

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的秒杀系统的数据库设计 | | |
| 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 | 2017级 |
| 学生姓名 | 邓聪 | 学号 | 201710424105 |
| 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 | 10 |  |
| PL/SQL设计 | 存储过程和函数设计正确 | 25 |  |
| 备份方案 | 备份方案设计正确 | 25 |  |
| 容灾方案 | DataGuard设计正确 | 10 |  |
| **得分合计** | | |  |

2019 年 11 月 18 日

目录

[秒杀系统数据设计 4](#_Toc25735485)

[一、引言 4](#_Toc25735486)

[二、数据库设计 4](#_Toc25735487)

[实体属性： 4](#_Toc25735488)

[关联表： 7](#_Toc25735489)

[三、表和数据添加 8](#_Toc25735490)

[一、建表和插入数据 8](#_Toc25735491)

[二、数据库表导入相应数据 11](#_Toc25735492)

[四、ORACLE中相关配置 14](#_Toc25735493)

[五、所遇到的问题和如何解决的 19](#_Toc25735494)

### 秒杀系统数据设计

## 一、引言

无论是双十一还是12306的购票系统，秒杀场景已经随处可见了。简单来说就是在同一时刻大量的数据情求与同一件商品并完成交易，从架构的角度来看，秒杀系统本质是一个高可用，高一致性高性能的三高系统 ，数据库的设计也是在抗并发的解决措施里面，本数据库设计则是应用于此类秒杀系统中。

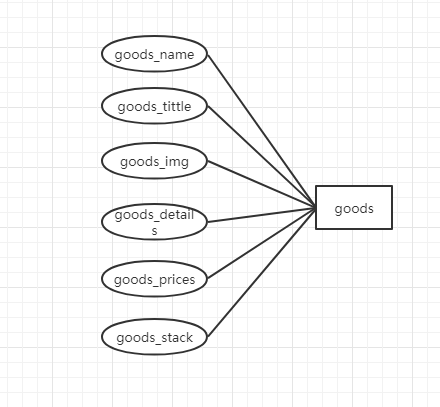
## 二、数据库设计

数据库中有8张表，分别是goods（商品表），miaosha\_goods（秒杀商品表），miaosha\_message（秒杀信息表），miaosha\_message\_user（秒杀信息用户表），miaosha\_order(订单表)，miaosha\_user(秒杀用户)，order\_info(订单详表).

### 实体属性：

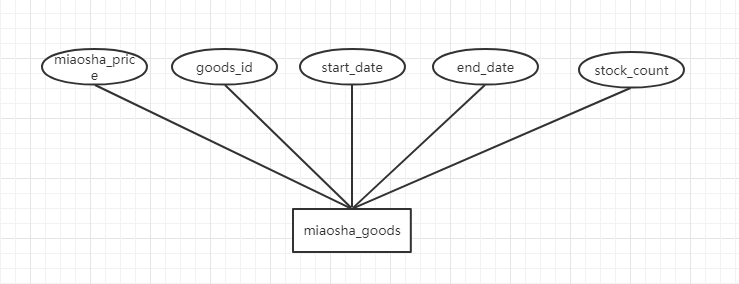
#### 商品实体：

属性的意思分别为goods\_name（商品名称），goods\_title（商品标题），goods\_img（商品图片），goods\_details（商品的详细描述），goods\_prices（商品价格），goods\_stack（商品的库存）。



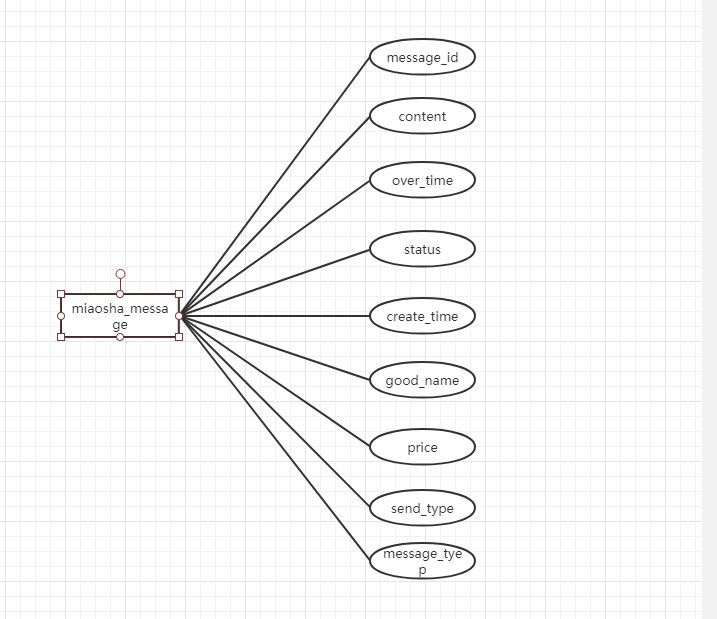
#### 秒杀商品实体：

Miaosha\_price（秒杀价格），goods\_id（商品id），start\_date（开始秒杀的时间），end\_date（秒杀结束的时间），stock\_count（库存数量）。



