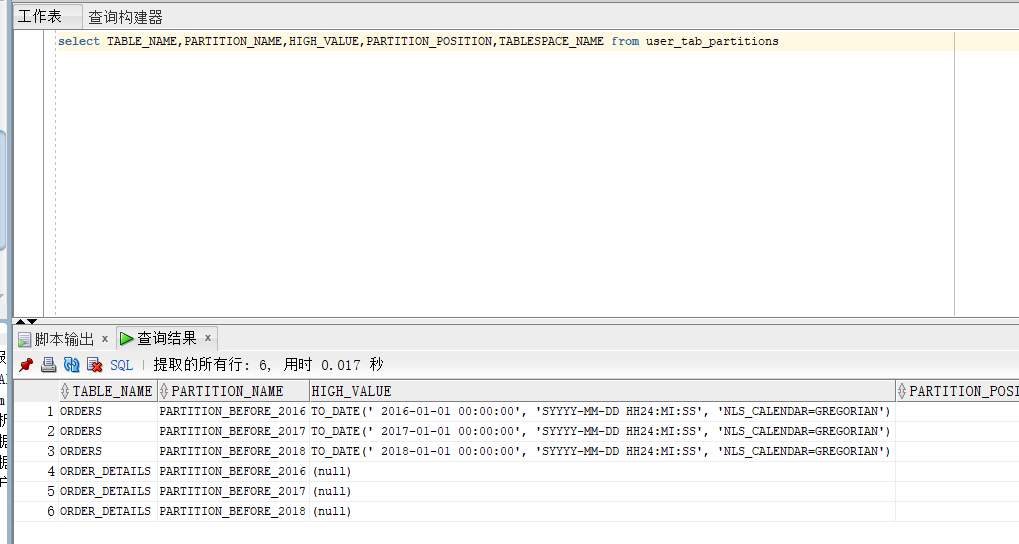
4.4.2数据库测试：

|  |
| --- |
| set serveroutput on  DECLARE  V\_EMPLOYEE\_ID NUMBER;  BEGIN  V\_EMPLOYEE\_ID := 1;  MYPACK.Get\_Employees ( V\_EMPLOYEE\_ID => V\_EMPLOYEE\_ID) ;  V\_EMPLOYEE\_ID := 11;  MYPACK.Get\_Employees ( V\_EMPLOYEE\_ID => V\_EMPLOYEE\_ID) ;  END; |



