

**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 | 2017级 |
| 学生姓名 | 陈佳豪 | 学号 | 201710414201 |
| 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**

[一、 项目背景 1](#_Toc10339)6

[二、 系统用表 1](#_Toc1961)6

[1. 藏书信息表(book) 1](#_Toc2401)7

[2. 进货表(stock) 1](#_Toc16523)7

[3. 出货表(sold) 1](#_Toc21831)8

[4. 用户表(users) 1](#_Toc23255)8

[三、 数据初始化 1](#_Toc26511)9

[1. 藏书信息表插入数据 1](#_Toc28446)9

[2. 进货表插入数据 2](#_Toc32127)0

[3. 出货表插入数据 2](#_Toc6305)1

[4. 用户表插入数据 2](#_Toc5149)2

[四、 索引 2](#_Toc7151)3

[五、 视图 2](#_Toc13016)4

[1.进货信息视图 (stock\_infor) 2](#_Toc19310)4

[2. 出货信息视图 (sold\_infor) 2](#_Toc28776)4

[3.盈利视图(profit) 2](#_Toc24892)5

[六、 过程 2](#_Toc11178)6

[七、 函数 2](#_Toc268)8

[八、 包 2](#_Toc148)9

[九、 用户创建 3](#_Toc358)0

[1.创建用户 3](#_Toc24308)0

[2.创建角色 3](#_Toc16070)0

[3.授予角色权限 3](#_Toc18382)4

[十、 复杂功能实现 3](#_Toc15727)5

[十一、 Oracle体系结构 3](#_Toc19408)7

[十二：心得体会 3](#_Toc11923)8

# 任务

设计表5张表

向表中插入5万条数据

使用PL/SQL语言涉及存储过程与函数

对数据库进行备份

使用两台主机，对数据库进行容灾处理，实现数据库整体的异地备份

创建数据库：

CREATE PLUGGABLE DATABASE cjhpdb admin user cjh IDENTIFIED by 123 file\_name\_convert=

('/home/oracle/app/oracle/oradata/orcl/pdbseed','/home/oracle/app/oracle/oradata/orcl/cjhpdb');



切换pdb：

alter session set container=cjhpdb;



创建表空间：

CREATE TABLESPACE USERS02

DATAFILE

'/home/oracle/app/oracle/oradata/orcl/cjhpdb/pdbcjh\_users02\_1.dbf'

size 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED,

'/home/oracle/app/oracle/oradata/orcl/cjhpdb/pdbcjh\_users02\_2.dbf'

SIZE 100M AUTOEXTEND ON next 256M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

CREATE TABLESPACE USERS03

DATAFILE

'/home/oracle/app/oracle/oradata/orcl/cjhpdb/pdbcjh\_users03\_1.dbf'

size 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED,

'/home/oracle/app/oracle/oradata/orcl/cjhpdb/pdbcjh\_users03\_2.dbf'

SIZE 100M AUTOEXTEND ON next 256M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;



创建用户与角色，并给用户授权：

-- 创建usercjh用户：

CREATE USER usercjh IDENTIFIED BY 123

DEFAULT TABLESPACE "USERS02"

TEMPORARY TABLESPACE "TEMP";

-- 给用户usercjh分配表空间：

ALTER USER usercjh QUOTA UNLIMITED ON USERS02;

ALTER USER usercjh QUOTA UNLIMITED ON USERS03;

-- 创建角色并给用户分配权限：

CREATE ROLE cjhrole1;

GRANT connect,resource,CREATE VIEW TO cjhrole1;

GRANT cjhrole1 to usercjh;

CREATE ROLE cjhrole2;

GRANT dba TO cjhrole2;

GRANT cjhrole2 to usercjh;



创建BOOK表：

CREATE TABLE BOOK

(

BOOKNO NUMBER(20, 0) NOT NULL

, BOOKNAME VARCHAR2(20 BYTE)

, SPECIES VARCHAR2(5 BYTE) NOT NULL

, BOOKPRICE NUMBER(7, 2) NOT NULL

, PUBLISH\_HOUSE VARCHAR2(20 BYTE)

, PUBLISHDATE DATE

, AUTHOR VARCHAR2(10 BYTE) NOT NULL

, NUMBERS NUMBER

, CONSTRAINT SYS\_C0010008 PRIMARY KEY

(

BOOKNO

)

