

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的废物回收系统的数据库设计 | | |
| 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 | 覃龙 | 学号 | 201810414121 |
| 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** | | |  |

2021 年 6 月 3 日

目录

[废物回收系统数据设计 4](#_Toc74572183)

[一、引言 4](#_Toc74572184)

[二、概念模型设计（E-R图） 4](#_Toc74572185)

[三、数据库表 4](#_Toc74572186)

[四、创建表空间 7](#_Toc74572187)

[五、 创建角色及用户 7](#_Toc74572188)

[六、表和数据添加 8](#_Toc74572189)

[七、数据库表导入相应数据 12](#_Toc74572190)

[八、ORACLE中相关配置 15](#_Toc74572191)

[九、PL/SQL设计 17](#_Toc74572192)

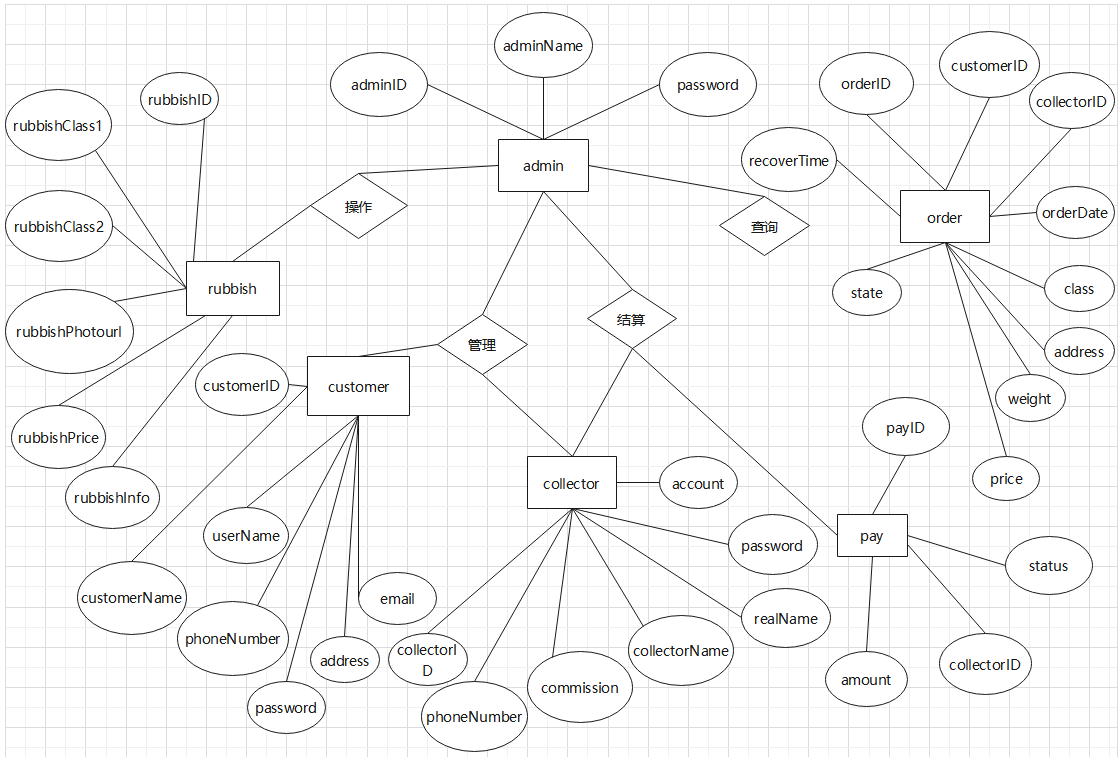
[十、备份设计 18](#_Toc74572193)