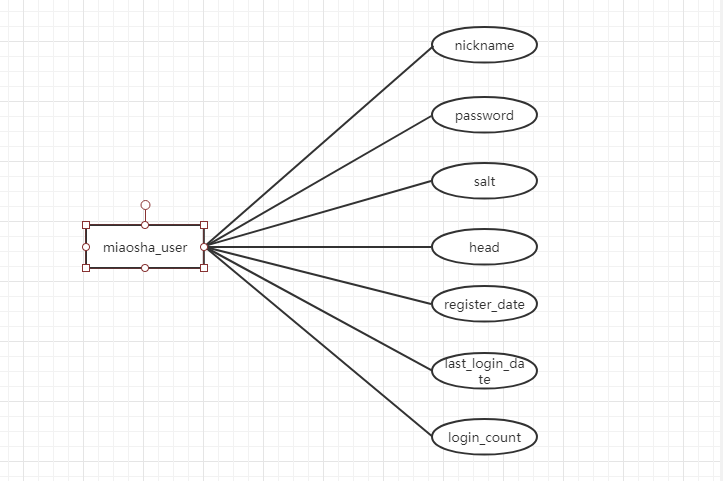
#### 秒杀信息实体；

message\_id(信息id), content(信息内容), over\_time(结束时间), status(状态), create\_time(创建时间), good\_name(商品名称), price(价格), send\_type(发送类型), message\_type(0 秒杀消息 1 购买消息 2 推送消息).



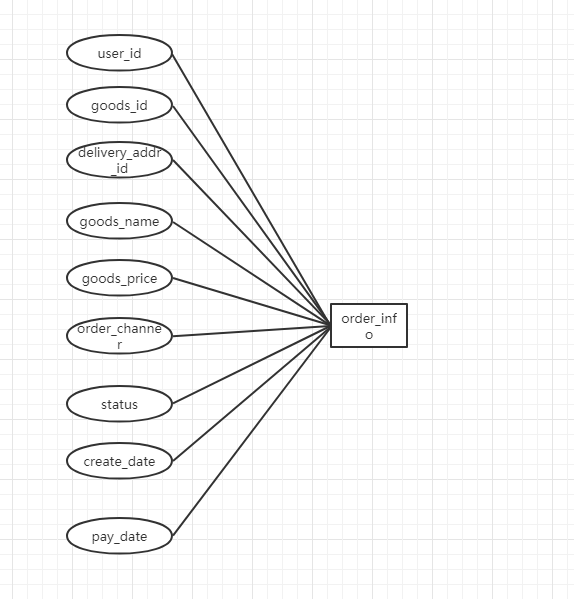
#### 秒杀用户实体；

Nickname(昵称), password(密码), salt， head(头像), register\_date(注册日期), last\_login\_date(最后登录日期), login\_count(登录次数).



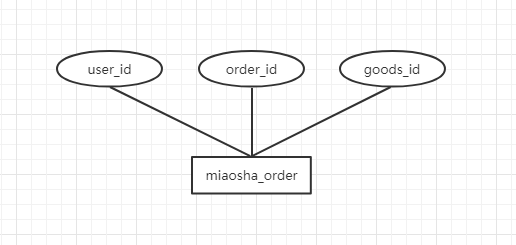
#### 订单详情实体：

User\_id(用户id), goods\_id(商品id), delivery\_addr\_id(邮寄方式), goods\_name(商品名称), goods\_price(商品价格), order\_chnner(1pc，2android，3ios), status(订单状态), create\_date(创建日期), pay\_date(付款日期).

