

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的图书管理系统 | | |
| 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 | 陈云 | 学号 | 201810414408 |
| 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

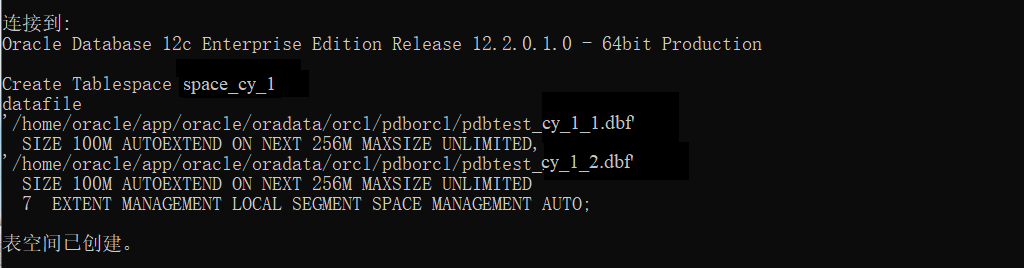
**实验6（期末考核）基于Oracle的图书系统数据库设计**

图书管理系统是学校管理图书的重要工具，是学校不可或缺的一部分。随着在校人数的不断增加，系统的数量也不断的上涨。学校工作繁杂，资料众多，人工管理信息的难度也越来越大，显然是不能满足实际的需要，效率也是很低的。并且这种传统的方式存在着众多的弊端，如：保密性差.查询不便.效率低，很难维护和更新等，然而，本系统针对以上的缺点能够极大的提高图书信息管理的效率，也是科学化.正规化的管理，与世界接轨的重要条件。所以如何自动高效地管理信息是这些年来许多人所研究的。

**一．创建表空间**

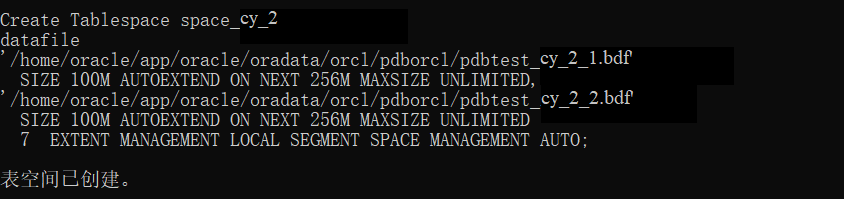
space\_cy\_1

 Create Tablespace space\_cy\_1  
 datafile  
 '/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_cy\_1\_1.dbf'  
  SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED,  
 '/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_cy\_1\_2.dbf'  
  SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED  
 EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;



space\_cy\_2

 Create Tablespace space\_cy\_2  
 datafile  
 '/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_cy\_2\_1.dbf'  
  SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED,  
 '/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_cy\_2\_2.dbf'  
  SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED  
 EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;



1. **创建角色、表及用户分配权限**

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 (Tablespace) 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。

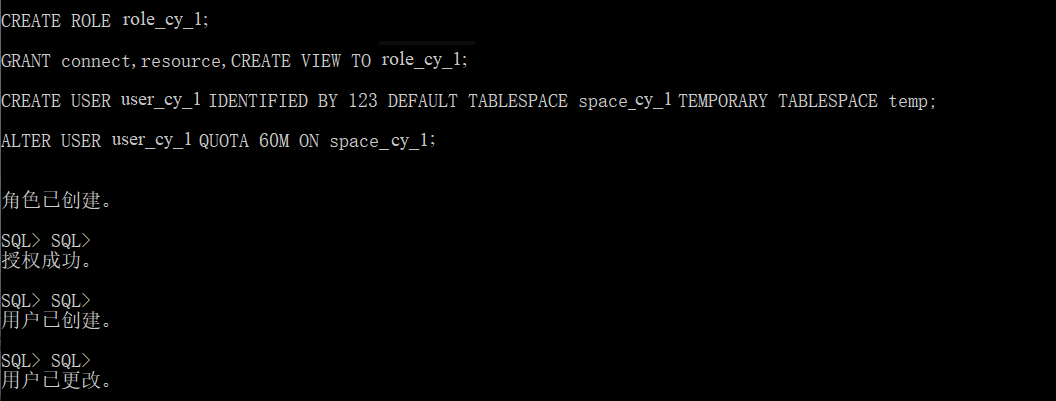
1. **用户默认使用表空间space\_cy\_1** **创建第一个角色和用户**

①创建角色role\_cy\_1,将connect,resource,create view授权给role\_cy\_1

②创建用户user\_cy\_1

③分配60M空间给xgh\_1并将角色xgh1授权给用户xgh\_1

 CREATE ROLE role\_cy\_1;  
 ​  
 GRANT connect,resource,CREATE VIEW TO role\_cy\_1;  
 ​  
 CREATE USER user\_cy\_1 IDENTIFIED BY 123 DEFAULT TABLESPACE space\_cy\_1 TEMPORARY TABLESPACE temp;  
 ​  
 ALTER USER user\_cy\_1 QUOTA 60M ON space\_cy\_1;  
 ​  
 GRANT role\_cy\_1 TO user\_cy\_1;