USING INDEX

(

CREATE UNIQUE INDEX SYS\_C0010008 ON BOOK (BOOKNO ASC)

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 2

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

)

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NOPARALLEL

PARTITION BY RANGE (PUBLISHDATE)

(

PARTITION P1 VALUES LESS THAN (TO\_DATE(' 2018-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION P2 VALUES LESS THAN (TO\_DATE(' 2019-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION P3 VALUES LESS THAN (TO\_DATE(' 2020-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

LOGGING

TABLESPACE USERS03

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION P4 VALUES LESS THAN (MAXVALUE)

LOGGING

TABLESPACE USERS03

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

);

ALTER TABLE BOOK

ADD CONSTRAINT SYS\_C0010009 UNIQUE

(

BOOKNAME

)

USING INDEX

(

CREATE UNIQUE INDEX SYS\_C0010009 ON BOOK (BOOKNAME ASC)

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 2

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

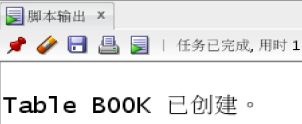
ENABLE;

ALTER TABLE BOOK

ADD CONSTRAINT SYS\_C0010007 CHECK

(numbers>=0)

ENABLE;



创建SOLD表：

CREATE TABLE SOLD

(

OONO VARCHAR2(6 BYTE) NOT NULL

, BOOKNO NUMBER(20, 0)

, NUMBERS NUMBER

, OUT\_PRICE NUMBER NOT NULL

, OUT\_TIME DATE NOT NULL

, CONSTRAINT SYS\_C0010013 PRIMARY KEY

(

OONO

)

USING INDEX

(

CREATE UNIQUE INDEX SYS\_C0010013 ON SOLD (OONO ASC)

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 2

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

)

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NO INMEMORY

NOPARALLEL;

ALTER TABLE SOLD

ADD CONSTRAINT FK\_SOLD FOREIGN KEY

(

BOOKNO

)

REFERENCES BOOK

(

BOOKNO

)

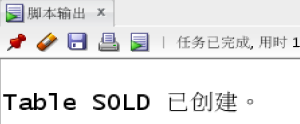
ENABLE;

ALTER TABLE SOLD

ADD CONSTRAINT SYS\_C0010012 CHECK

(numbers>=0)

ENABLE;



创建STOCK表：

CREATE TABLE STOCK

(

IONO VARCHAR2(6 BYTE) NOT NULL

, BOOKNO NUMBER(20, 0)

, NUMBERS NUMBER

, IN\_PRICE NUMBER NOT NULL

, IN\_TIME DATE NOT NULL

, CONSTRAINT SYS\_C0010018 PRIMARY KEY

(

IONO

)

USING INDEX

(

CREATE UNIQUE INDEX SYS\_C0010018 ON STOCK (IONO ASC)

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 2

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

)

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NO INMEMORY

NOPARALLEL;

ALTER TABLE STOCK

ADD CONSTRAINT FK\_STOCK FOREIGN KEY

(

BOOKNO

)

REFERENCES BOOK

(

BOOKNO

)

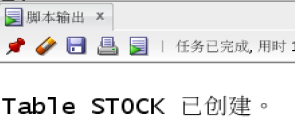
ENABLE;

ALTER TABLE STOCK

ADD CONSTRAINT SYS\_C0010017 CHECK

(numbers>=0)

ENABLE;



创建BO\_USER表：

CREATE TABLE BO\_USER

(

BO\_USERNAME VARCHAR2(10 BYTE) NOT NULL

, BO\_PASSWORD VARCHAR2(15 BYTE) NOT NULL

, SAL NUMBER

, COMM NUMBER

, REMARK VARCHAR2(15 BYTE)

)

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NO INMEMORY

NOPARALLEL;



创建TABLET表：

CREATE TABLE TABLET

(

T\_ID NUMBER(10, 0)

, T\_DATE DATE

)

LOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NO INMEMORY

NOPARALLEL;



插入5万条数据：

declare

u\_name VARCHAR2(10);

u\_pwd VARCHAR2(15);

u\_sal NUMBER;

u\_comm NUMBER;

u\_re VARCHAR2(15);

begin

/\*

system login:

ALTER USER "TEACHER" QUOTA UNLIMITED ON USERS;