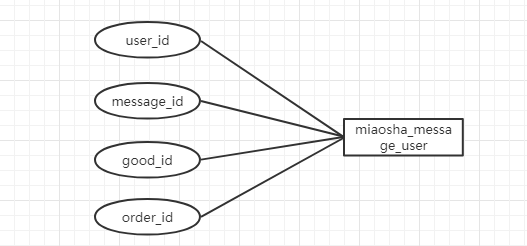


### 关联表：

miaosha\_order（秒杀-订单-商品），是将用户，商品、订单三张表关联起来的数据表。

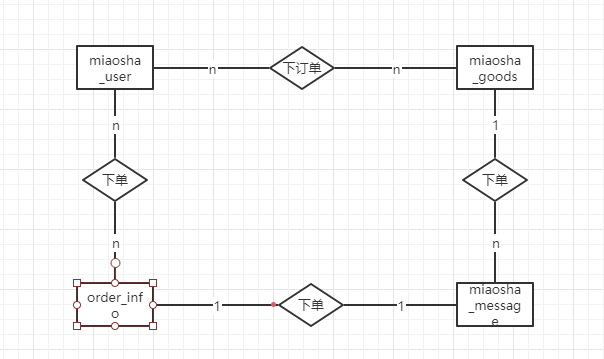


Miaosha\_message\_user（用户-信息-商品-订单），是将用户、商品、信息、订单关联起来的表。



E-R图

简易的e-r图，如下：



## 三、表和数据添加

### 一、建表和插入数据

如下是本次数据库的建表sql，使用的是pl/sql语言写的一个文件。

在建表之前应该判断数据库中是否有该表的存在，如果有删除，如果没有，则执行建表语句。

这里使用的是查找该表中的数据条数，来判断是是否有表，然后执行drop table 来删除表。Declare表示申明，begin表示执行开始，需要在结尾加上end；/ 表示执行以上所有代码。

declare

num number;

begin

select count(1) into num from user\_tables where TABLE\_NAME = 'GOODS';

if num=1 then

execute immediate 'drop table GOODS cascade constraints PURGE';

end if;

select count(1) into num from user\_tables where TABLE\_NAME = 'MIAOSHA\_GOODS';

if num=1 then

execute immediate 'drop table MIAOSHA\_GOODS cascade constraints PURGE';

end if;

select count(1) into num from user\_tables where TABLE\_NAME = 'MIAOSHA\_MESSAGE';

if num=1 then

execute immediate 'drop table MIAOSHA\_MESSAGE cascade constraints PURGE';

end if;

select count(1) into num from user\_tables where TABLE\_NAME = 'MIAOSHA\_MESSAGE\_USER';

if num=1 then

execute immediate 'drop table MIAOSHA\_MESSAGE\_USER cascade constraints PURGE';

end if;

select count(1) into num from user\_tables where TABLE\_NAME = 'MIAOSHA\_ORDER';

if num=1 then

execute immediate 'drop table MIAOSHA\_ORDER cascade constraints PURGE';

end if;

select count(1) into num from user\_tables where TABLE\_NAME = 'MIAOSHA\_USER';

if num=1 then

execute immediate 'drop table MIAOSHA\_USER cascade constraints PURGE';

end if;

select count(1) into num from user\_tables where TABLE\_NAME = 'ORDER\_INFOR';

if num=1 then

execute immediate 'drop table ORDER\_INFOR cascade constraints PURGE';

end if;

select count(1) into num from user\_tables where TABLE\_NAME = 'USERS';

if num=1 then

execute immediate 'drop table USERS cascade constraints PURGE';

end if;

end;

/

以下是建表语句，根据数据字段的设计，在数据库中设计数据库表，这里不赘述。

--创建goods 表

CREATE TABLE GOODS (

ID NUMBER(6, 0) NOT NULL ,

GOODS\_NAME VARCHAR2(16 BYTE) DEFAULT NULL,

GOODS\_TITLE VARCHAR2(64 BYTE) DEFAULT NULL,

GOODS\_IMG VARCHAR2(64 BYTE) DEFAULT NULL,

GOODS\_DETAILS VARCHAR2(100 BYTE) ,

GOODS\_PRICE NUMBER(10,2)DEFAULT '0.00',

GOODS\_STAOCK NUMBER(11,0)

-- PRIMARY KEY (`id`)

) ;

-- 创建miaosha\_goods表

CREATE TABLE MIAOSHA\_GOODS(

ID NUMBER(6,0) NOT NULL,

GOODS\_ID NUMBER(20, 0) DEFAULT NULL,

MIAOSHA\_PRICE NUMBER(10,2) DEFAULT '0.00',

STOCK\_COUNT NUMBER(11) DEFAULT NULL,

START\_DATE DATE DEFAULT NULL,

END\_DATE DATE DEFAULT NULL

-- PRIMARY KEY (`id`)

);

--创建miaosha\_message表

CREATE TABLE MIAOSHA\_MESSAGE(

ID NUMBER(20, 0) NOT NULL,

MESSAGE\_ID NUMBER(20,0) NOT NULL,

CONTENT VARCHAR2(100) ,

CREATE\_TIME DATE DEFAULT NULL ,

STATUS NUMBER(1,0) NOT NULL ,

OVER\_TIME DATE DEFAULT NULL,

MESSAGE\_TYPE NUMBER(1, 0) DEFAULT '3' ,

SEND\_TYPE NUMBER(1) DEFAULT '3' ,

GOOD\_NAME VARCHAR2(50) DEFAULT '' ,

PRICE NUMBER(10,2) DEFAULT '0.00'