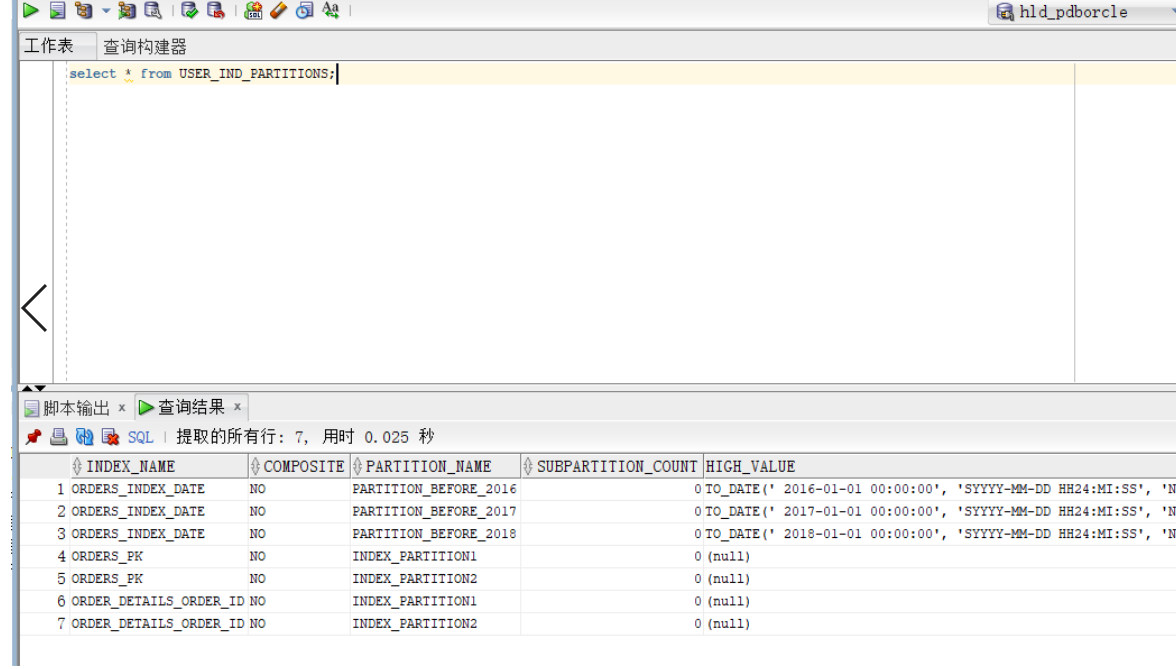
4.4.3查询分区表情况

|  |
| --- |
| select TABLE\_NAME,PARTITION\_NAME,HIGH\_VALUE,PARTITION\_POSITION,TABLESPACE\_NAME from user\_tab\_partitions |



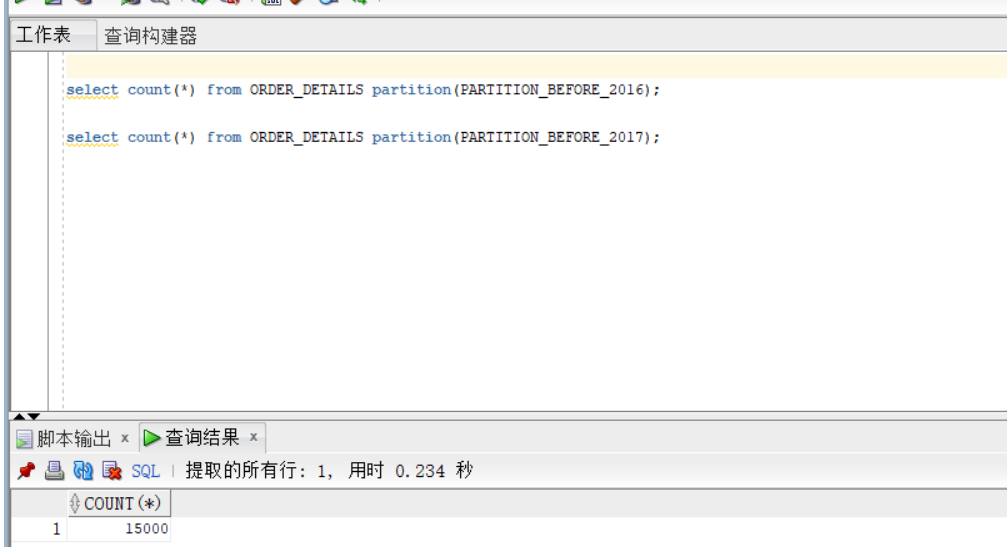
查询分区索引情况

|  |
| --- |
| select \* from USER\_IND\_PARTITIONS; |



查询一个分区中的数据

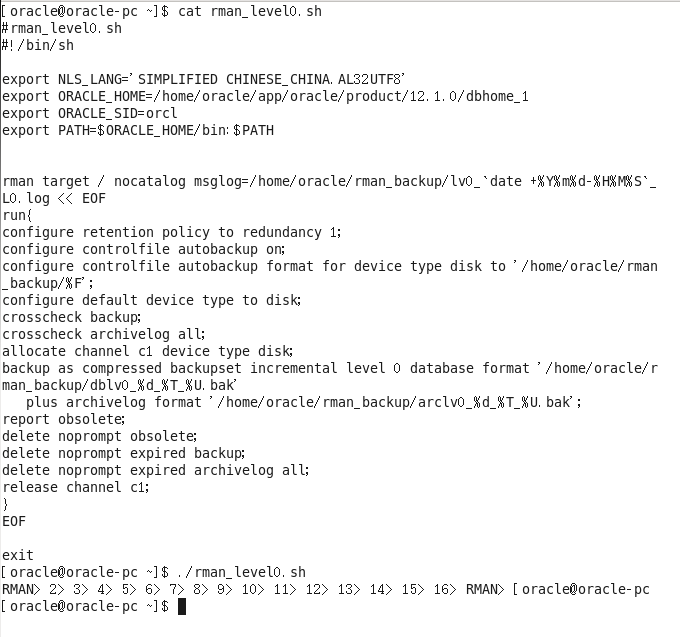
|  |
| --- |
| select count(\*) from ORDERS partition(PARTITION\_BEFORE\_2016);  select count(\*) from ORDERS partition(PARTITION\_BEFORE\_2017); |



5.手工全备份和全恢复

5.1.1开始全备份

|  |
| --- |
| [oracle@oracle-pc ~]$ cat rman\_level0.sh  [oracle@oracle-pc ~]$ ./rman\_level0.sh |