### 废物回收系统数据设计

## 一、引言

从资源回收产业链改造的角度去说，互联网取缔了“回收站点”，将零散的回收从业者个人统一起来，使旧物直接从用户家中到达回收基地（大型回收分拣处理企业），缩短了产业链流程以节约成本，包括单人、单点覆盖面的时间成本，运输成本、层层转手产生的二次成本。如果从规模上来说，在完全规模化以后，还能实现城市回收基地的取代，直接与再生产企业建立业务往来，这中间成本几乎为零。另一方面，如果旧物回收工具能够成功进入家庭或社区，成为一道入口，在这个方向上将更加具有想象力。

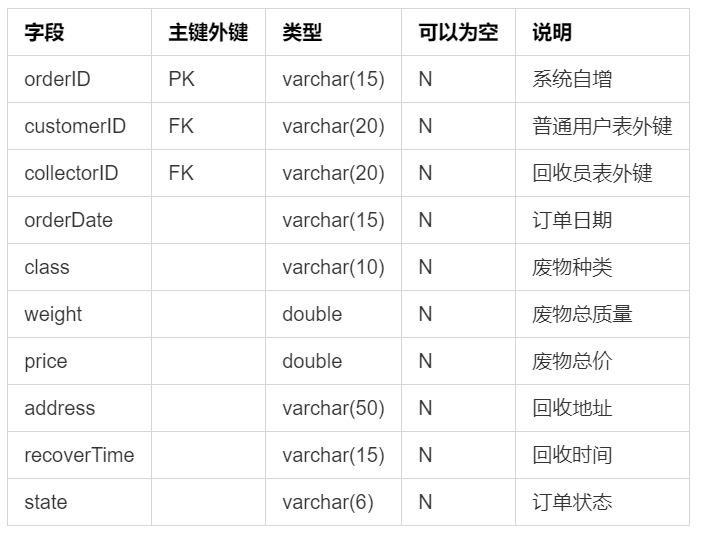
## 二、概念模型设计（E-R图）



图表 1 E-R模型

## 三、数据库表

3.1订单表（orders）：



3.2普通用户表(user)：



3.3回收员表(collector)：



3.4转账信息表(pay)：



3.5管理员admin表:



3.6废物rubbish表:



## 四、创建表空间

#### space\_echo\_no1

Create Tablespace space\_echo\_no1  
datafile  
'/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_echo\_no1\_1.dbf'  
 SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED,  
'/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_echo\_no1\_2.dbf'  
 SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED  
EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

#### space\_echo\_no2

Create Tablespace space\_echo\_no2  
datafile  
'/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_echo\_no2\_1.dbf'  
 SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED,  
'/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_echo\_no2\_2.dbf'  
 SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED  
EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

## 五、 创建角色及用户

用户默认使用表空间 space\_echo\_no1

#### 创建第一个角色和用户

* 创建角色 ql1 将 connect,resource,create view 授权给 ql1
* 创建用户 ql1\_1
* 分配 60M 空间给 ql1\_1 并将角色 ql1 授权给用户 ql1\_1

CREATE ROLE ql1;  
GRANT connect,resource,CREATE VIEW TO ql1;  
CREATE USER ql1\_1 IDENTIFIED BY 123 DEFAULT TABLESPACE space\_echo\_no1 TEMPORARY TABLESPACE temp;  
ALTER USER ql1\_1 QUOTA 60M ON space\_echo\_no1;  
GRANT ql1 TO ql1\_1;

#### 创建第二个角色和用户

* 创建角色 ql2，将 connect,resource 权限给 ql2
* 创建用户 ql1\_2
* 分配 60M 空间给 ql1\_2 并将角色 qhl2 授权给用户 ql1\_2

CREATE ROLE qhl2;  
GRANT connect,resource TO qhl2;  
CREATE USER ql1\_2 IDENTIFIED BY 123 DEFAULT TABLESPACE space\_echo\_no1 TEMPORARY TABLESPACE temp;  
ALTER USER ql1\_2 QUOTA 60M ON space\_echo\_no1;  
GRANT qhl2 TO ql1\_2;

## 六、表和数据添加

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 = 'RUBBISH';

if num=1 then

execute immediate 'drop table RUBBISH cascade constraints PURGE';

end if;