ALTER USER "TEACHER" QUOTA UNLIMITED ON USERS02;

ALTER USER "TEACHER" QUOTA UNLIMITED ON USERS03;

\*/

/\*v\_order\_detail\_id:=1;

delete from order\_details;

delete from orders;

\*/

for i in 1..50000

loop

-- if i mod 6 =0 then

-- dt:=to\_date('2018-01-01','yyyy-mm-dd')+(i mod 60); --PARTITION\_2018

-- elsif i mod 6 =1 then

-- dt:=to\_date('2019-01-01','yyyy-mm-dd')+(i mod 60); --PARTITION\_2019

-- elsif i mod 6 =2 then

-- dt:=to\_date('2020-01-01','yyyy-mm-dd')+(i mod 60); --PARTITION\_2020

-- 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

u\_name:=CASE I MOD 6 WHEN 0 THEN 'zkx' WHEN 1 THEN 'yjq' WHEN 2 THEN 'lh'

WHEN 3 THEN 'yyf' WHEN 4 THEN 'wc' ELSE 'cjh' END;

u\_pwd:='123';

u\_sal:=CASE I MOD 6 WHEN 0 THEN 5000 WHEN 1 THEN 10000 WHEN 2 THEN 15000

WHEN 3 THEN 20000 WHEN 4 THEN 25000 ELSE 30000 END;

u\_comm:=5000;

u\_re:=CASE I MOD 6 WHEN 0 THEN 'admin' WHEN 1 THEN 'student' WHEN 2 THEN 'teacher'

ELSE 'putong' END;

insert /\*+append\*/ into BO\_USER (BO\_USERNAME,BO\_PASSWORD,SAL,COMM,REMARK)

values (u\_name,u\_pwd,u\_sal,u\_comm,u\_re);

--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\_NAME,PRODUCT\_NUM,PRODUCT\_PRICE)

-- values (v\_order\_detail\_id,v\_order\_id,v\_name,2,v);

-- v:=dbms\_random.value(1000,50);

-- v\_name:='paper'|| (i mod 3 + 1);

-- v\_order\_detail\_id:=v\_order\_detail\_id+1;

-- insert /\*+append\*/ into ORDER\_DETAILS(ID,ORDER\_ID,PRODUCT\_NAME,PRODUCT\_NUM,PRODUCT\_PRICE)

-- values (v\_order\_detail\_id,v\_order\_id,v\_name,3,v);

-- v:=dbms\_random.value(9000,2000);

-- v\_name:='phone'|| (i mod 3 + 1);

--

-- v\_order\_detail\_id:=v\_order\_detail\_id+1;

-- insert /\*+append\*/ into ORDER\_DETAILS(ID,ORDER\_ID,PRODUCT\_NAME,PRODUCT\_NUM,PRODUCT\_PRICE)

-- values (v\_order\_detail\_id,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;

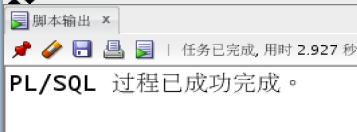
-- IF I MOD 1000 =0 THEN

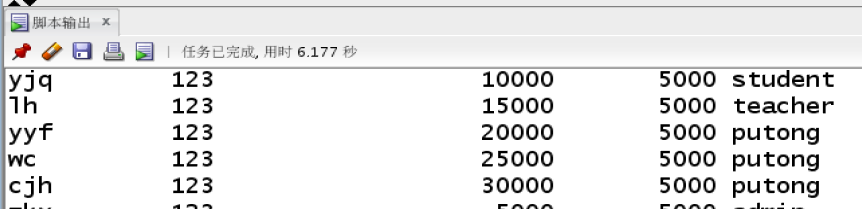
-- commit; --每次提交会加快插入数据的速度

-- END IF;

end loop;

end;





创建程序包：

create or replace PACKAGE MyPack IS

/\*

包MyPack中有：

一个函数:Get\_Count，

一个过程:Get\_UserType 统计每个用户类型的总数

\*/

FUNCTION Get\_Count RETURN NUMBER;

PROCEDURE Get\_UserType;

END MyPack;

create or replace PACKAGE BODY MyPack IS

FUNCTION Get\_Count RETURN NUMBER

AS

N NUMBER(20,2);

BEGIN

SELECT COUNT(\*) into N FROM BO\_USER;

RETURN N;

END;