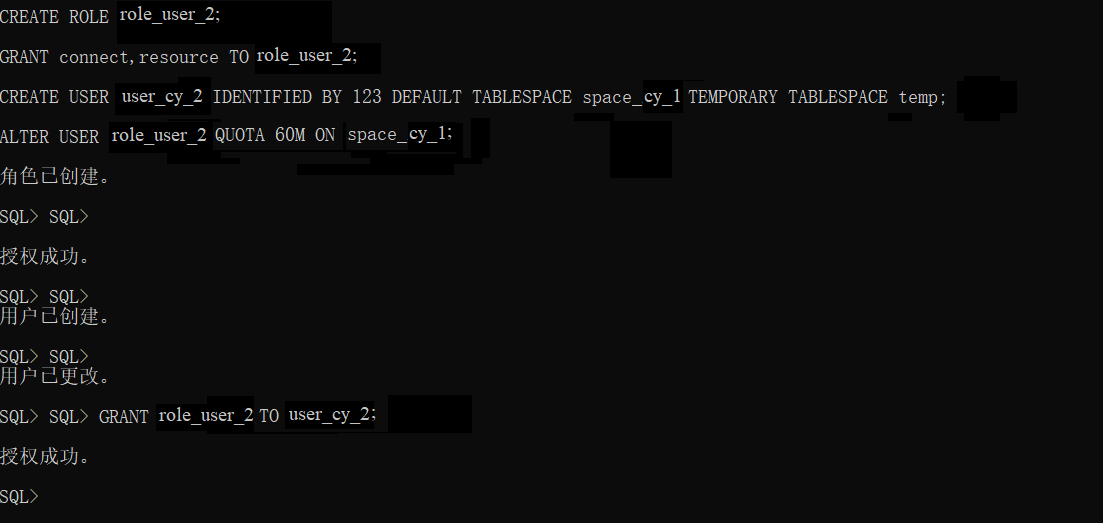
**2.创建第二个角色和用户**

①创建角色role\_cy\_2，将connect,resource权限给role\_cy\_2

②创建用户user\_cy\_2

③分配60M空间给user\_cy\_2并将角色role\_cy\_2授权给用户user\_cy\_2

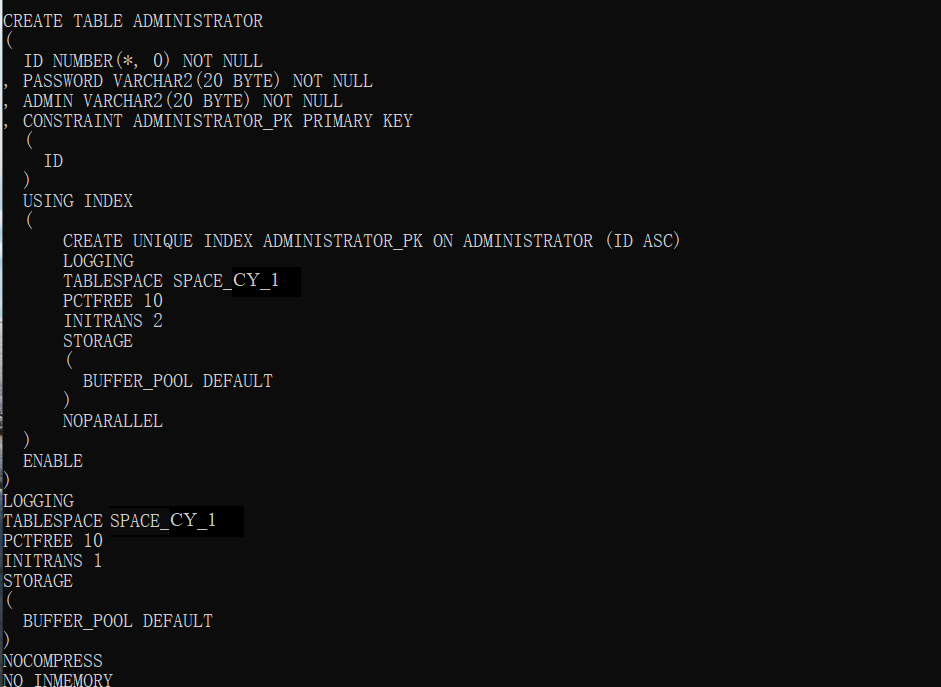
 CREATE ROLE role\_cy\_2;  
 ​  
 GRANT connect,resource TO role\_cy\_2;  
 ​  
 CREATE USER user\_cy\_2 IDENTIFIED BY 123 DEFAULT TABLESPACE space\_cy\_1 TEMPORARY TABLESPACE temp;  
 ​  
 ALTER USER user\_cy\_2 QUOTA 60M ON space\_cy\_1;  
 ​  
 GRANT role\_cy\_2 TO user\_cy\_2;



**3. 在用户user\_cy\_1下创建表**

**①创建管理员表——**id为主键

 CREATE TABLE ADMINISTRATOR   
 (  
  ID NUMBER(\*, 0) NOT NULL   
 , PASSWORD VARCHAR2(20 BYTE) NOT NULL   
 , ADMIN VARCHAR2(20 BYTE) NOT NULL   
 , CONSTRAINT ADMINISTRATOR\_PK PRIMARY KEY   
  (  
    ID   
  )  
  USING INDEX   
  (  
       CREATE UNIQUE INDEX ADMINISTRATOR\_PK ON ADMINISTRATOR (ID ASC)   
      LOGGING   
      TABLESPACE SPACE\_CY\_!  
      PCTFREE 10   
      INITRANS 2   
      STORAGE   
      (   
        BUFFER\_POOL DEFAULT   
      )   
      NOPARALLEL   
  )  
  ENABLE   
 )   
 LOGGING   
 TABLESPACE SPACE\_CY\_1  
 PCTFREE 10   
 INITRANS 1   
 STORAGE   
 (   
  BUFFER\_POOL DEFAULT   
 )   
 NOCOMPRESS   
 NO INMEMORY   
 NOPARALLEL;

