

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的外卖系统的数据库设计 | | |
| 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 | 王浩 | 学号 | 201810414123 |
| 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 E-R模型设计 3](#_Toc22930)

[1.1 实体模型 3](#_Toc19292)

[1.2 实体联系模型 6](#_Toc15116)

[2 数据表的设计 7](#_Toc11711)

[2.1 用户表设计 7](#_Toc7570)

[2.2 外卖店表设计 7](#_Toc3784)

[2.3 菜品表设计 8](#_Toc28620)

[2.4 订单表设计 8](#_Toc2749)

[2.5 订单详单表设计 8](#_Toc11296)

[3 用户创建与表空间分配 10](#_Toc11454)

[3.1 表空间创建 10](#_Toc2622)

[3.2 角色和用户创建 11](#_Toc27299)

[4 表，程序包，函数和存储过程创建及执行计划分析 14](#_Toc23029)

[4.1 外卖用户表创建 14](#_Toc10477)

[4.2 外卖店表创建 17](#_Toc14828)

[4.3 菜品表创建 19](#_Toc20266)

[4.4 外卖订单表创建 21](#_Toc12533)

[4.5 外卖订单详单表创建 24](#_Toc30834)

[4.6 随机生成表数据 25](#_Toc9998)

[4.7 创建程序包、函数和过程及执行计划分析 27](#_Toc26866)

[5 备份与恢复 31](#_Toc25420)

[5.1 全库0级备份 31](#_Toc30876)

[5.2 全库1级增量备份 33](#_Toc7750)

[5.3 全库完全恢复 34](#_Toc25708)

[5.4 单库完全恢复 35](#_Toc29884)

# 1 E-R模型设计

## 1.1 实体模型

根据外卖系统的应用场景分析，共有5个实体（Entity）,它们分别是外卖用户、外卖店、菜品、外卖订单、外卖订单详情。具体说明如下：

1、外卖用户（WM\_USER）：用户包括用户ID（ID）,用户名（USER\_NAME）,用户密码（USER\_PWD）,用户地址（USER\_ADDRESS）,用户电话（USER\_PHONE）,余额（BALANCE）,注册时间（CREATE\_DATE）。用户实体如下图1.1.1。

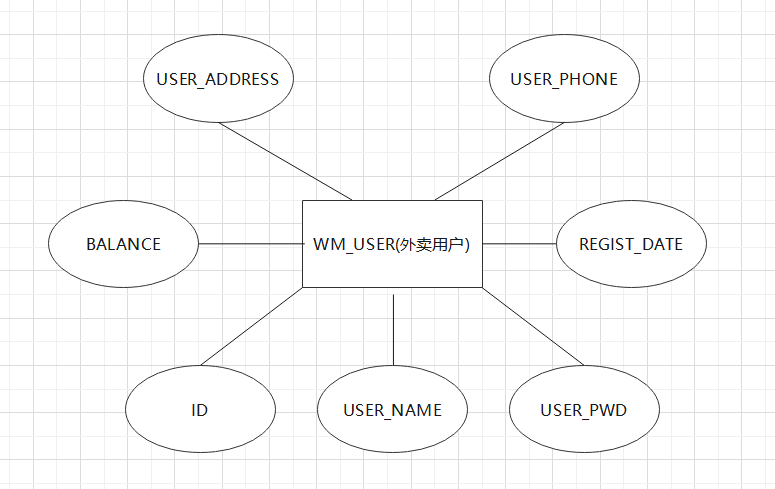


图 1.1.1 WM\_USER实体

2、外卖店（SHOP）：外卖店包括外卖店ID（ID）,外卖店名（USER\_NAME）,外卖店电话（SHOP\_PHONE）,销售量（SELL\_NUM）,评分（SCORE）。外卖店实体如下图1.1.2。

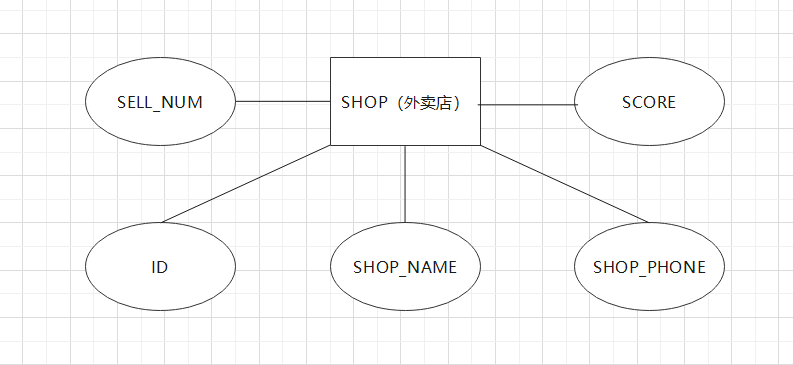


图 1.1.2 SHOP实体

3、菜品（DISH）：菜品包括菜品ID（ID）,菜品名（DISH\_NAME）,菜品价格（DISH\_PRICE）等,其中菜品的属性中还包括外卖店ID（SHOP\_ID），外卖店ID不能为空，表示菜品必须属于某个外卖店。菜品实体如下图1.1.3。

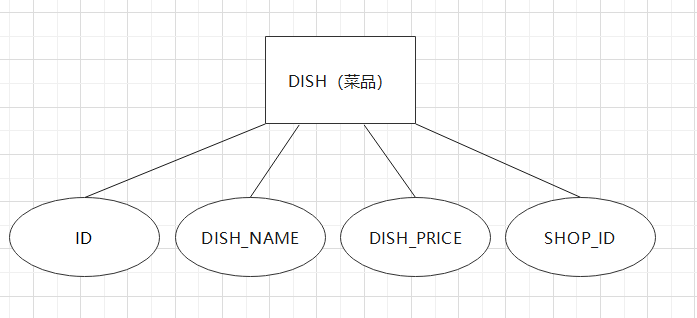


图 1.1.3 DISH实体

4、外卖订单（WM\_ORDER）：订单包括订单ID（ID）,下单时间（ORDER\_TIME）,订单总价（SUM\_PRICE）等,其中订单的属性中还包括用户（USER\_ID），用户ID不能为空，表示订单必须属于某个用户。订单的属性还有外卖店(SHOP\_ID),外卖店ID不能为空，一个订单必须来自某个外卖店。订单实体如下图1.1.4。