-- PRIMARY KEY (`id`)

);

--创建miaosha\_message\_user

CREATE TABLE MIAOSHA\_MESSAGE\_USER(

ID NUMBER(20) NOT NULL,

USER\_ID NUMBER(20) NOT NULL,

MESSAGE\_ID NUMBER(38) NOT NULL,

GOODS\_ID NUMBER(20) DEFAULT NULL,

ORDER\_ID NUMBER(20) DEFAULT NULL

);

--创建表miaosha\_order表

CREATE TABLE MIAOSHA\_ORDER(

ID NUMBER(20) NOT NULL ,

USER\_ID NUMBER(20) DEFAULT NULL ,

ORDER\_ID NUMBER(20) DEFAULT NULL ,

GOODS\_ID NUMBER(20) DEFAULT NULL

);

--创建表miaosha\_user

CREATE TABLE MIAOSHA\_USER(

ID NUMBER(20) NOT NULL,

USER\_ID NUMBER(20) DEFAULT NULL,

NICKNAME VARCHAR2(255) NOT NULL,

PASSWORD VARCHAR2(32) DEFAULT NULL ,

SALT VARCHAR2(10) DEFAULT NULL,

HEAD VARCHAR2(128) DEFAULT NULL ,

REGISTER\_DATE DATE DEFAULT NULL ,

LAST\_LOGIN\_DATE DATE DEFAULT NULL ,

LOGIN\_COUNT NUMBER(11) DEFAULT '0'

);

-- 创建表order\_info

CREATE TABLE ORDER\_INFOR (

ID NUMBER(20) NOT NULL,

ORDER\_ID NUMBER(20) DEFAULT NULL,

USER\_ID NUMBER(20) DEFAULT NULL,

GOODS\_ID NUMBER(20) DEFAULT NULL ,

DELIVERY\_ADDR\_ID NUMBER(20) DEFAULT NULL,

GOODS\_NAME VARCHAR2(16) DEFAULT NULL ,

GOODS\_COUNT NUMBER(11) DEFAULT '0',

GOODS\_PRICE NUMBER(10,2) DEFAULT '0.00',

ORDER\_CHANNER NUMBER(4) DEFAULT '0',

STATUS NUMBER(4) DEFAULT '0',

CREATE\_DATE DATE DEFAULT NULL,

PAY\_DATE DATE DEFAULT NULL

) ;

### 二、数据库表导入相应数据

使用pl/sql语句来添加数据。

#### --1、向goods表中添加数据

这里定义了6个数组，数据库表中的每个字段随机从每个数组中选取数据，构成一个记录，插入到数据库中相应表中，数据条数为10000条。

set SERVEROUTPUT ON;

create or replace function RANDOM

return number

is

a number ;

begin

select round(dbms\_random.value(1,5)) rnum

into a

from dual;

return a ;

end;

/

DECLARE

type goods\_name is varray(5) of varchar2(20);

type goods\_details is varray(5) of VARCHAR2(100);

type goods\_img is varray(5) of VARCHAR2(20);

type goods\_info is varray(5) of VARCHAR2(50);

type goods\_price is varray(5) of VARCHAR2(20);

type goods\_have is varray(5) of VARCHAR2(20);

indexRandom NUMBER;

goods\_name\_list goods\_name:=goods\_name('iphoneX','华为Meta9','小米6','一加','vivo');

goods\_details\_list goods\_details:=goods\_details('64GB 银色 移动联通电信4G手机','4GB+32GB版 移动联通电信4G手机 双卡双待','64GB 银色 移动联通电信3G手机','月光银 移动联通电信4G手机','玫瑰金 64g 晓龙a382');

goods\_img\_list goods\_img:=goods\_img('/img/iphonex.png','/img/meta10.png','/img/iphone8.png','/img/mi6.png','/img/mi9.png');

goods\_info\_list goods\_info:=goods\_info('移动联通电信4G手机','4GB+32GB版','月光银','玫瑰金','64GB 银色');

goods\_price\_list goods\_price:=goods\_price('8765.00','3212.00','5589.00','3212.00','7212.00');

goods\_have\_list goods\_have:=goods\_have('8765','-1','558','3212','7212');

BEGIN

dbms\_output.put\_line(indexRandom);

DBMS\_OUTPUT.PUT\_LINE(goods\_name\_list(5));

for i in 1..10000

loop

indexRandom:=RANDOM();

INSERT INTO GOODS VALUES (i, goods\_name\_list(indexRandom), goods\_details\_list(indexRandom), goods\_img\_list(indexRandom), goods\_info\_list(indexRandom), goods\_price\_list(indexRandom), goods\_have\_list(indexRandom));

end loop;