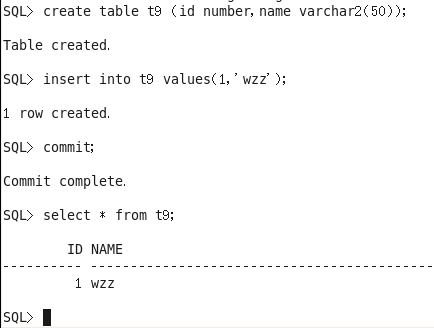
5.1.2每天定时开始增量备份（可选）

|  |
| --- |
| [oracle@oracle-pc ~]$ cat rman\_level1.sh  [oracle@oracle-pc ~]$ ./rman\_level1.sh |



备份后修改数据

|  |
| --- |
| [oracle@oracle-pc ~]$ sqlplus study/123@pdborcl  SQL> create table t1 (id number,name varchar2(50));  SQL> insert into t1 values(1,'zhang');1 row created.  SQL> commit;  SQL> select \* from t1;  SQL> exit |



5.1.3删除数据库文件

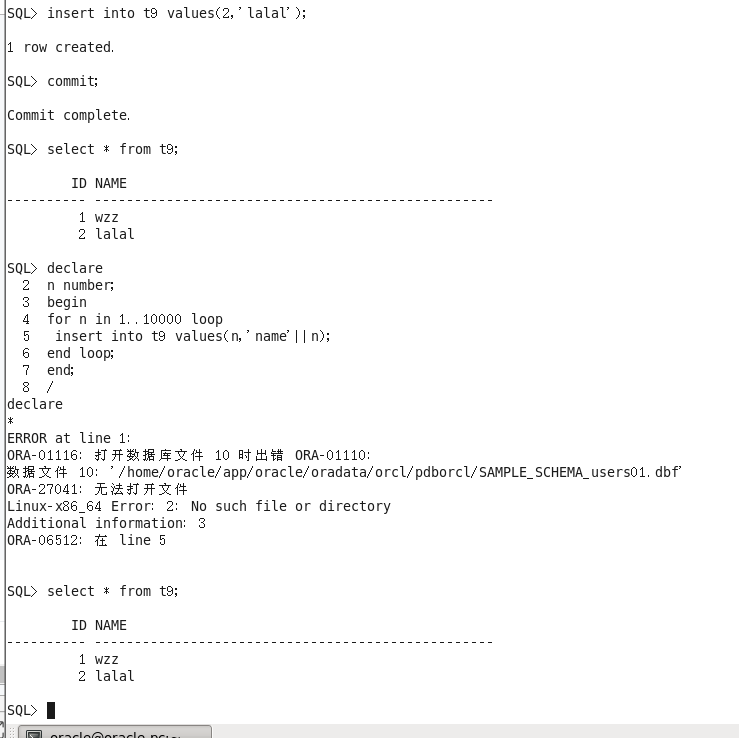
模拟数据库文件损坏**（**删除数据文件后，仍然可以增加一条数据。这是因为增加的数据并没有写入数据文件，而是写到了日志文件中。如果增加的数据较多的时候，就会出问题了。**）**

删除

|  |
| --- |
| [oracle@oracle-pc ~]$ rm /home/oracle/app/oracle/oradata/orcl/pdborcl/SAMPLE\_SCHEMA\_users01.dbf |



|  |
| --- |
| [oracle@oracle-pc ~]$ sqlplus study/123@pdborcl  SQL> insert into t1 values(2,'wang');  SQL> commit;  SQL> select \* from t1;  SQL> declare  2 n number;  3 begin  4 for n in 1..10000 loop  5 insert into t1 values(n,'name'||n);  6 end loop;  7 end;  SQL> select \* from t1;  SQL> exit |