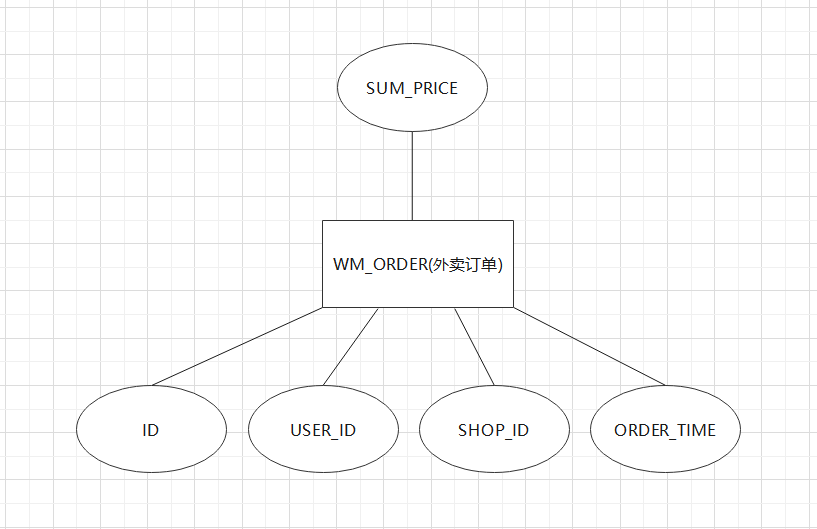


图 1.1.4 WM\_ORDER实体

5、订单详单（WM\_ORDER\_DETAIL）：订单详细包括订单详单ID（ID）,菜品价格（DISH\_PRICE）,菜品数量（DISH\_NUM）等,其中订单详单的属性中还包括菜品（DISH\_ID），菜品ID不能为空，表示订单详单必须属于含有菜品。订单的属性还有订单详单所属的订单(ORDER\_ID),订单ID不能为空，一个订单详单必须属于某个订单。订单详单实体如下图1.1.5。

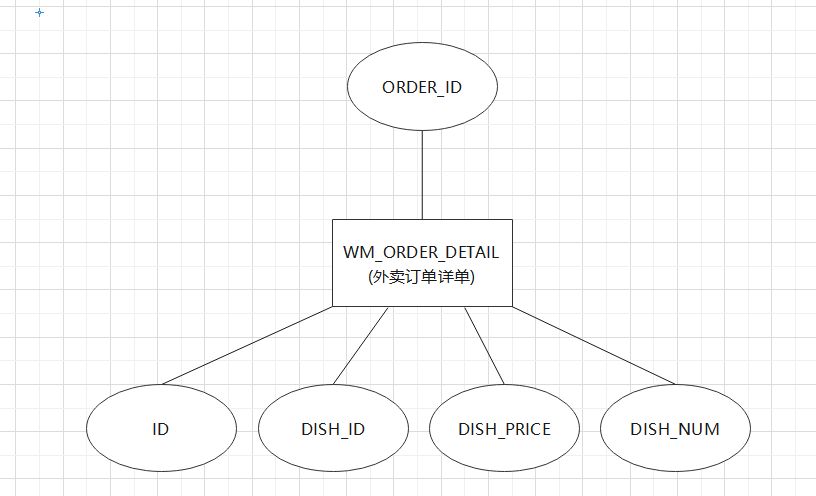


图 1.1.5 WM\_ORDER\_DETAIL实体

## 1.2 实体联系模型

1、用户可以点外卖，因此用户和菜品之间就有一个“购买”的联系，一个用户可以购买多个菜品且一个菜品可以被多个用户购买，所以菜品和用户是多对多的对应关系。外卖店可以“出售”多个菜品，因此外卖店和菜品之间存在一对多的对应关系。对应关系下图1.2.1。

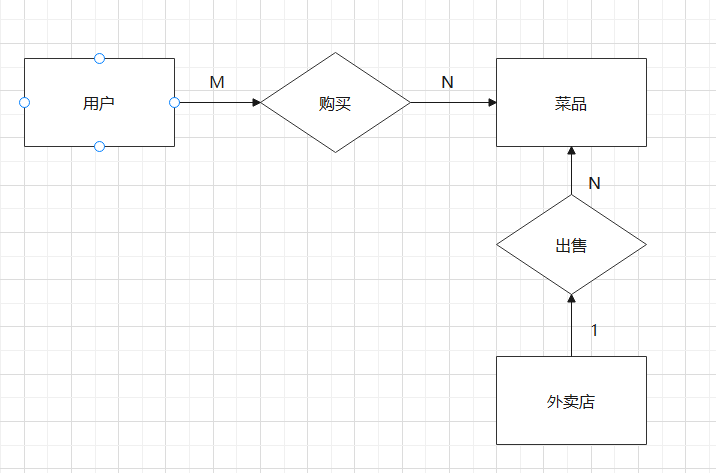


图 1.2.1 用户、菜品、外卖店关系简图

2、销售关系可以细分为订单和订单详单两个实体，订单中存储下单时间、送达时间、订单金额信息，订单详单中存储菜品ID、菜品数量、菜品价格信息。一个用户可以拥有多个订单，每个订单包含多个订单详单，一个菜品可以被多个订单详单包含。对应关系下图1.2.2。