END;

/

#### --2、向miaosha\_goods中添加数据

使用for loop语句构造2到2000 的数据，然后根据i值的不同，其后的数据相同，数据条数为2000条，插入到数据库中。

declare

result number;

begin

for i in 2..2000

loop

result:=i mod 3;

if result =0 then

INSERT INTO MIAOSHA\_GOODS VALUES (i, i, '0.01', '9', ('04-12月-17'), ('04-12月-17'));

DBMS\_OUTPUT.PUT\_LINE(i);

elsif result = 1 then

INSERT INTO MIAOSHA\_GOODS VALUES (i, i, '0.25', '12', ('08-11月-18'), ('08-11月-18'));

DBMS\_OUTPUT.PUT\_LINE(i);

else

INSERT INTO MIAOSHA\_GOODS VALUES (i, i, '0.05', '8', ('12-11月-17'), ('12-11月-19'));

DBMS\_OUTPUT.PUT\_LINE(i);

end if;

exit when i=2000;

end loop;

end;

/

#### --3、向miaosha\_message 中添加数据

如上面的导入数据方式，使用i值的不同，构造不同记录。

declare

begin

for i in 2..5000

loop

INSERT INTO MIAOSHA\_MESSAGE VALUES (i,'533324506110885888', '尊敬的用户你好，你已经成功注册！', null, '0', null, null, '0', null, null);

end loop;

end;

/

-- miaosha\_user

declare

id number;

user\_id number;

nickname number;

begin

id:=18912341247;

nickname:=18612766444;

user\_id:=1;

loop

id:=id+1;

user\_id:=user\_id+1;

nickname:=nickname+1;

INSERT INTO MIAOSHA\_USER VALUES (id, user\_id, nickname, 'b7797cce01b4b131b433b6acf4add449', '1a2b3c4d', null, '11-1月-19', null, '0');

exit when id=18912344246;

end loop;

end;

/

#### --4、向order\_info中添加数据

导入order\_info使用的是循环loop….end，创建变量order\_id,user\_id，goods\_id，并赋以初值，然后在执行完一条insert之后，是这些变量的值进行相应的改变，达到数据导入的目的。

DECLARE

-- INSERT INTO `order\_info` VALUES ('1564', '18912341234', '3', null, 'iphone8', '1', '0.01', '1', '0', '2017-12-16 16:35:20', null);

ORDER\_ID NUMBER;

USER\_ID NUMBER;

GOODS\_ID NUMBER;

ID NUMBER:=1;

BEGIN

--20000

ORDER\_ID:=1564;

USER\_ID:=1;

GOODS\_ID:=1;

LOOP

INSERT INTO ORDER\_INFOR VALUES(ID, ORDER\_ID, USER\_ID, GOODS\_ID,null, 'iphone8', '1', '0.01', '1', '0', '16-12月-17', null);

ORDER\_ID:=ORDER\_ID+1;

USER\_ID:=USER\_ID+1;

GOODS\_ID:=GOODS\_ID+1;

EXIT ???

END LOOP;

END;

/

## 四、ORACLE中相关配置

首先就是新建pdb 的操作，oracle没有办法对cdb进行操作，只能操作pdb，所以在oracle中的开始，我就需要新建一个pdb数据库，以上的相关操作，都是建立在这次之后的操作，这里新建一个salespdb的pdb数据库。

大致解释以下语句的含义：

Create pluggable database 就是新建一个pdb的语句，其中salespdb是数据库的名称，然后就是用户名和密码，使用的tablespace的大小，默认的存储文件地址。

CREATE PLUGGABLE DATABASE salespdb ADMIN USER sale5deng IDENTIFIED BY sale5deng STORAGE (MAXSIZE 2G) DEFAULT TABLESPACE sales DATAFILE '/database/oracle/oracle/oradata/orcl/salespdb/sales01.dbf' SIZE 250M AUTOEXTEND ON PATH\_PREFIX = '/database/oracle/oracle/oradata/orcl/salespdb/' FILE\_NAME\_CONVERT = ('/database/oracle/oracle/oradata/orcl/pdbseed/', '/database/oracle/oracle/oradata/orcl/salespdb/');

#### 一、表设计

创建表空间的过程，创建了三个表空间，分别叫做sales，sales02，sales03，大小最大为50M，数据文件存放在/database/oracle/oracle/oradata/orcl/orclpdb/目录下面。

CREATE TABLESPACE SALES

DATAFILE '/database/oracle/oracle/oradata/orcl/salespdb/sales.dbf'

SIZE 100M AUTOEXTEND ON NEXT 50M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