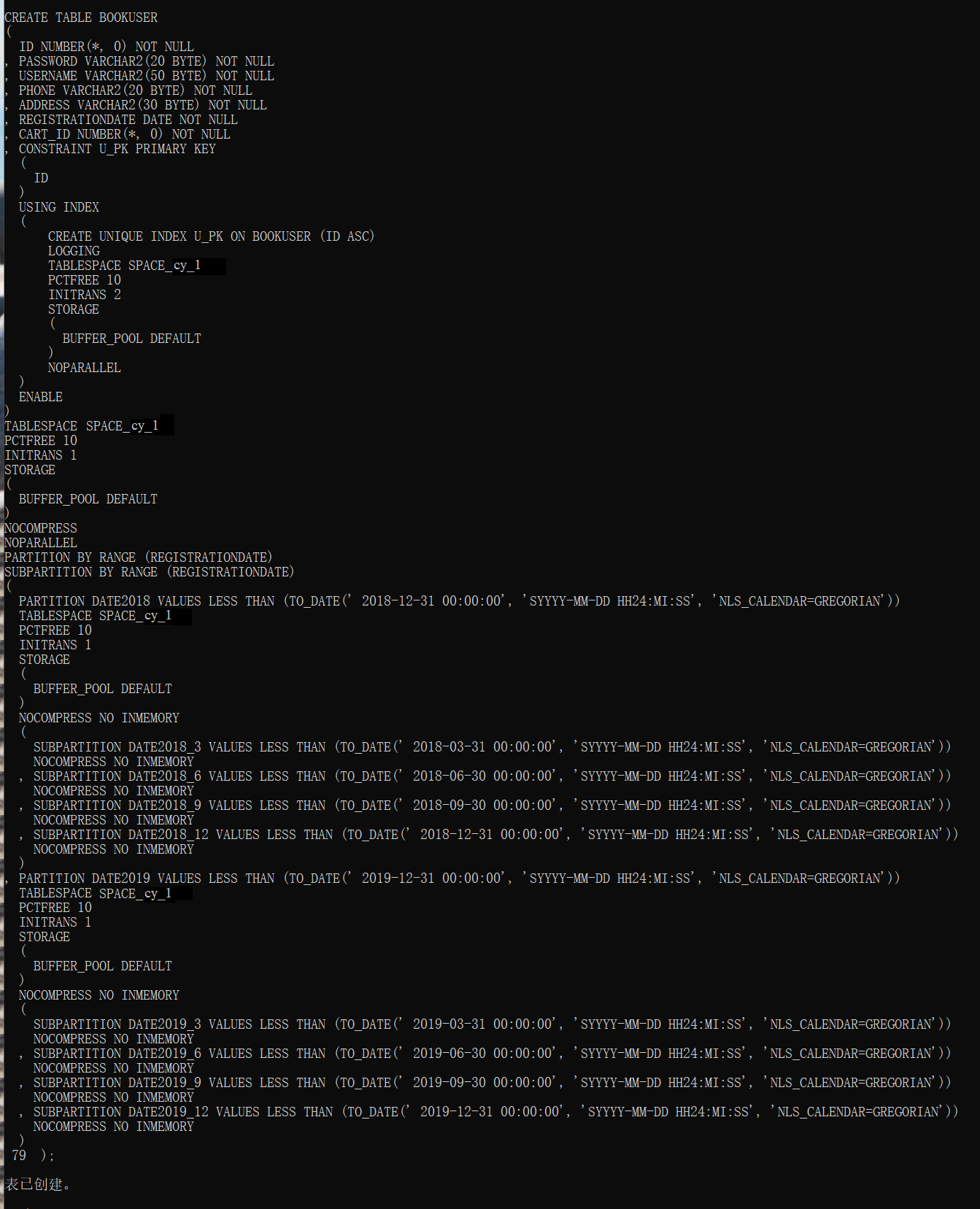
****

**②创建用户表**

id为主键；根据注册日期按范围分区

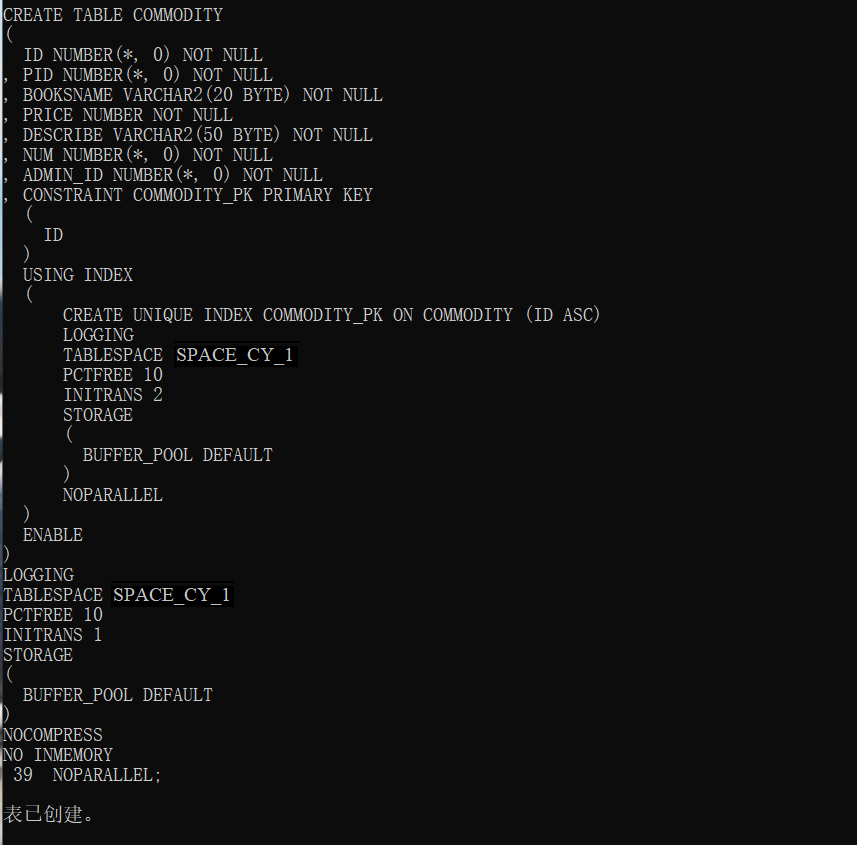
分为2018和2019年两个分区，每年按季度划4个子分区

 CREATE TABLE BOOKUSER   
 (  
  ID NUMBER(\*, 0) NOT NULL   
 , PASSWORD VARCHAR2(20 BYTE) NOT NULL   
 , USERNAME VARCHAR2(50 BYTE) NOT NULL   
 , PHONE VARCHAR2(20 BYTE) NOT NULL   
 , ADDRESS VARCHAR2(30 BYTE) NOT NULL   
 , REGISTRATIONDATE DATE NOT NULL   
 , CART\_ID NUMBER(\*, 0) NOT NULL   
 , CONSTRAINT U\_PK PRIMARY KEY   
  (  
    ID   
  )  
  USING INDEX   
  (  
       CREATE UNIQUE INDEX U\_PK ON BOOKUSER (ID ASC)   
      LOGGING   
      TABLESPACE SPACE\_CY\_1  
      PCTFREE 10   
      INITRANS 2   
      STORAGE   
      (   
        BUFFER\_POOL DEFAULT   
      )   
      NOPARALLEL   
  )  
  ENABLE   
 )   
 TABLESPACE SPACE\_CY\_1  
 PCTFREE 10   
 INITRANS 1   
 STORAGE   
 (   
  BUFFER\_POOL DEFAULT   
 )   
 NOCOMPRESS   
 NOPARALLEL   
 PARTITION BY RANGE (REGISTRATIONDATE)   
 SUBPARTITION BY RANGE (REGISTRATIONDATE)   
 (  
  PARTITION DATE2018 VALUES LESS THAN (TO\_DATE(' 2018-12-31 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
  TABLESPACE SPACE\_CY\_1  
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY   
  (  
    SUBPARTITION DATE2018\_3 VALUES LESS THAN (TO\_DATE(' 2018-03-31 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
    NOCOMPRESS NO INMEMORY    
  , SUBPARTITION DATE2018\_6 VALUES LESS THAN (TO\_DATE(' 2018-06-30 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
    NOCOMPRESS NO INMEMORY    
  , SUBPARTITION DATE2018\_9 VALUES LESS THAN (TO\_DATE(' 2018-09-30 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
    NOCOMPRESS NO INMEMORY    
  , SUBPARTITION DATE2018\_12 VALUES LESS THAN (TO\_DATE(' 2018-12-31 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