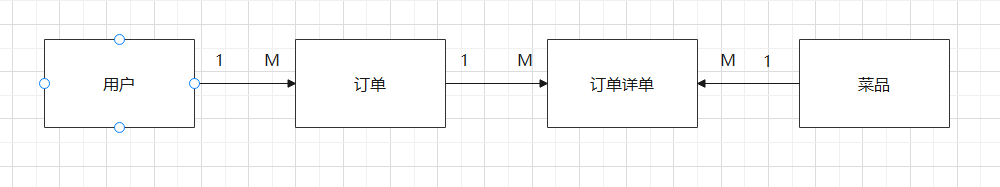


图 1.2.2 订单、订单详单关系简图

# 2 数据表的设计

## 2.1 用户表设计

外卖用户表（WM\_USER）包括ID、USER\_NAME、USER\_PWD、USER\_ADDRESS、USER\_PHONE、BALANCE、REGIST\_DATE属性。详见下表2.1.1。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ID | NUMBER(6,0) | NO | 用户ID,主键 |
| USER\_NAME | VARCHAR2(40 BYTE) | NO | 用户账户，非空 |
| USER\_PWD | VARCHAR2(40 BYTE) | NO | 用户密码，非空 |
| USER\_ADDRESS | VARCHAR2(80 BYET) | NO | 用户收货地址 |
| USER\_PHONE | VARCHAR2(40 BYTE) | NO | 用户电话号码 |
| BALANCE | NUMBER(8,2) | YES | 用户账号余额，默认为0 |
| REGIST\_DATE | DATE | NO | 注册日期，采用分区存储方式 |

表 2.1.1 用户表USER

## 2.2 外卖店表设计

外卖店表（SHOP）包括ID、SHOP\_NAME、SHOP\_PHONE、SELL\_NUM、SCORE属性。详见下表2.2.1。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ID | NUMBER(6,0) | NO | 外卖店ID,主键 |
| SHOP\_NAME | VARCHAR2(40 BYTE) | NO | 外卖店名，非空 |
| SHOP\_PHONE | VARCHAR2(40 BYTE) | NO | 外卖店电话号码 |
| SELL\_NUM | NUMBER(8,2) | YES | 外卖店销售总量，默认为0 |
| SCORE | NUMBER(8,2) | NO | 外卖店评分，只能取值：1，2，3，4，5 |

表 2.2.1 外卖店表SHOP

## 2.3 菜品表设计

菜品表（DISH）包括ID、DISH\_NAME、DISH\_PRICE、SHOP\_ID属性。详见下表2.3.1。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ID | NUMBER(6,0) | NO | 菜品ID,主键 |
| DISH\_NAME | VARCHAR2(40 BYTE) | NO | 菜品名，非空 |
| DISH\_PRICE | NUMBER(8,2) | NO | 菜品售价 |
| SHOP\_ID | NUMBER(6,0) | NO | 所属的外卖店号，外卖店表SHOP的外键 |

表 2.3.1 菜品表DISH

## 2.4 订单表设计

外卖订单表（WM\_ORDER）包括ID、USER\_ID、SHOP\_ID、ORDER\_TIME、SUM\_PRICE属性。详见下表2.4.1。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ID | NUMBER(10,0) | NO | 订单编号,主键,值来自于序列:SEQ\_ORDER\_ID |
| USER\_ID | NUMBER(6,0) | NO | 用户ID，用户表USER的外键 |
| SHOP\_ID | NUMBER(6,0) | NO | 外卖店ID，外卖店表SHOP的外键 |
| ORDER\_TIME | DATE | NO | 下单时间，应该采用分区存储方式 |
| SUM\_PRICE | SUM\_PRICE | YES | 外卖订单总金额，默认为0 |

表 2.4.1 订单表ORDER

## 2.5 订单详单表设计

外卖订单详单表（WM\_ORDER\_DETAIL）包括ID、ORDER\_ID、DISH\_NAME、DISH\_NUM,DISH\_PRICE属性。详见下表2.5.1。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ID | NUMBER(10,0) | NO | 订单详单编号,主键,值来自于序列:SEQ\_ORDER\_DETAILS\_ID |
| ORDER\_ID | NUMBER(10,0) | NO | 订单ID，订单表ORDER的外键 |
| DISH\_NAME | VARCHAR2(40 BYTE) | NO | 菜品名 |
| DISH\_NUM | NUMBER(8,2) | NO | 菜品销售数量，必须大于0 |
| DISH\_PRICE | NUMBER(8,2) | NO | 菜品售价 |

表 2.5.1 订单详单表ORDER\_DETAIL

# 3 用户创建与表空间分配

## 3.1 表空间创建

本次实验需要用到wh01、wh02两个表空间，下面用SYSTEM用户创建表空间wh01和wh02，给wh01表空间分配两个数据文件：pdbtest\_wh01\_1.dbf和pdbtest\_wh01\_2.dbf，这两个数据文件初始大小都为100M，即表空间的初始大小为200M。给wh02表空间分配两个数据文件：pdbtest\_wh02\_1.dbf和pdbtest\_wh02\_2.dbf，这两个数据文件初始大小都为100M，即表空间的初始大小为200M

代码如下：

CREATE TABLESPACE wh01 DATAFILE

'E:\app\root\oradata\orcl\pdborcl\pdbtest\_wh01\_1.dbf'

SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED，

'E:\app\root\oradata\orcl\pdborcl\pdbtest\_wh01\_2.dbf'

SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

CREATE TABLESPACE wh02 DATAFILE

'E:\app\root\oradata\orcl\pdborcl\pdbtest\_wh02\_1.dbf'

SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED，

'E:\app\root\oradata\orcl\pdborcl\pdbtest\_wh02\_2.dbf'

SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

运行截图如下：



## 3.2 角色和用户创建

创建shopkeeper和customer两个角色,tom和wanghao两个用户，给shopkeeper角色授予connect, resource, create view权限,给customer角色授予connect, resource权限，。用户默认使用表空间wh01,分配90M空间并将角色shopkeeper授权给用户tom,分配90M空间并将角色customer授权给用户wanghao。

代码如下：

CREATE ROLE shopkeeper;

GRANT connect,resource,CREATE VIEW TO shopkeeper;

CREATE USER tom IDENTIFIED BY 123 DEFAULT TABLESPACE wh01 TEMPORARY TABLESPACE temp;