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

if num=1 then

execute immediate 'drop table RUBBISH cascade constraints PURGE';

end if;

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

if num=1 then

execute immediate 'drop table MESSAGE cascade constraints PURGE';

end if;

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

if num=1 then

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

end if;

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

if num=1 then

execute immediate 'drop table ORDER cascade constraints PURGE';

end if;

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

if num=1 then

execute immediate 'drop table 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RUBBISH 表

create table rubbish

(

RUBBISHID INTEGER not null,

RUBBISHCLASS1 VARCHAR2(50) not null,

RUBBISHCLASS2 VARCHAR2(50) not null,

RUBBISHPHOTOURL VARCHAR2(255) not null,

RUBBISHPRICE INTEGER not null,

RUBBISHINFO VARCHAR2(255) not null

);

alter table rubbish

add constraint PK\_RUBBISH primary key (RUBBISHID);

-- 创建user表

create table "user"

(

CUSTOMERID INTEGER not null,

CUSTOMERNAME VARCHAR2(20) not null,

ADDRESS VARCHAR2(50) not null,

EMAIL VARCHAR2(20) not null,

PHONENUMBER VARCHAR2(20) not null,

USERNAME VARCHAR2(20) not null,

PASSWORD VARCHAR2(20) not null

);

alter table "user"

add constraint PK\_USER primary key (CUSTOMERID);

alter table orders

add constraint FK\_ORDERS\_REFERENCE\_USER foreign key (CUSTOMERID)

references "user" (CUSTOMERID);

alter table orders

add constraint FK\_ORDERS\_REFERENCE\_COLLECTO foreign key (COLLECTORID)

references collector (COLLECTORID);

alter table orders

add constraint FK\_ORDERS\_REFERENCE\_RUBBISH foreign key (RUBBISHID)

references rubbish (RUBBISHID);

alter table pay

add constraint FK\_PAY\_REFERENCE\_COLLECTO foreign key (COLLECTORID)

references collector (COLLECTORID);

--创建collector表

create table collector

(

COLLECTORID INTEGER not null,

COLLECTORNAME VARCHAR2(20) not null,

PASSWORD VARCHAR2(20) not null,

REALNAME VARCHAR2(20) not null,

COMMISSION INTEGER not null,

ACCOUNT VARCHAR2(20) not null,

PHONENUMBER VARCHAR2(20) not null

);

alter table collector

add constraint PK\_COLLECTOR primary key (COLLECTORID);

--创建orders表

create table orders

(

ORDERID INTEGER not null,

CUSTOMERID INTEGER,

COLLECTORID INTEGER,

RUBBISHID INTEGER,

ORDERDATE DATE not null,

CLASS VARCHAR2(10) not null,

WEIGHT INTEGER not null,

PRICE INTEGER not null,

ADDRESS VARCHAR2(50) not null,

RECOVERTIME DATE not null,

STATE VARCHAR2(1) not null

);

alter table orders

add constraint PK\_ORDERS primary key (ORDERID);--创建表order表

CREATE TABLE ORDER(

ID NUMBER(20) NOT NULL ,

USER\_ID NUMBER(20) DEFAULT NULL ,

ORDER\_ID NUMBER(20) DEFAULT NULL ,

RUBBISH\_ID NUMBER(20) DEFAULT NULL

);

--创建表user

CREATE TABLE 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pay

create table pay

(

PAYID INTEGER not null,

COLLECTORID INTEGER,

AMOUNT INTEGER not null,

STATUS VARCHAR2(1) not null

);

alter table pay

add constraint PK\_PAY primary key (PAYID);

-- 创建admin表

create table admin

(

ADMINID INTEGER not null,

ADMINNAME VARCHAR2(20) not null,

PASSWORD VARCHAR2(20) not null

);

alter table admin

add constraint PK\_ADMIN primary key (ADMINID);