    NOCOMPRESS NO INMEMORY    
  )    
 , PARTITION DATE2019 VALUES LESS THAN (TO\_DATE(' 2019-12-31 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
  TABLESPACE SPACE\_CY\_1  
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY   
  (  
    SUBPARTITION DATE2019\_3 VALUES LESS THAN (TO\_DATE(' 2019-03-31 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
    NOCOMPRESS NO INMEMORY    
  , SUBPARTITION DATE2019\_6 VALUES LESS THAN (TO\_DATE(' 2019-06-30 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
    NOCOMPRESS NO INMEMORY    
  , SUBPARTITION DATE2019\_9 VALUES LESS THAN (TO\_DATE(' 2019-09-30 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
    NOCOMPRESS NO INMEMORY    
  , SUBPARTITION DATE2019\_12 VALUES LESS THAN (TO\_DATE(' 2019-12-31 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))   
    NOCOMPRESS NO INMEMORY    
  )    
 );

****

**③创建商品表**

 CREATE TABLE COMMODITY   
 (  
  ID NUMBER(\*, 0) NOT NULL   
 , PID NUMBER(\*, 0) NOT NULL   
 , BOOKSNAME VARCHAR2(20 BYTE) NOT NULL   
 , PRICE NUMBER NOT NULL   
 , DESCRIBE VARCHAR2(50 BYTE) NOT NULL   
 , NUM NUMBER(\*, 0) NOT NULL   
 , ADMIN\_ID NUMBER(\*, 0) NOT NULL   
 , CONSTRAINT COMMODITY\_PK PRIMARY KEY   
  (  
    ID   
  )  
  USING INDEX   
  (  
       CREATE UNIQUE INDEX COMMODITY\_PK ON COMMODITY (ID ASC)   
      LOGGING   
      TABLESPACE SPACE\_CY\_1  
      PCTFREE 10   
      INITRANS 2   
      STORAGE   
      (   
        BUFFER\_POOL DEFAULT   
      )   
      NOPARALLEL   
  )  
  ENABLE   
 )   
 LOGGING   
 TABLESPACE SPACE\_CY\_1   
 PCTFREE 10   
 INITRANS 1   
 STORAGE   
 (   
  BUFFER\_POOL DEFAULT   
 )   
 NOCOMPRESS   
 NO INMEMORY   
 NOPARALLEL;