ALTER USER tom QUOTA 90M ON wh01;

GRANT shopkeeper TO tom;

CREATE ROLE customer;

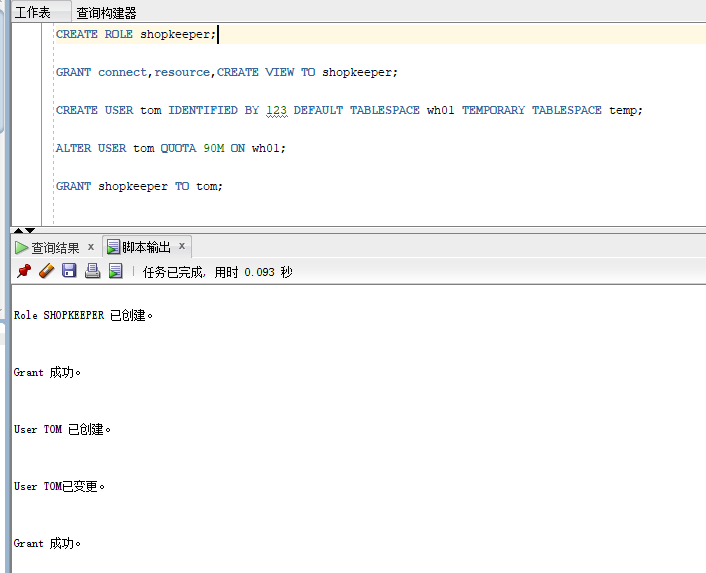
GRANT connect,resource TO customer;

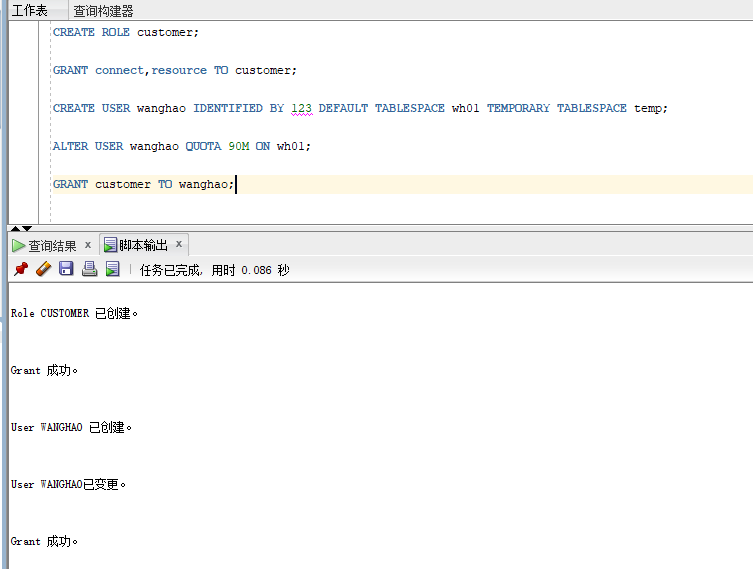
CREATE USER wanghao IDENTIFIED BY 123 DEFAULT TABLESPACE wh01 TEMPORARY TABLESPACE temp;

ALTER USER wanghao QUOTA 90M ON wh01;

GRANT customer TO wanghao;

截图如下：





# 4 表，程序包，函数和存储过程创建及执行计划分析

本次实验需要创建外卖用户表WM\_USER、外卖店表SHOP、菜品表DISH、外卖订单表WM\_ORDER、外卖订单详单表WM\_ORDER\_DETAIL五个表。

## 4.1 外卖用户表创建

创建外卖用户表WM\_USER，id为主键，根据注册日期按范围分区,将2019年之前到2020年之间的数据存储在wh01表空间中，将2021年数据存储在wh02表空间中。

代码如下：

CREATE TABLE WM\_USER

(

ID NUMBER(6, 0) NOT NULL

, USER\_NAME VARCHAR2(40 BYTE) NOT NULL

, USER\_PWD VARCHAR2(40 BYTE) NOT NULL

, USER\_ADDRESS VARCHAR2(80 BYTE) NOT NULL

, USER\_PHONE VARCHAR2(40 BYTE) NOT NULL

, BALANCE NUMBER(8,2) DEFAULT 0

, REGIST\_DATE DATE NOT NULL

, CONSTRAINT USER\_PK PRIMARY KEY

(

ID

)

USING INDEX

(

CREATE UNIQUE INDEX USER\_PK ON WM\_USER (ID ASC)

LOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 2

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

)

TABLESPACE WH01

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NOPARALLEL

PARTITION BY RANGE (REGIST\_DATE)

(

PARTITION PARTITION\_2019 VALUES LESS THAN (TO\_DATE(' 2020-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION PARTITION\_2020 VALUES LESS THAN (TO\_DATE(' 2021-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION PARTITION\_2021 VALUES LESS THAN (TO\_DATE(' 2022-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE WH02

PCTFREE 10

INITRANS 1

STORAGE

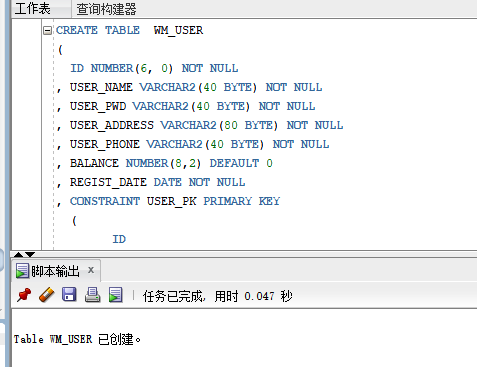
(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

);运行结果截图如下：



## 4.2 外卖店表创建

创建外卖用户表SHOP，id为主键。

代码如下：

CREATE TABLE SHOP

(

ID NUMBER(6, 0) NOT NULL

, SHOP\_NAME VARCHAR2(40 BYTE) NOT NULL

, SHOP\_PHONE VARCHAR2(40 BYTE) NOT NULL

, SELL\_NUM NUMBER(8,2) DEFAULT 0

, SCORE NUMBER(8,2) NOT NULL

, CONSTRAINT SHOP\_PK PRIMARY KEY

(

ID

)