CREATE TABLESPACE USERS01

DATAFILE '/database/oracle/oracle/oradata/orcl/ salespdb / sales02.dbf'

SIZE 100M AUTOEXTEND ON NEXT 50M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

CREATE TABLESPACE USERS02

DATAFILE '/database/oracle/oracle/oradata/orcl/ salespdb / sales03.dbf'

SIZE 100M AUTOEXTEND ON NEXT 50M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

#### 二、用户管理

##### 创建用户

这里创建了两个用户，分别叫做sale5deng和buyer5deng

SYSTEM@192.168.44.183:1521/salespdb>create role sales5deng identified sales5deng;

角色已创建。

SYSTEM@192.168.44.183:1521/salespdb>create role buyer5deng identified buyer5deng;

角色已创建。

##### 权限配置

给刚创建的两个用户添加connect，resource，create view的权限

SYSTEM@192.168.44.183:1521/salespdb>grant connect, resource, CREATE VIEW TO sales5deng;

授权成功。

SYSTEM@192.168.44.183:1521/salespdb>grant connect, resource, CREATE VIEW TO buyer5deng;

授权成功。

##### 表空间分配

数据库中有三个刚才创建的表空间，分别为sales，sales02，sales03.

SYSTEM@192.168.44.229:1521/salespdb>select tablespace\_name from user\_tablespaces;

TABLESPACE\_NAME

------------------------------

SYSTEM

SYSAUX

UNDOTBS1

TEMP

SALES

SALES02

SALES03

已选择 7 行。

#### 三、PL/SQL设计

查找miaosha\_order 表中的数据，使用miaosha\_user的user\_id，使用存储过程queryUser传入user\_id，从miasha\_order表中查出相应的数据记录，然后取出goods，使用goods\_id，在miaosha\_goods中进行查询，查询出相应的记录

set serveroutput on;

create or replace procedure queryUser

(

u\_user\_id in MIAOSHA\_USER.user\_id%type,

u\_goods\_id out miaosha\_goods.goods\_id%type

)

as

begin

select goods\_id into u\_goods\_id from order\_infor where order\_id = (select order\_id from (select order\_id from miaosha\_order where rownum=1 and miaosha\_order.user\_id=u\_user\_id));

dbms\_output.put\_line(u\_goods\_id);

-- select \* from order\_infor where order\_infor.order\_id=u\_order\_id;

exception

when no\_data\_found then

dbms\_output.put\_line('error');

when others then

dbms\_output.put\_line('one error');

end queryUser;

/

--调用

declare

v1 miaosha\_goods.goods\_id%TYPE;

BEGIN

queryUser('178', v1);

dbms\_output.put\_line('name');

end;

输出结果如下。

Procedure QUERYUSER 已编译

423

name

PL/SQL 过程已成功完成。

#### 四、备份设计

**备份**

从虚拟机中拷贝出脚本文件rman\_leve10.sh(全备份)，rman\_level1.sh(增量备份)，查看脚本内容

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

#rman\_level0.sh

#!/bin/sh

export NLS\_LANG='SIMPLIFIED CHINESE\_CHINA.AL32UTF8'

export ORACLE\_HOME=/home/oracle/app/oracle/product/12.1.0/dbhome\_1

export ORACLE\_SID=orcl

export PATH=$ORACLE\_HOME/bin:$PATH

rman target / nocatalog msglog=/home/oracle/rman\_backup/lv0\_`date +%Y%m%d-%H%M%S`\_L0.log << EOF

run{

configure retention policy to redundancy 1;

configure controlfile autobackup on;

configure controlfile autobackup format for device type disk to '/home/oracle/rman\_backup/%F';

configure default device type to disk;

crosscheck backup;

crosscheck archivelog all;

allocate channel c1 device type disk;

backup as compressed backupset incremental level 0 database format '/home/oracle/rman\_backup/dblv0\_%d\_%T\_%U.bak'

plus archivelog format '/home/oracle/rman\_backup/arclv0\_%d\_%T\_%U.bak';

report obsolete;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

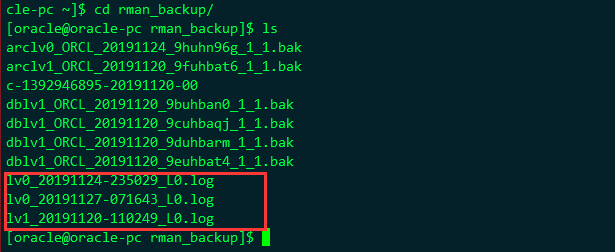
release channel c1;

}

EOF

exit

在用户oracle下运行脚本rman\_level10.sh，



\*.log 是日志文件

Dblv0\*.bak 是数据库的备份文件

arclv0\*.bak是归档日期的备份文件

c-1392946895-20191120-01是控制文件和参数的备份。

**修改数据**

[oracle@oracle-pc ~]$ sqlplus study/123@pdborcl

SQL> create table t2 (id number,name varchar2(50));

Table created.