PROCEDURE Get\_UserType

AS

LEFTSPACE VARCHAR(2000);

begin

--通过LEVEL判断递归的级别

LEFTSPACE:=' ';

--使用游标

for v in

(SELECT REMARK,SUM(REMARK) U\_SUM FROM BO\_USER

GROUP BY REMARK)

LOOP

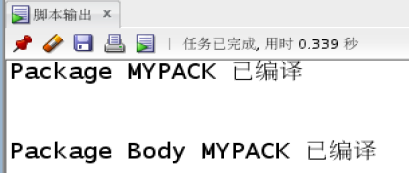
DBMS\_OUTPUT.PUT\_LINE(LPAD(LEFTSPACE,(1+1),' ')||

V.REMARK||' '||v.U\_SUM);

END LOOP;

END;

END MyPack;



测试Get\_Count函数：

SELECT MyPack.Get\_Count as 总记录数 from dual;



测试Get\_UserType存储过程：

set SERVEROUTPUT ON

BEGIN

MyPack.Get\_UserType;

END;

编写脚本文件rman\_cjh\_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



开始备份：

[oracle@oracle-pc ~]$ chmod 755 rman\_cjh\_level0.sh

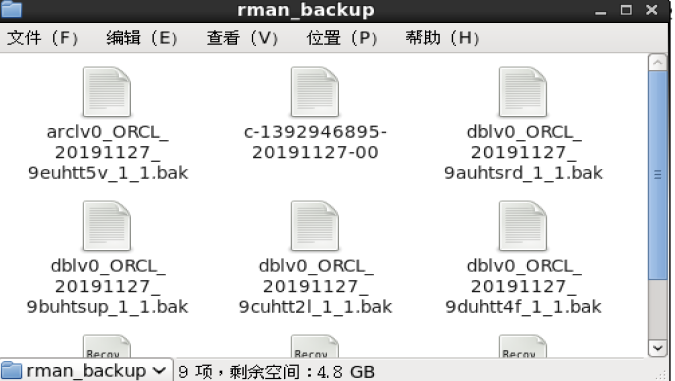
[oracle@oracle-pc ~]$ ./rman\_

bash: ./rman\_: 没有那个文件或目录

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

RMAN> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15> 16> RMAN> [oracle@oracle-pc ~]$

[oracle@oracle-pc ~]$



# 

# 书店图书销售系统

1. **项目背景**

随着信息时代的到来，IT产业和Internet获得了飞速发展，计算机应用已渗透到了各个领域，网络应用越来越普遍，而且走进了千家万户，一直使用手工记账的小镇书店老板，也想购买一个图书销售系统，但想先体验一下计算机的快捷方便。因此，设计了如下测试用例，供其初步体验！

1. **系统用表**

系统用表共四张，分别为：**藏书信息表(book)**、**进货表(stock)**、**出货表(sold)**、**用户表(users)**。

**藏书信息表**：用来存放书店现有图书的信息。

**进货表**：用来存放书店进货信息。

**出货表**：用来存放书店卖出的图书信息。

**用户表**：用来存放书店员工信息。

**建表：**

**1. 藏书信息表(book)**

藏书信息表(book) 表1

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 描述 |
| bookno | varchar2 | 6 | 图书编号，定义为主键 |
| bookname | varchar2 | 20 | 书名， 唯一 |
| species | varchar2 | 5 | 图书分类， 不允许为空 |
| bookprice | number | (7,2) | 图书单价， 不允许为空 |
| publish\_house | varchar2 | 20 | 出版社， 不允许为空 |
| publishdate | date |  | 出版日期， 不允许为空 |
| author | varchar2 | 10 | 作者， 不允许为空 |
| number | number | 4 | 藏书量， 大于0 |

建表语句：

create table book

( bookno varchar2(6) primary key,

bookname varchar2(20) unique,

species varchar2(5) not null,

bookprice number(7,2) not null,

publish\_house varchar2(20),

publishdate date,

author varchar2(10) not null,

numbers number check(numbers>=0) );

修改表结构：

alter table book modify species varchar2(15);

alter table book modify bookname varchar2(20);

alter table book modify numbers number(4);

**2. 进货表(stock)**

进货表(stock) 表2

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 描述 |
| iono | varchar2 | 6 | 进货编号，定义为主键 |
| bookno | varchar2 | 6 | 图书编号，定义为外键 |
| numbers | number |  | 进货量， 大于等于0 |
| in\_price | number |  | 进价，不能为空 |
| in\_time | date |  | 进货时间，不能为空 |