USING INDEX

(

CREATE UNIQUE INDEX SHOP\_PK ON SHOP (ID ASC)

LOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 2

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

)

LOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

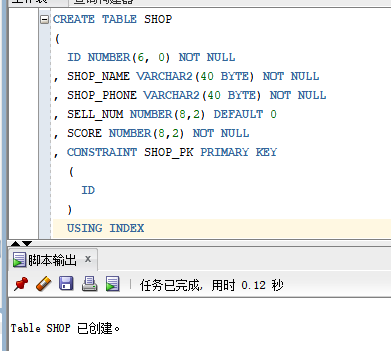
)

NOCOMPRESS

NO INMEMORY

NOPARALLEL;

运行结果截图如下：



## 4.3 菜品表创建

创建菜品表DISH，id为主键，SHOP\_ID为外卖店表SHOP的外键。

代码如下：

CREATE TABLE DISH

(

ID NUMBER(6, 0) NOT NULL

, DISH\_NAME VARCHAR2(40 BYTE) NOT NULL

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

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

, CONSTRAINT DISH\_PK PRIMARY KEY

(

ID

)

USING INDEX

(

CREATE UNIQUE INDEX DISH\_PK ON DISH (ID ASC)

LOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 2

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

, CONSTRAINT SHOP\_DISH FOREIGN KEY

(

SHOP\_ID

)

REFERENCES SHOP

(

ID

)

ENABLE

)

LOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

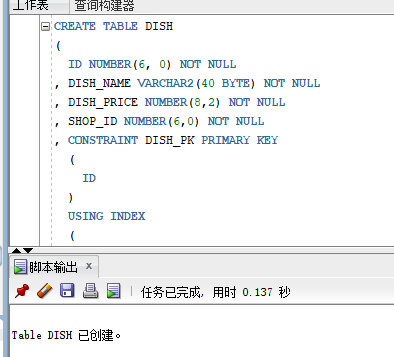
)

NOCOMPRESS

NO INMEMORY

NOPARALLEL;

运行结果截图如下：



## 4.4 外卖订单表创建

创建外卖订单表WM\_ORDER，id为主键，外卖订单表按范围分区进行存储，将2019年之前到2020年的数据存储在wh01表空间中，将2021年的数据存储在wh02表空间中。

代码如下：

CREATE TABLE WM\_ORDER

(

ID NUMBER(10,0) NOT NULL

, ORDER\_TIME DATE NOT NULL

, SUM\_PRICE NUMBER(8,2) DEFAULT 0

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

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

, CONSTRAINT ORDER\_PK PRIMARY KEY

(

ID

)

USING INDEX

(

CREATE UNIQUE INDEX ORDER\_PK ON WM\_ORDER (ID ASC)

LOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 2

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

)

TABLESPACE WH01

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NOPARALLEL

PARTITION BY RANGE (ORDER\_TIME)

(

PARTITION PARTITION\_2019 VALUES LESS THAN (TO\_DATE(' 2020-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION PARTITION\_2020 VALUES LESS THAN (TO\_DATE(' 2021-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE WH01

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION PARTITION\_2021 VALUES LESS THAN (TO\_DATE(' 2022-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE WH02

PCTFREE 10

INITRANS 1

STORAGE

(

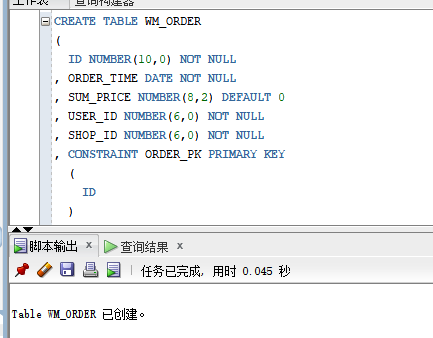
BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

);

运行结果截图如下：



## 4.5 外卖订单详单表创建

创建外卖订单表WM\_ORDER\_DETAIL，id为主键，使用引用分区进行存储让其数据也按主表WM\_ORDER的日期范围进行分区存储，通过引用分区语句"PARTITION BY REFERENCE(order\_detail\_fk1)",利用外键order\_detail\_fk1关联到主表wm\_order，使从表按主表的分区方案与主表存储在同一分区中。

代码如下：

CREATE TABLE wm\_order\_detail

(

id NUMBER(10, 0) NOT NULL

, order\_id NUMBER(10, 0) NOT NULL

, dish\_name VARCHAR2(40 BYTE) NOT NULL

, dish\_num NUMBER(8, 2) NOT NULL

, dish\_price NUMBER(8, 2) NOT NULL

, CONSTRAINT order\_detail\_fk1 FOREIGN KEY (order\_id)

REFERENCES wm\_order ( id )

ENABLE

)

TABLESPACE wh01

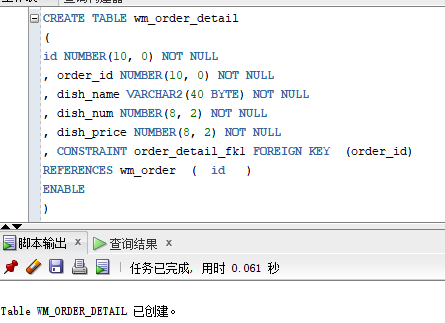
PCTFREE 10 INITRANS 1

STORAGE (BUFFER\_POOL DEFAULT )

NOCOMPRESS NOPARALLEL

PARTITION BY REFERENCE (order\_detail\_fk1);

运行结果截图：



## 4.6 随机生成表数据

本次实验使用PL/SQL完成随机数据的生成，使用for循环5w次，当循环次数小于1w时生成1w条用户数据，然后生成5w条外卖店、菜品、订单数据，每个订单关联三个订单详单。外卖订单和订单详单的生成顺序是生成外卖订单数据后给这个订单生成三个订单详单，最后计算得出的订单详单中菜品数量乘菜品价格的总和就是外卖订单表中订单总价。