****

**④创建购物车表**

用户表字段BOOKUSER\_ID为购物车表的外键

购物车采用引用分区

 CREATE TABLE CART   
 (  
  ID NUMBER(\*, 0) NOT NULL   
 , AMOUNT NUMBER(\*, 0) NOT NULL   
 , PID NUMBER(\*, 0) NOT NULL   
 , BOOKUSER\_ID NUMBER(\*, 0) NOT NULL   
 , CONSTRAINT CART\_PK PRIMARY KEY   
  (  
    ID   
  )  
  USING INDEX   
  (  
       CREATE UNIQUE INDEX CART\_PK ON CART (ID ASC)   
      LOGGING   
      TABLESPACE SPACE\_CY\_1   
      PCTFREE 10   
      INITRANS 2   
      STORAGE   
      (   
        BUFFER\_POOL DEFAULT   
      )   
      NOPARALLEL   
  )  
  ENABLE   
 , CONSTRAINT CART\_BOOKUSER FOREIGN KEY  
  (  
  BOOKUSER\_ID   
  )  
  REFERENCES BOOKUSER  
  (  
  CART\_ID   
  )  
  ENABLE   
 )   
 PCTFREE 10   
 PCTUSED 40   
 INITRANS 1   
 STORAGE   
 (   
  BUFFER\_POOL DEFAULT   
 )   
 NOCOMPRESS   
 NOPARALLEL   
 PARTITION BY REFERENCE (CART\_BOOKUSER)   
 (  
  PARTITION DATE2018\_3   
  LOGGING   
  TABLESPACE SPACE\_CY\_1  
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY    
 , PARTITION DATE2018\_6   
  LOGGING   
  TABLESPACE SPACE\_CY\_1   
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY    
 , PARTITION DATE2018\_9   
  LOGGING   
  TABLESPACE SPACE\_CY\_1  
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY    
 , PARTITION DATE2018\_12   
  LOGGING   
  TABLESPACE SPACE\_CY\_1  
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY    
 , PARTITION DATE2019\_3   
  LOGGING   
  TABLESPACE SPACE\_CY\_1   
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY    
 , PARTITION DATE2019\_6   
  LOGGING   
  TABLESPACE SPACE\_CY\_1   
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY    
 , PARTITION DATE2019\_9   
  LOGGING   
  TABLESPACE SPACE\_CY\_1  
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY    
 , PARTITION DATE2019\_12   
  LOGGING   
  TABLESPACE SPACE\_CY\_1  
  PCTFREE 10   
  INITRANS 1   
  STORAGE   
  (   
    BUFFER\_POOL DEFAULT   
  )   
  NOCOMPRESS NO INMEMORY    
 );

**⑤论坛表**

CREATE TABLE TABLE1

(

ID INT NOT NULL

, CONTENT NVARCHAR2(50) NOT NULL

, CONSTRAINT TABLE1\_PK PRIMARY KEY

(

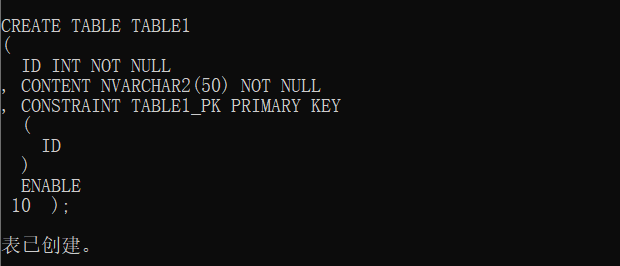
ID

)

ENABLE

);

**创建完成后表结构如下**



1. **存储过程和函数创建**

**①创建视图计算每个用户购车中单个商品的总价**

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）。

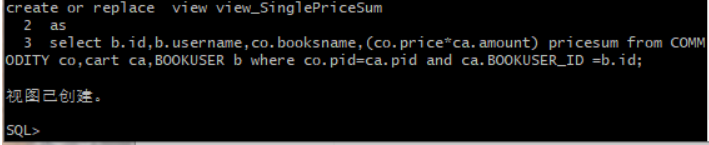
还有一种视图：物化视图（MATERIALIZED VIEW ），也称实体化视图，快照 （8i 以前的说法） ，它是含有数据的，占用存储空间。

create or replace view view\_SinglePriceSum

as

select b.id,b.username,co.booksname,(co.price\*ca.amount) pricesum from COMMODITY co,cart ca,BOOKUSER b where co.pid=ca.pid and ca.BOOKUSER\_ID =b.id;

select \* from view\_SinglePriceSum;



**用户user\_cy\_1空间不足，修改user\_cy\_1空间大小**

ALTER USER user\_cy\_1 QUOTA 90M ON space\_cy\_1;

**②插入用户、商品、购物车数据（50000条数据）**

declare

id number(38,0);

username varchar2(50);

phone varchar2(20);

address varchar2(30);

REGISTRATIONDATE date;

booksname varchar2(50);

price number(5,2);

num number(38,0);

amount number(38,0);

begin

for i in 1..50000

loop

if i mod 2 =0 then

REGISTRATIONDATE:=to\_date('2018-5-6','yyyy-mm-dd')+(i mod 60);

else

REGISTRATIONDATE:=to\_date('2019-5-6','yyyy-mm-dd')+(i mod 60);

end if;

--插入用户

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

username := 'aa'|| 'aa';

username := 'wang' || i;

phone := '131785693' || i;

booksname := '十万个为什么' || i;

address :='云南'|| '昆明';

price :=(dbms\_random.value() \* 100);

num :=(i mod 5);

insert /\*+append\*/ into bookuser (id,password,username,phone,address,REGISTRATIONDATE,cart\_id)

values (id,username,username,phone,address,REGISTRATIONDATE,id);

--插入货品

insert into commodity(id,pid,booksname,price,describe,num,admin\_id)

values (id,id,booksname,price,'good',num,1);

--插入购物车

amount :=(id mod 3 ) + 1;

insert into cart(id,amount,pid,bookuser\_id)

values (id,amount,id,id);