建表语句：

create table stock

(iono varchar2(6) primary key,

bookno varchar2(6),

numbers number check(numbers>=0),

in\_price number not null,

in\_time date not null,

constraint fk\_stock foreign key(bookno) references book(bookno));

1. **出货表(sold)**

出货表(sold) 表3

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 描述 |
| oono | varchar2 | 6 | 出货编号，定义为主键 |
| bookno | varchar2 | 6 | 图书编号，定义为外键 |
| numbers | number |  | 出货量， 大于等于0 |
| out\_price | number |  | 售卖价格，不能为空 |
| out\_time | date |  | 售卖时间，不能为空 |

建表语句：

create table sold

(oono varchar2(6) primary key,

bookno varchar2(6),

numbers number check(numbers>=0),

out\_price number not null,

out\_time date not null,

constraint fk\_sold foreign key(bookno) references book(bookno));

1. **用户表(users)**

用户表(users) 表4

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 描述 |
| username | varchar2 | 10 | 用户名，定义为主键 |
| password | varchar2 | 15 | 密码， 不允许为空 |
| sal | number |  | 月薪，可以为空 |
| comm | number |  | 奖金，可以为空 |
| remark | varchar2 | 15 | 用户类型，允许为空 |

建表语句：

create table users

(username varchar2(10) primary key,

password varchar2(15) not null,

sal number,

comm number,

remark varchar2(15));

1. **数据初始化**
2. **藏书信息表插入数据**

--insert语句插入

insert into book values('001010','《细说PHP》','IT',109.00,'电子工业出版社',to\_date('2013-04','yyyy-mm'),'高洛峰',50);

--替代变量插入--语句保存到d:\oracle\replace\_sql\insert\_book.sql

accept v\_bookno prompt '编号(六位数字)：'

accept v\_bookname prompt '书名《书名》：'

accept v\_species prompt '种类：'

accept v\_bookprice prompt '价格：'

accept v\_ph prompt '出版社：'

accept v\_pt prompt '出版日期（yyyy-mm）：'

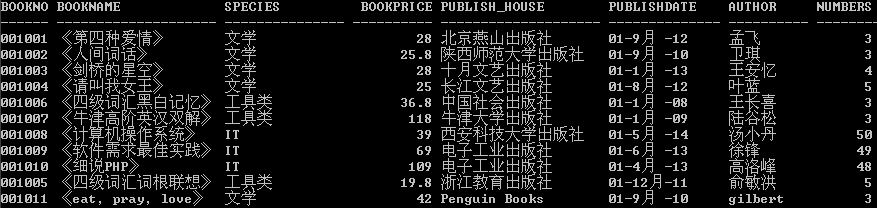
accept v\_author prompt '作者：'

accept v\_numbers prompt '库存量（>=0）：'

insert into boss.book values

('&v\_bookno','&v\_bookname','&v\_species',&v\_bookprice,'&v\_ph',to\_date('&v\_pt','yyyy-mm'),'&v\_author',&v\_numbers);

执行：start d:\oracle\replace\_sql\insert\_book.sql

****

1. **进货表插入数据**

--insert语句插入

insert into stock values

('000001','001001',3,15.00,to\_date('2014-11-07','yyyy-mm-dd'));

--替代变量插入--语句保存到d:\oracle\replace\_sql\insert\_stock.sql

accept v\_iono prompt '进货编号(六位数字)：'

accept v\_bookno prompt '图书编号(六位数字)：'

accept v\_numbers prompt '进货数量：'

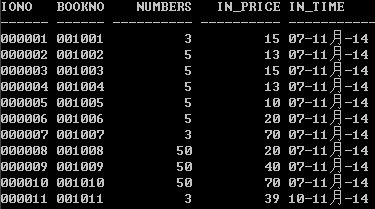
accept v\_in\_price prompt '进货价格：'

accept v\_in\_time prompt '进货日期(yyyy-mm-dd)：'

insert into stock values

('&v\_iono','&v\_bookno',&v\_numbers,&v\_in\_price,to\_date('&v\_in\_time','yyyy-mm-dd'));