部分代码如下：

declare

--定义外卖用户变量

v\_user\_id number(6);

v\_user\_name varchar2(100);

v\_user\_pwd varchar2(100);

......

begin

v\_order\_detail\_id:=0;

delete from wm\_order\_detail;

delete from wm\_order;

for i in 1..50000

loop

if i mod 3 =0 then

dt:=to\_date('2019-6-1','yyyy-mm-dd')+(i mod 60); --PARTITION\_2019

elsif i mod 3 =1 then

dt:=to\_date('2020-6-1','yyyy-mm-dd')+(i mod 60); --PARTITION\_2020

else

dt:=to\_date('2021-6-1','yyyy-mm-dd')+(i mod 60); --PARTITION\_2021

end if;

--插入用户数据

......

--计算每个订单的应收金额：

select sum(DISH\_NUM\*DISH\_PRICE) into m from WM\_ORDER\_DETAIL where ORDER\_ID=v\_order\_id;

if m is null then

m:=0;

end if;

UPDATE WM\_ORDER SET SUM\_PRICE = m WHERE ID=v\_order\_id;

IF I MOD 1000 =0 THEN

commit;

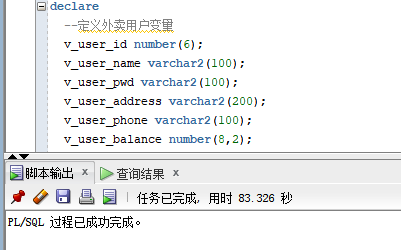
END IF;

end loop;

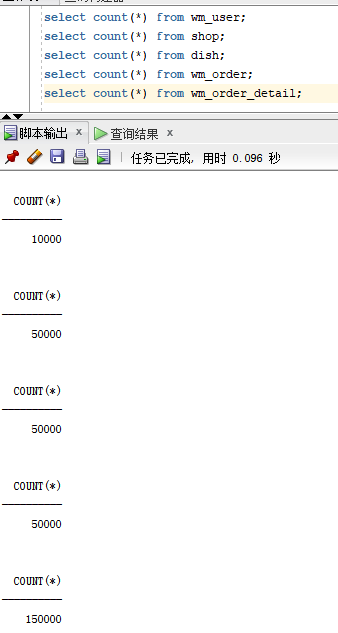
end;

/

运行结果截图如下：



查询每个表数据的数量：



## 4.7 创建程序包、函数和过程及执行计划分析

1、创建程序包waimai\_package，设计了函数get\_userpaybydate（）来查询用户点外卖花费的总金额，设计了过程add\_dish（）来根据外卖店id添加菜品。

代码如下：

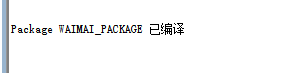
create or replace PACKAGE waimai\_package Is

function get\_userpay(user\_id number) return number;

procedure add\_dish(shop\_id number,dish\_name varchar2,dish\_price number);

end waimai\_package;

运行结果截图如下：



2、创建程序体定义函数和过程。

部分代码如下：

create or replace PACKAGE body waimai\_package Is

function get\_userpaybydate(user\_id number) return number as

begin

declare sum\_pay number;

query\_sql varchar2(200);

begin

query\_sql := 'select sum(sum\_price) from wm\_order where user\_id=' || user\_id ;

execute immediate query\_sql into sum\_pay;

return sum\_pay;

end;

end get\_userpaybydate;

procedure add\_dish(shop\_id number,dish\_name varchar2,dish\_price number) as

begin

declare maxId number;

begin

select max(id) into maxId from dish;

insert into dish values(maxId+1,dish\_name,dish\_price,shop\_id);

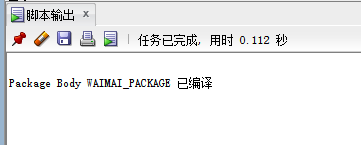
commit;

end;

end add\_dish;

end waimai\_package;