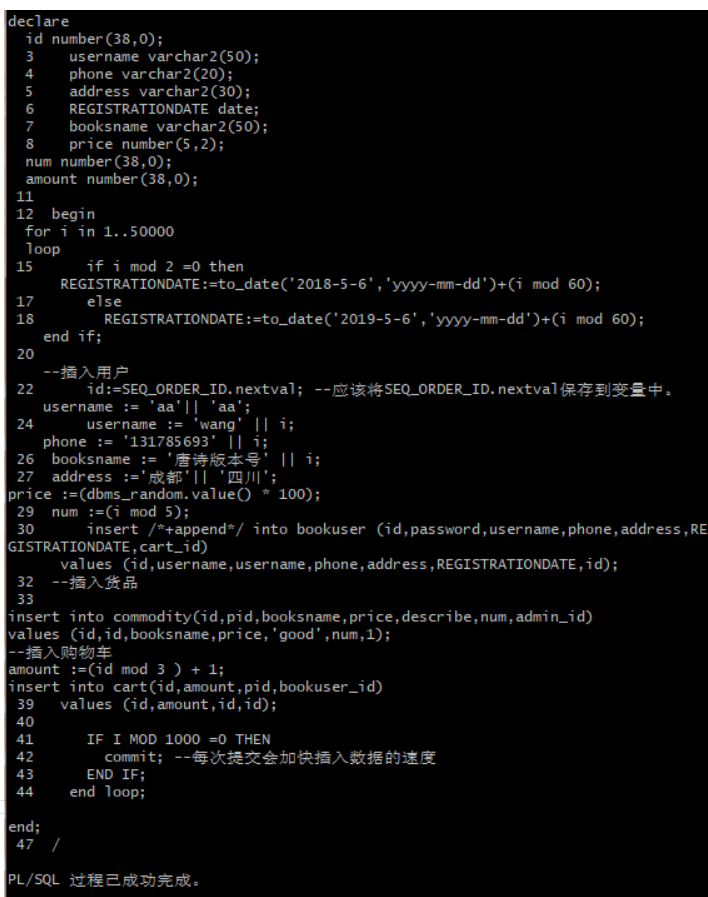
IF I MOD 1000 =0 THEN

commit; --每次提交会加快插入数据的速度

END IF;

end loop;

end;



**③创建程序包、存储过程、函数执行分析计划**

**（1）创建程序包**

函数getcartsumprice计算每个用户的购物车商品总金额

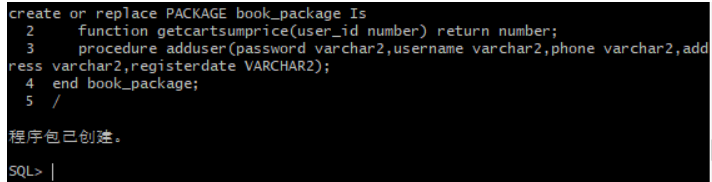
存储过程adduser插入用户信息

create or replace PACKAGE book\_package Is

function getcartsumprice(user\_id number) return number;

procedure adduser(password varchar2,username varchar2,phone varchar2,address varchar2,registerdate VARCHAR2);

end book\_package;

****

**（2）创建函数、存储过程**

create or replace PACKAGE body book\_package Is

function getcartsumprice(user\_id number) return number as

begin

declare cart\_sum number;

query\_sql varchar2(200);

begin

query\_sql:='select sum(pricesum) from view\_SinglePriceSum where ID=' || user\_id;

execute immediate query\_sql into cart\_sum;

return cart\_sum;

end;

end getcartsumprice;

procedure addUser(password varchar2,username varchar2,phone varchar2,address varchar2,registerdate varchar2) as

begin

declare maxId number;

begin

select max(id) into maxId from bookuser;

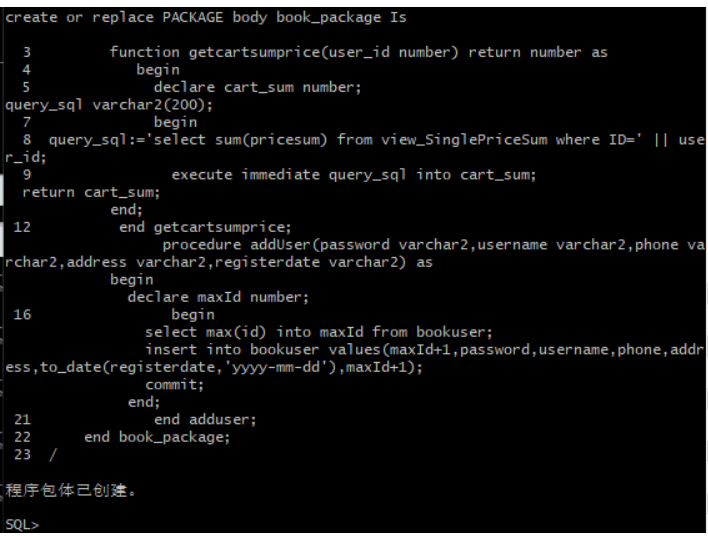
insert into bookuser values(maxId+1,password,username,phone,address,to\_date(registerdate,'yyyy-mm-dd'),maxId+1);

commit;

end;

end adduser;

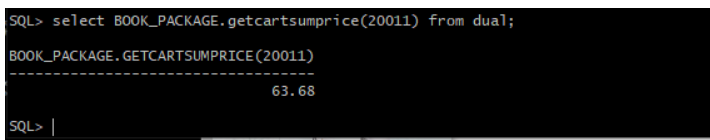
end book\_package;



**（3）存储过程、函数执行分析**

使用自定义函数getcartsumprice（）查询id号为20011的用户购物车商品总价

select BOOK\_PACKAGE.getcartsumprice(20011) from dual;