执行：start d:\oracle\replace\_sql\insert\_stock.sql



1. **出货表插入数据**

--insert语句插入

insert into sold values

('000001','001002',1,25.80,to\_date('2014-11-07','yyyy-mm-dd'));

--替代变量插入--语句保存到d:\oracle\replace\_sql\insert\_sold.sql

accept v\_oono prompt '出货编号(六位数字)：'

accept v\_bookno prompt '图书编号(六位数字)：'

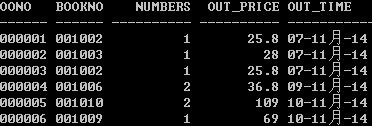
accept v\_numbers prompt '售卖数量：'

accept v\_in\_price prompt '售卖价格：'

accept v\_in\_time prompt '售卖日期(yyyy-mm-dd)：'

insert into sold values('&v\_oono','&v\_bookno',&v\_numbers,&v\_out\_price,to\_date('&v\_out\_time','yyyy-mm-dd'));

执行：start d:\oracle\replace\_sql\insert\_sold.sql

****

1. **用户表插入数据**

--insert语句插入

insert into users values('boss','boss','','','admin');

--替代变量插入--语句保存到d:\oracle\replace\_sql\insert\_users.sql

accept v\_name prompt '用户名：'

accept v\_pword prompt '密码：'

accept v\_sal prompt '月薪：'

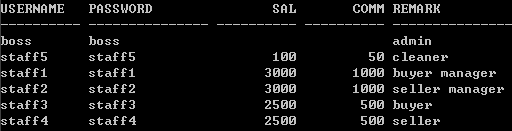
accept v\_comm prompt '奖金：'

accept v\_remark prompt '备注 ：'

insert into users values

('&v\_name','&v\_pword',&v\_sal,&v\_comm,'&v\_remark');

执行：start d:\oracle\replace\_sql\insert\_users.sql

****

1. **索引**

**1.由于对book表的图书的分类查找会比较频繁，所以对book表的species列建索引**

create index book\_species\_idx on book(species);

**2.由于对stock表的进货量，进价，进货时间的查询会比较频繁，所以对stock表的**

**numbers，in\_price，in\_time列建索引**

create index stock\_num\_ip\_idx on stock(numbers,in\_price);

create index stock\_num\_it\_idx on stock(numbers,in\_time);

**3.由于对sold表的进货量，进价，进货时间的查询会比较频繁，所以对sold表的**

**numbers，out\_price，out\_time列建索引**

create index sold\_num\_op\_idx on sold(numbers,out\_price);

create index sold\_num\_ot\_idx on sold(numbers,out\_time);

1. **视图**

视图共三个，分别为：**进货信息视图(stock\_infor)**、**出货信息视图(sold\_infor)**、**盈利视图(profit)**。

**进货信息视图**：由藏书信息表和进货表整合。包括进货编号，图书编号，书名，类别，进价，数量，总进价。

**出货信息视图**：由藏书信息表和出货表整合。包括出货编号，图书编号，书名，类别，卖价，售卖数量，总价，售卖日期。

**盈利视图**：由进货表和出货表整合。包括图书编号，进价，卖价，数量，盈利，售卖日期。

**1.进货信息视图 (stock\_infor)**

建视图语句：

create view stock\_infor

as

select s.iono"进货编号",b.bookno"图书编号",b.bookname"书名 ",b.species"类别",s.in\_price"进价",s.numbers"数量 ",s.in\_price\*s.numbers"总价",s.in\_time"进货日期"

from book b,stock s

where b.bookno=s.bookno;

****

1. **出货信息视图 (sold\_infor)**

建视图语句：

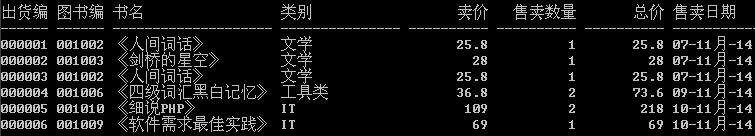
create view sold\_infor

as

select s.oono"出货编号",b.bookno"图书编号",b.bookname"书 名 ",b.species"类别",s.out\_price"卖价",s.numbers"售卖数量 ",s.out\_price\*s.numbers"总价",s.out\_time"售卖日期"

from book b,sold s

where b.bookno=s.bookno;