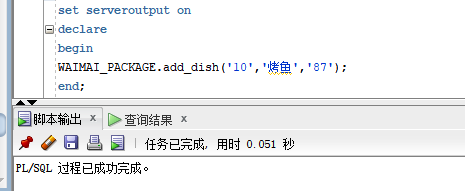
运行结果截图如下：



3、使用自定义函数get\_userpaybydate（）查询id为100的用户点的所有外卖的总金额。使用自定义存储过程add\_dish()插入菜品数据。

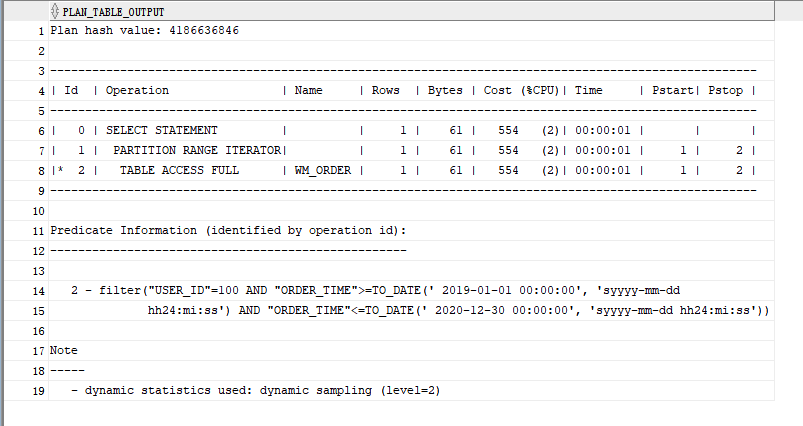
运行结果截图如下：



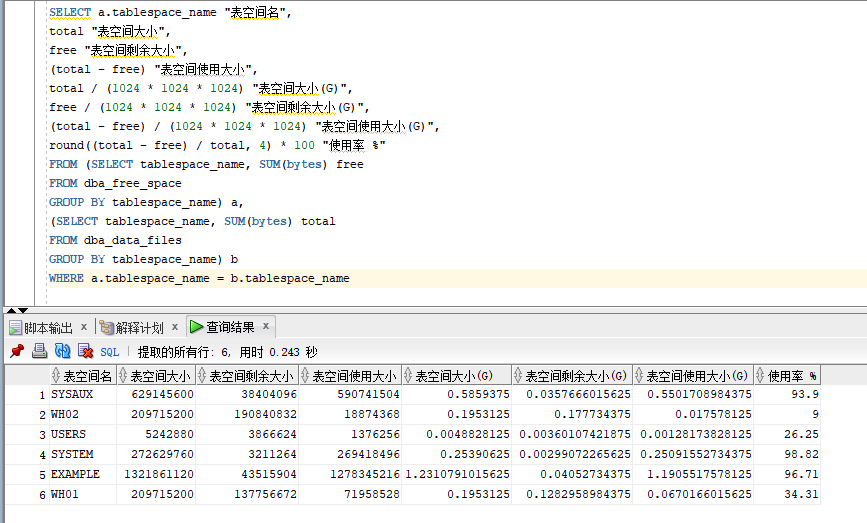




4、查询用户id为100的用户在2019-2020年间点外卖的所有订单的SQL语句执行计划分析。其中Rows=1，cost成本=554，一次全表搜索filter。在对于此实验中的大量数据执行查询操作时，由于我们按订单时间建立了范围分区，所以查询只需要在对应的Pstart=1和Pstop=2两个分区中进行查询，避免了对全部数据的搜索，极大提升了查询效率。



5、查询表空间使用情况



# 5 备份与恢复

查看pdborcl数据库的数据文件。

代码和运行结果如下：

sqlplus system/123 @ 127.0.0.1/pdborcl

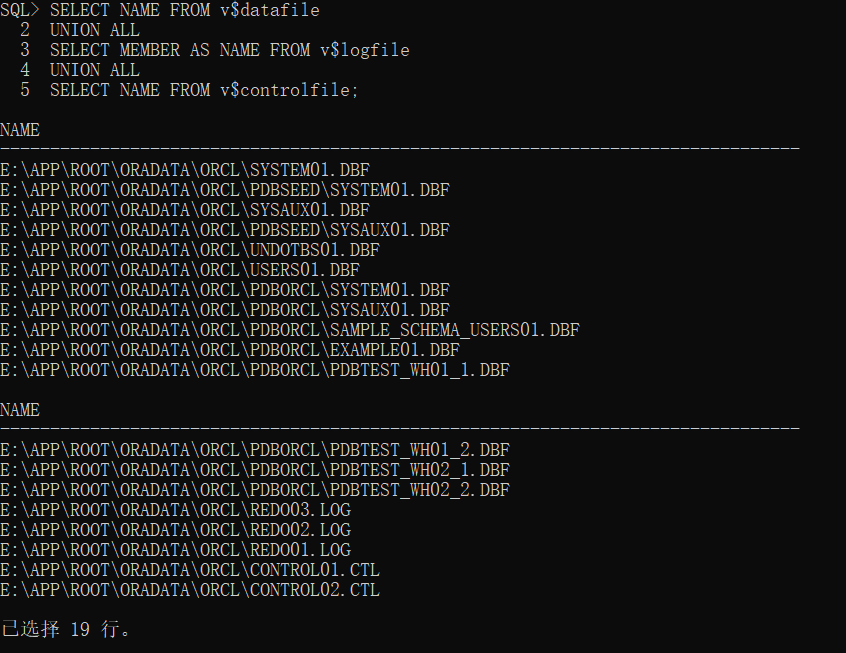
SELECT NAME FROM v$datafile

UNION ALL

SELECT MEMBER AS NAME FROM v$logfile

UNION ALL

SELECT NAME FROM v$controlfile;



## 5.1 全库0级备份

用sys用户以专用模式登录后进行全库0级备份（只作一次）。

代码如下：

rman target sys/123@127.0.0.1/orcl:dedicated

run{

configure retention policy to redundancy 1;

configure controlfile autobackup on;

configure controlfile autobackup format for device type disk to 'E:/APP/ROOT/rman\_backup/%F';

configure default device type to disk;

crosscheck backup;

crosscheck archivelog all;

allocate channel c1 device type disk;

allocate channel c2 device type disk;

allocate channel c3 device type disk;

backup incremental level 0 database format 'E:/APP/ROOT/rman\_backup/level0\_%d\_%T\_%U.bak';

report obsolete;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

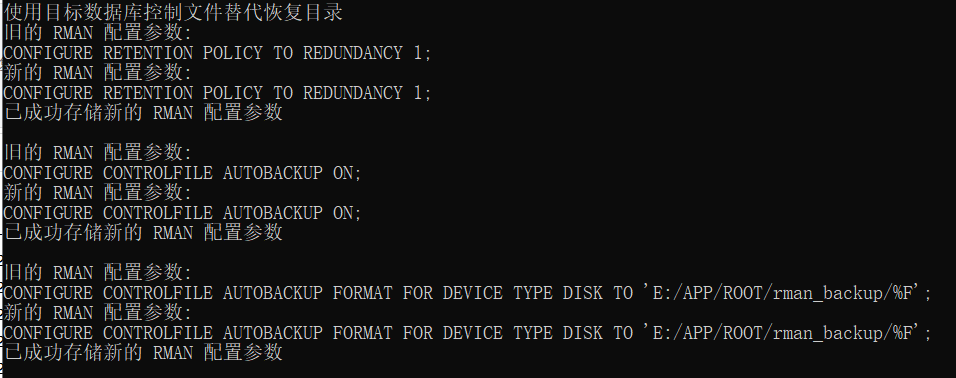
release channel c1;

release channel c2;

release channel c3;