## 七、数据库表导入相应数据

使用pl/sql语句来添加数据。

#### --1、向RUBBISH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 RUBBISH\_name is varray(5) of varchar2(20);

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

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

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

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

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

indexRandom NUMBER;

RUBBISH\_name\_list RUBBISH\_name:=RUBBISH\_name('金属','塑料',铝制品','瓶子','纸板');

RUBBISH\_details\_list RUBBISH\_details:=RUBBISH\_details('4kg 银色 纸板','4kg 银色 纸板','4kg 银色 金属','4kg 银色 瓶子','4kg 银色 金属');

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

RUBBISH\_info\_list RUBBISH\_info:=RUBBISH\_info('金属','4kg','月光银','玫瑰金',');

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

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

BEGIN

dbms\_output.put\_line(indexRandom);

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

for i in 1..10000

loop

indexRandom:=RANDOM();

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

end loop;

END;

/

#### --2、向RUBBISH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 RUBBISH 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 RUBBISH VALUES (i, i, '0.25', '12', ('08-11月-18'), ('08-11月-18'));

DBMS\_OUTPUT.PUT\_LINE(i);

else

INSERT INTO RUBBISH 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orders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 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;

/

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

exit when id=18912344246;

end loop;

end;

/

#### --4、向orders中添加数据

导入orders使用的是循环loop….end，创建变量order\_id,user\_id，RUBBISH\_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;

RUBBISH\_ID NUMBER;

ID NUMBER:=1;

BEGIN

--20000

ORDER\_ID:=1564;

USER\_ID:=1;

RUBBISH\_ID:=1;

LOOP

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

ORDER\_ID:=ORDER\_ID+1;

USER\_ID:=USER\_ID+1;

RUBBISH\_ID:=RUBBISH\_ID+1;

EXIT ???

END LOOP;

END;

/

## 八、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。

## 九、PL/SQL设计

查找order 表中的数据，使用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RUBBISH，使用RUBBISH\_id，在RUBBISH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 USER.user\_id%type,

u\_RUBBISH\_id out RUBBISH.RUBBISH\_id%type

)

as

begin

select RUBBISH\_id into u\_RUBBISH\_id from order\_infor where order\_id = (select order\_id from (select order\_id from order where rownum=1 and order.user\_id=u\_user\_id));

dbms\_output.put\_line(u\_RUBBISH\_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 RUBBISH.RUBBISH\_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。

## 十、备份设计

**编写rman增量备份脚本**

#rman\_level1.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/lv1\_`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 1 database format '/home/oracle/rman\_backup/dblv1\_%d\_%T\_%U.bak'

plus archivelog format '/home/oracle/rman\_backup/arclv1\_%d\_%T\_%U.bak';

report obsolete;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

release channel c1;

}

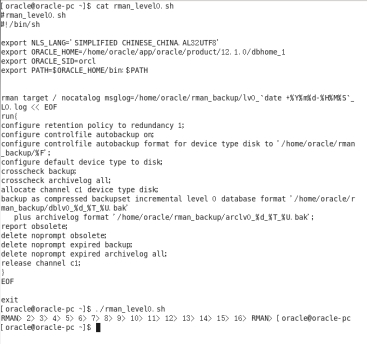
EOF

Exit

2. 开启全备份

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

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



3.每天定时开始增量备份

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

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

4. 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.删除数据库，模仿数据库被损坏

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

6. 修改

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

7. 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



8. 开始恢复数据库

oracle@oracle-pc ~]$ rman target /

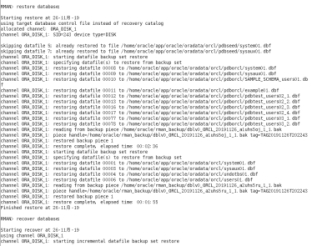
RMAN> restore database ;

RMAN> recover database;

RMAN> alter database open;

Statement processed

RMAN> exit



9. 查看数据库是否恢复

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

SQL> select \* from t9;