SQL> insert into t2 values(1,'zhang');

1 row created.

SQL> commit;

Commit complete.

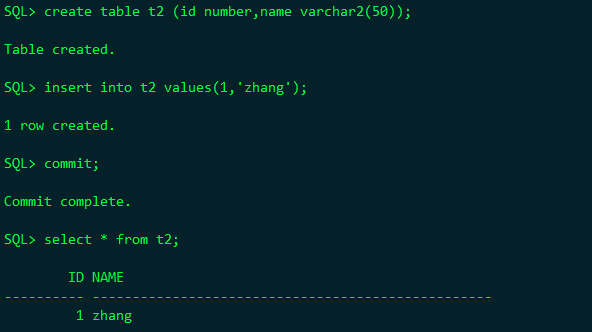
SQL> select \* from t2;

ID NAME

---------- --------------------------------------------------

1 zhang

SQL> exit



**删除数据**

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

挂载数据库到mount状态

SQL> shutdown immediate

ORA-01116: 打开数据库文件 10 时出错

ORA-01110: 数据文件 10: '/home/oracle/app/oracle/oradata/orcl/pdborcl/SAMPLE\_SCHEMA\_users01.dbf'

ORA-27041: 无法打开文件

Linux-x86\_64 Error: 2: No such file or directory

Additional information: 3

SQL> shutdown abort

ORACLE instance shut down.

SQL> startup mount

ORACLE instance started.

Total System Global Area 1577058304 bytes

Fixed Size 2924832 bytes

Variable Size 738201312 bytes

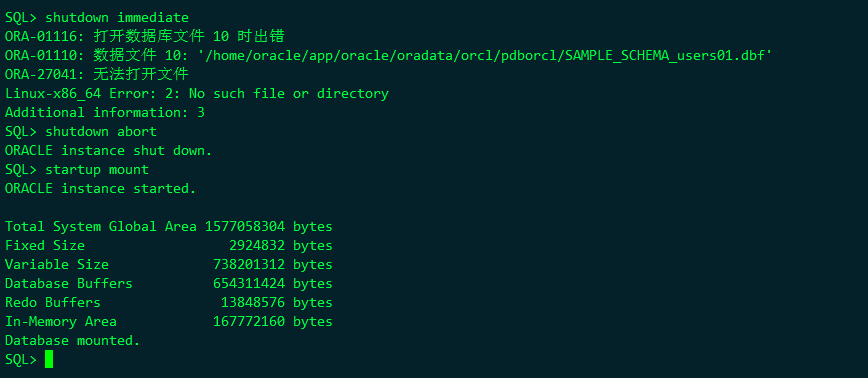
Database Buffers 654311424 bytes

Redo Buffers 13848576 bytes

In-Memory Area 167772160 bytes

Database mounted.

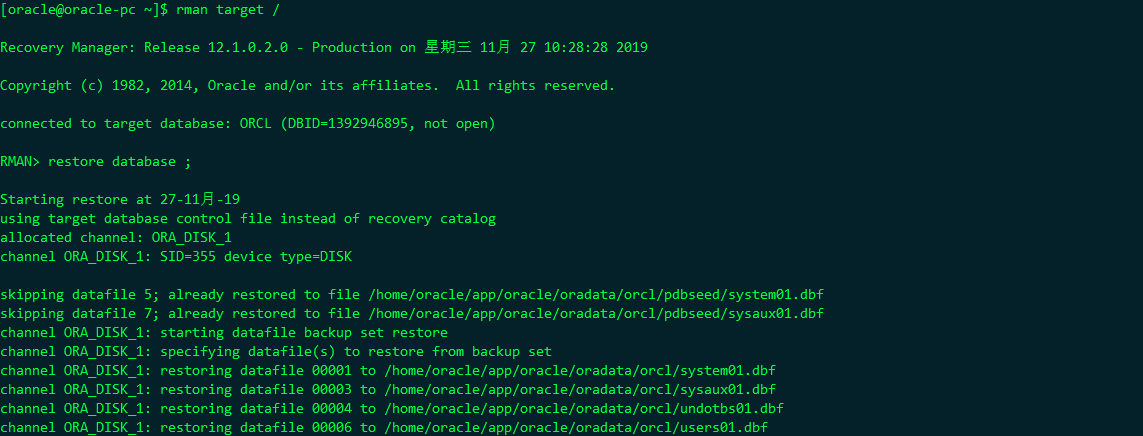
SQL>



恢复数据

[oracle@oracle-pc ~]$ rman target /

RMAN> restore database ;