}

运行结果截图如下：



## 5.2 全库1级增量备份

代码如下：

run{

configure retention policy to redundancy 1;

configure controlfile autobackup on;

configure controlfile autobackup format for device type disk to 'E:/APP/ROOT/rman\_backup/%F';

configure default device type to disk;

crosscheck backup;

crosscheck archivelog all;

allocate channel c1 device type disk;

allocate channel c2 device type disk;

allocate channel c3 device type disk;

backup incremental level 1 database format 'E:/APP/ROOT/rman\_backup/level1\_%d\_%T\_%U.bak';

report obsolete;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

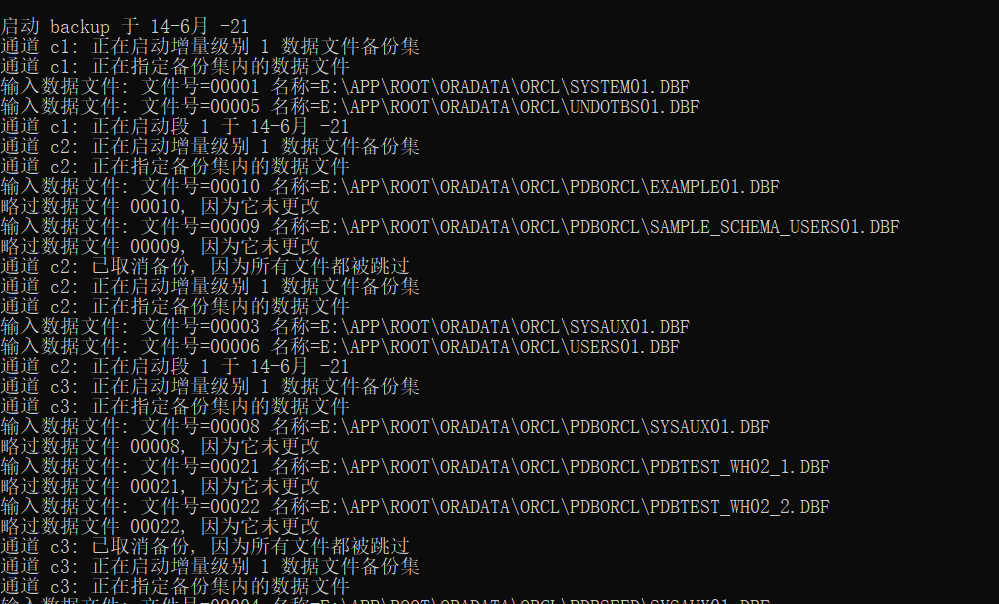
release channel c1;

release channel c2;

release channel c3;

}

运行结果截图如下：



## 5.3 全库完全恢复

代码如下：

sqlplus / as sysdba

SQL> select file\_name from dba\_data\_files;

- 全库停机

rman target /

RMAN> shutdown immediate; 或者 shutdown abort;

RMAN> exit

- 数据文件改名，模拟文件损失

del E:/app/root/oradata/orcl/pdborcl/pdbtest\_wh01\_1.dbf

- 全库恢复

rman target /

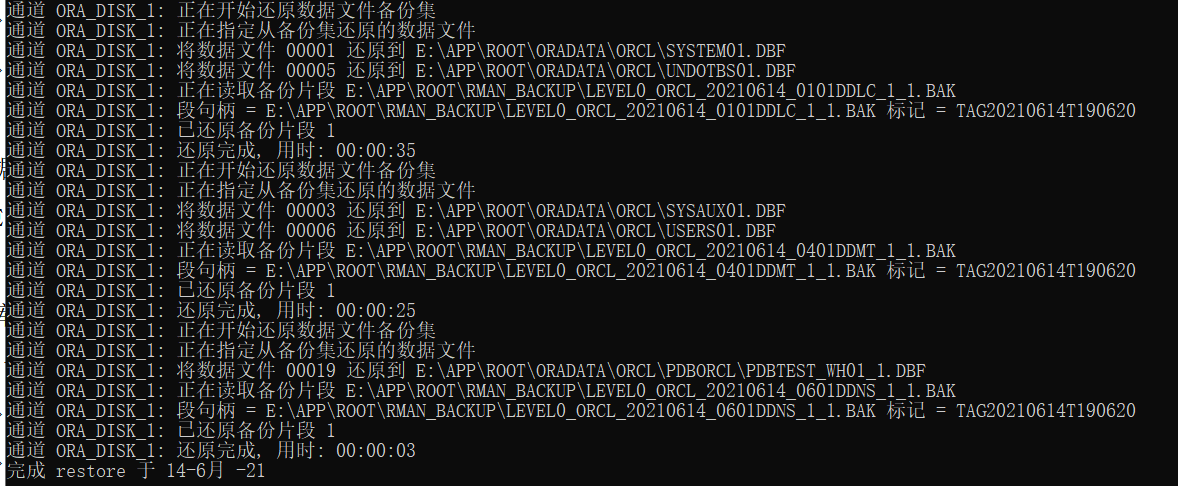
RMAN> startup mount;

RMAN> restore database;

RMAN> recover database;

RMAN> alter database open;

运行结果截图如下：



## 5.4 单库完全恢复

代码如下：

- system登录到pdborcl，查看pdborcl的数据文件

sqlplus system/123@127.0.0.1/pdborcl

SQL> select file\_name from dba\_data\_files;

- 关闭pdborcl数据库

rman target sys/123@127.0.0.1/orcl:dedicated

RMAN> alter pluggable database pdborcl close;

RMAN> exit;

- 模拟文件损失

del E:/app/root/oradata/orcl/pdborcl/pdbtest\_wh01\_1.dbf

del E:/app/root/oradata/orcl/pdborcl/pdbtest\_wh01\_2.dbf

- 单库完全恢复

rman target sys/123@127.0.0.1/orcl:dedicated

RMAN> restore pluggable database pdborcl;

RMAN> recover pluggable database pdborcl;

RMAN> alter pluggable database pdborcl open;

RMAN> exit;

运行结果截图如下：