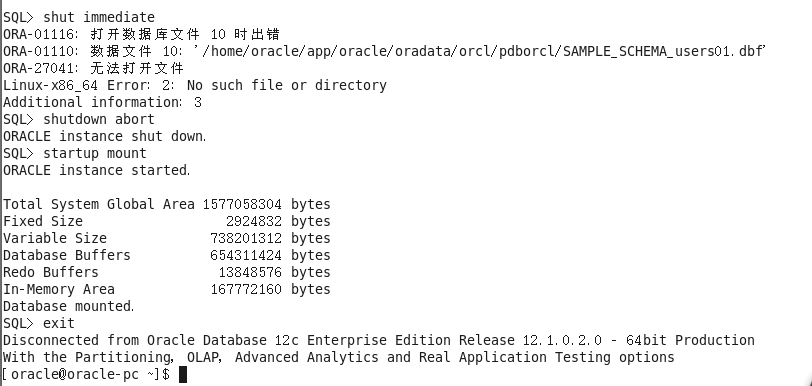


5.2开始全恢复

5.2.1重启损坏的数据库到mount状态

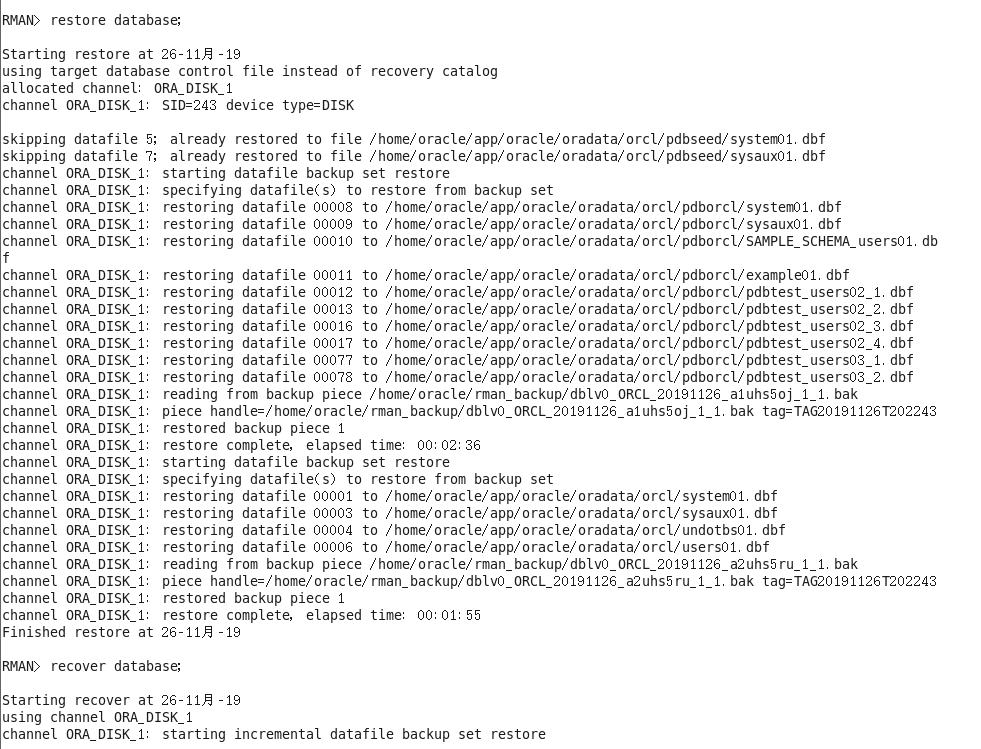
-通过shutdown immediate无法正常关闭数据库，只能通过shutdown abort强制关闭。然后将数据库启动到mount状态。

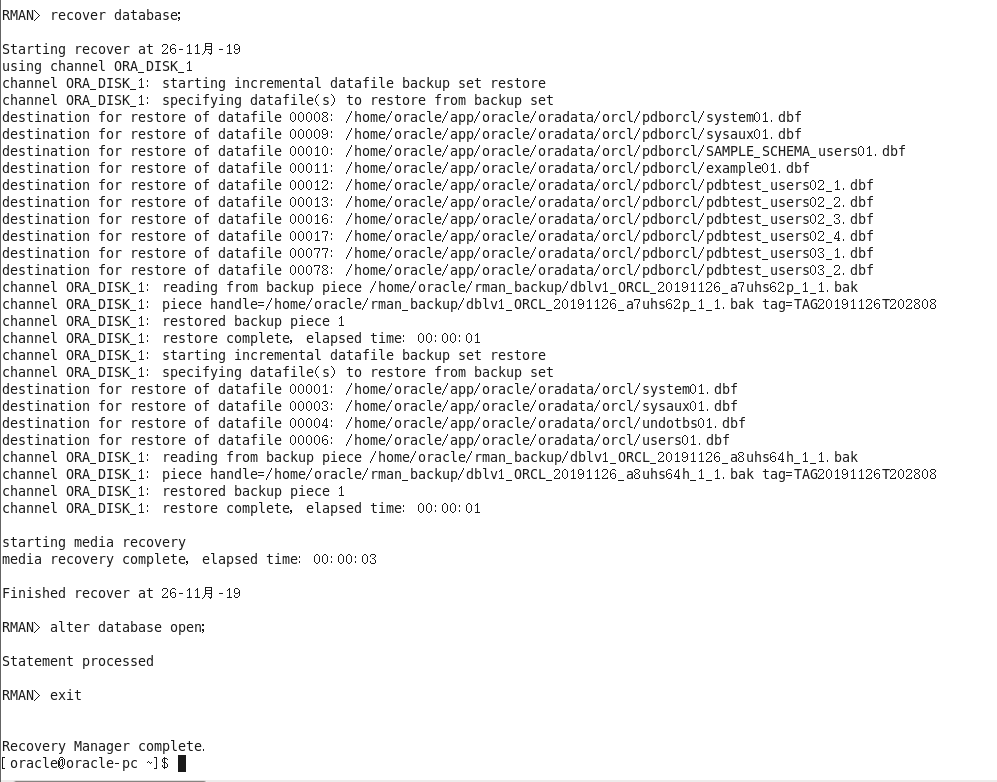
|  |
| --- |
| [oracle@oracle-pc ~]$ sqlplus / as sysdba  SQL> shutdown immediate  SQL> shutdown abort  SQL> startup mount  SQL> exit |