## 五、所遇到的问题和如何解决的

①、lsnrctl status 无监听

使用lsnrctl start来开启监听，但是在开启的过程中，监听要去寻找一个叫做listen.ora的文件，当时我将此文件更名为listen.ora.bak，所以在启动的时候没有找到文件，一直没有启动起来。

Lsnrctl的基本操作有server，start，stop。

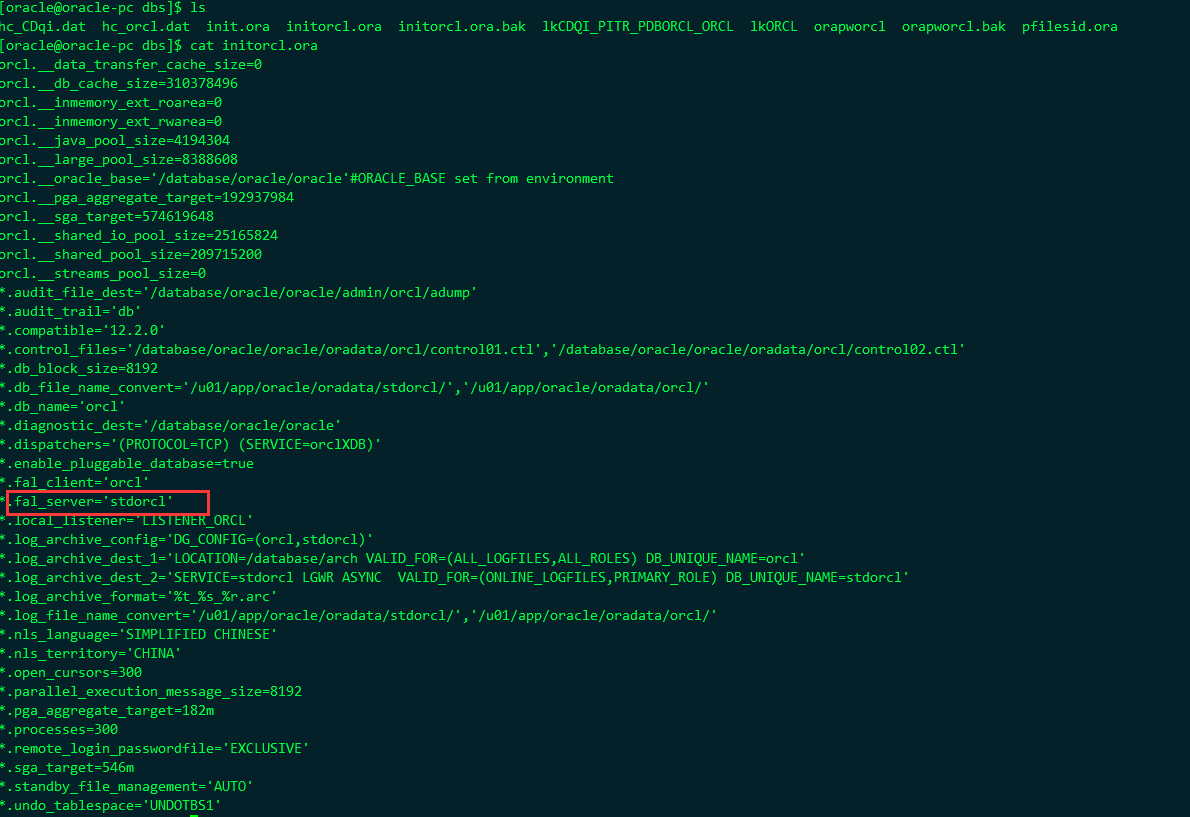
②、ORACEL12C ORA-01033: ORACLE 正在初始化或关闭 进程 ID: 0 会话 ID: 0 序列号: 0

Oracle 中和其他版本的数据的启动关闭都是一样的，但是12c中有一个特殊点，就是在启动的时候需要修改会话到链接的pdb上面。例如：alter session set container=orclpdb.

参考下面地址：<https://www.jianshu.com/p/5571e0413ff4>

③、容灾实验过程 中，无法使用rman登录到指定的数据。

这里是我在填写监听的时候，还是写的是orcl，不是stdorcl，所以使用rman target sys1/123@stdorcl则是无法登录的。



④、SQL Oracle 查询出来的数据取第一条

select \* from (select \* from <table> order by <key>) where rownum=1;

select \* from (select \* from <table> order by <key> desc) where rownum=1;

⑤、使用sys或system用户登录，却没有权限？

Sqlplus “/ as sysdba” 这样登录进去、