使用存储过程adduser插入用户数据

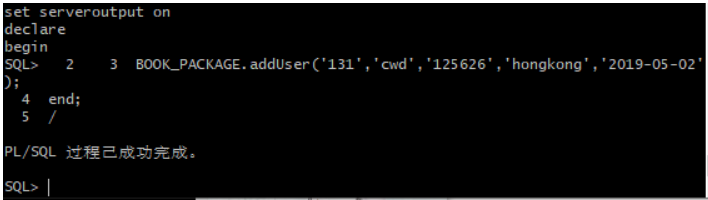
set serveroutput on

declare

begin

BOOK\_PACKAGE.addUser('131','cwd','125626','hongkong','2019-05-02');

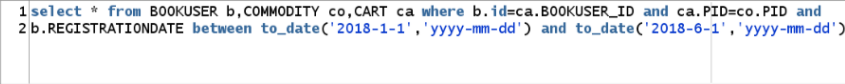
end;



**（4）执行计划分析**

select \* from BOOKUSER b,COMMODITY co,CART ca where b.id=ca.BOOKUSER\_ID and ca.PID=co.PID and

b.REGISTRATIONDATE between to\_date('2018-1-1','yyyy-mm-dd') and to\_date('2018-6-1','yyyy-mm-dd');



**表空间使用状况**

SELECT a.tablespace\_name "表空间名",

total "表空间大小",

free "表空间剩余大小",

(total - free) "表空间使用大小",

total / (1024 \* 1024 \* 1024) "表空间大小(G)",

free / (1024 \* 1024 \* 1024) "表空间剩余大小(G)",

(total - free) / (1024 \* 1024 \* 1024) "表空间使用大小(G)",

round((total - free) / total, 4) \* 100 "使用率 %"