5.2.2开始恢复数据库

|  |
| --- |
| [oracle@oracle-pc ~]$ rman target /  RMAN> restore database ;  RMAN> recover database;  RMAN> alter database open;  Statement processed  RMAN> exit |



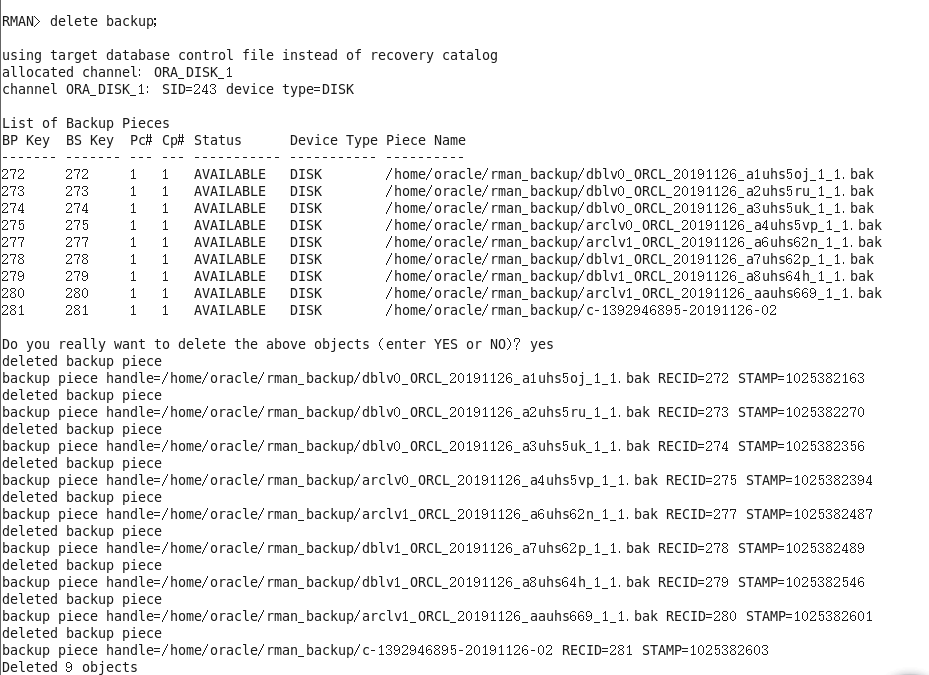


5.2.3

|  |
| --- |
| [oracle@oracle-pc ~]$ sqlplus study/123@pdborcl  SQL> select \* from t9; |

5.2.4删除备份集（可选）

|  |
| --- |
| [oracle@oracle-pc ~]$ rman target /  RMAN> delete backup;  RMAN> exit |



5.2.5删除备份集之后，备份文件也删除了。只留下日志文件。

|  |
| --- |
| [oracle@oracle-pc ~]$ cd rman\_backup  [oracle@oracle-pc rman\_backup]$ ls |



### 6总结

本次实验中，发现自己在Oracle方面有许多不足，虽然上课有听讲。但是发现有些知识点只有你自己运用之后，才会发现自己的理解不够，因此需要量的练习和操作，做到熟练，才能不用看课本就可以写一些基本的东西。