FROM (SELECT tablespace\_name, SUM(bytes) free

FROM dba\_free\_space

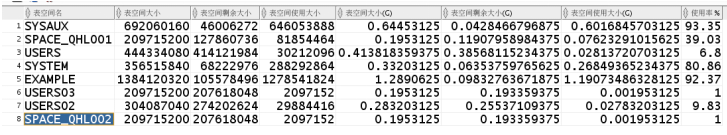
GROUP BY tablespace\_name) a,

(SELECT tablespace\_name, SUM(bytes) total

FROM dba\_data\_files

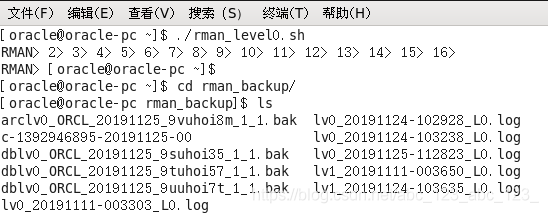
GROUP BY tablespace\_name) b

WHERE a.tablespace\_name = b.tablespace\_name

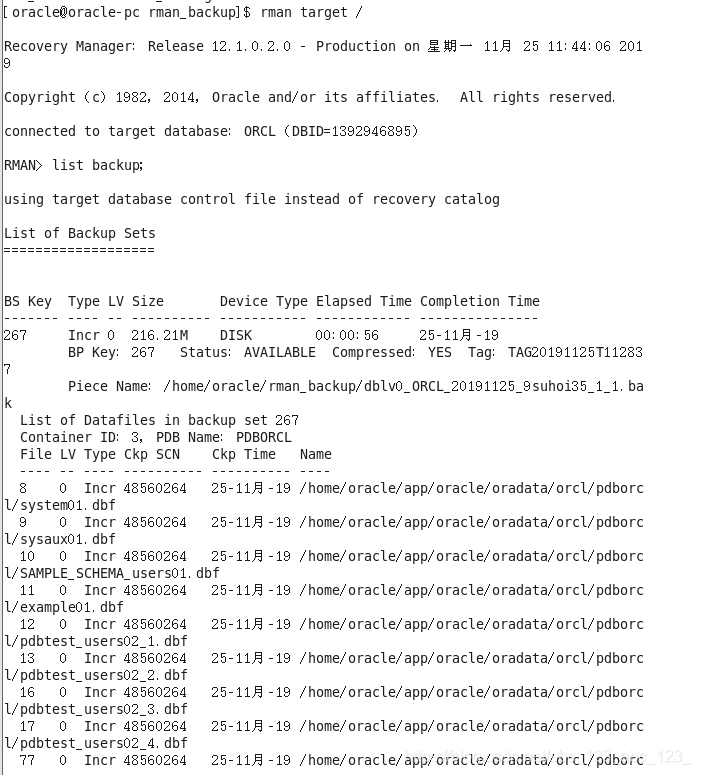


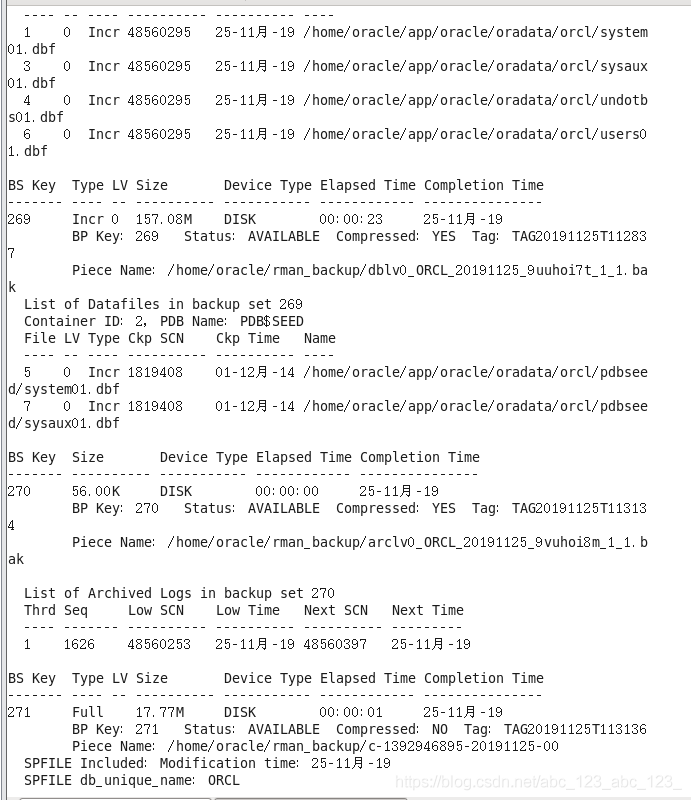
**四．备份恢复**

**①备份./rman\_level0.sh**



**②查看备份内容**

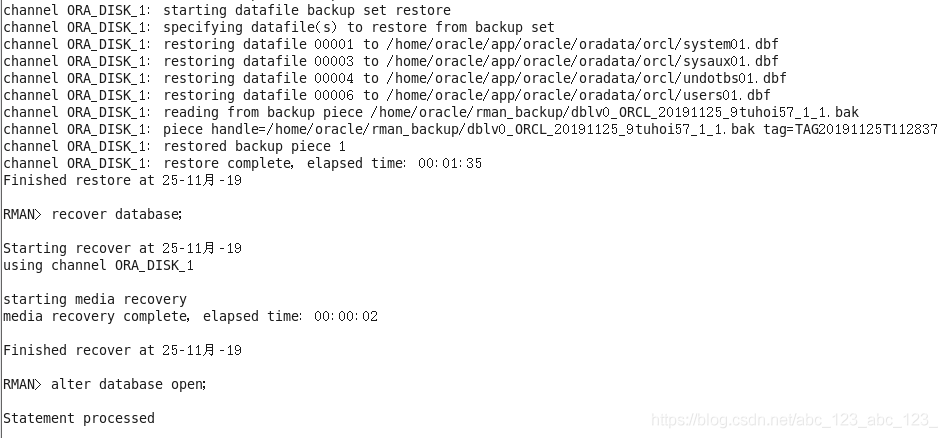
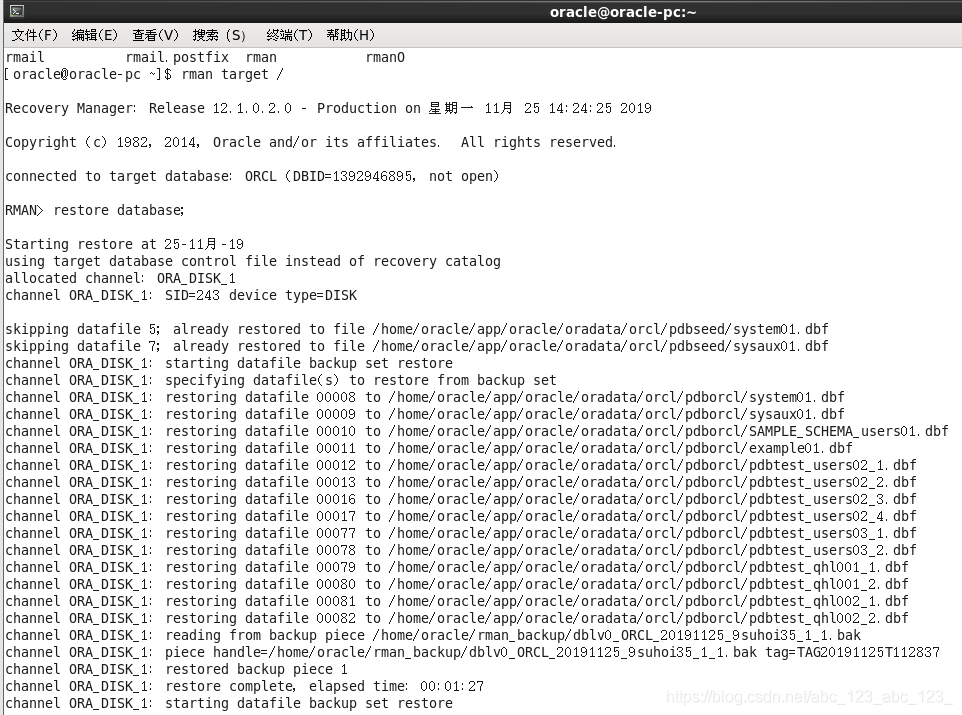




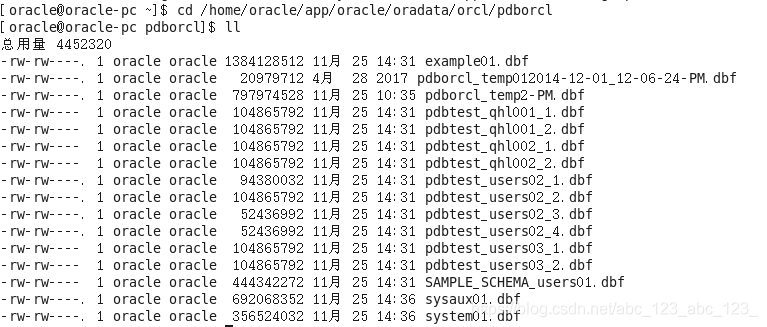
**③删除数据**

E:\GitWorkSpace\Oracle\test06\pic\15.png

**④恢复备份**



**⑤数据已恢复**



### **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。