**3.盈利视图(profit)**

建视图语句：

create view profit

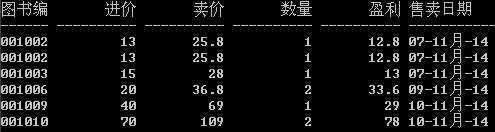
as

select so.bookno"图书编号",st.in\_price"进价",so.out\_price"卖价",so.numbers"数量",((so.out\_price-st.in\_price)\*so.numbers)"盈利",so.out\_time"售卖日期"

from (select distinct bookno,in\_price from stock) st,sold so

where st.bookno=so.bookno

order by so.out\_time,so.bookno;

****

1. **过程**
2. **输入（图书编号，增/减，数量）实现增减库存----条件选择**

create or replace procedure xgkc(bno varchar2,sign char,num number)is

v\_bno boss.book.bookno%type;

begin

select bookno into v\_bno from boss.book where bookno=bno;

if sign='+' then

update boss.book set numbers=numbers+num where bookno=v\_bno;

dbms\_output.put\_line(v\_bno||'号图书成功增加库存'||num);

elsif sign='-' then

update boss.book set numbers=numbers-num where bookno=v\_bno;

dbms\_output.put\_line(v\_bno||'号图书成功减少库存'||num);

else

dbms\_output.put\_line('第二个参数只能为+/-');

end if;

exception

when no\_data\_found then

dbms\_output.put\_line('库存表中不存在该图书，请修改图书编号或增加该图书信息到库存中');

end;

/

执行：exec xgkc('001002','-',1);

exec xgkc('001002','+',1);

1. **输入（图书类型），根据参数传递来查询图书编号，图书名称，价格----游标**

create or replace procedure lxcx(c1species varchar2)is

type book\_record\_type is record(

v\_no boss.book.bookno%type,

v\_name boss.book.bookname%type,

v\_price boss.book.bookprice%type);

book\_no\_name\_price book\_record\_type;

cursor c1(c1species varchar2) return book\_no\_name\_price is

select bookno,bookname,bookprice from boss.book where species=c1species;

begin

open c1 (c1species);

loop

fetch c1 into book\_no\_name\_price;

if c1%found then

dbms\_output.put\_line('类型为'||c1species||'，编号为'||book\_no\_name\_price.v\_no||'，名称为'||book\_no\_name\_price.v\_name||'，价格为'||book\_no\_name\_price.v\_price);

else

dbms\_output.put\_line('查询结束！');

exit;

end if;

end loop;

close c1;

end;

/

执行：exec lxcx('IT');

1. **执行过程，输出提示库存少于5的图书信息方便进货。**

create or replace procedure jhtx is

type book\_record\_type is record(

v\_bno boss.book.bookno%type,

v\_bname boss.book.bookname%type,

v\_num boss.book.numbers%type,

v\_sum boss.book.numbers%type);

v\_no\_name\_num\_sum book\_record\_type;

cursor c1 return v\_no\_name\_num\_sum is

select b.bookno,b.bookname,b.numbers,s.sumnumber from boss.book b,(select 图书编号,sum(售卖数量) sumnumber from boss.sold\_infor group by 图书编号) s where b.bookno=图书编号 and b.numbers<=5;

begin

open c1;

loop

fetch c1 into v\_no\_name\_num\_sum;

if c1%found then

dbms\_output.put\_line('图书编号'||v\_no\_name\_num\_sum.v\_bno||'图书名称'||v\_no\_name\_num\_sum.v\_bname||'现库存量--'

||v\_no\_name\_num\_sum.v\_num||

'--已售卖--'||v\_no\_name\_num\_sum.v\_sum);

else

dbms\_output.put\_line('查询结束！');

exit;

end if;

end loop;

close c1;

end;

/

执行：exec jhtx;

1. **函数**

**输入（员工姓名），求出员工年薪**

create or replace function y\_sal (name varchar2) return number is ssal number;

begin

select sal\*12+nvl(comm,0)\*12 into ssal from boss.users

where username=name;

return ssal;

end;

/

调用：var yearsal number

call y\_sal('staff2') into:yearsal;

print yearsal

1. **包**

创建一个包实现以下功能：

1. 输入（图书编号，增/减，数量）实现增减库存
2. 查询某类型所有图书的图书编号，图书名称和价格

3.执行过程，输出提示库存少于5的图书信息方便进货。

4.输入（员工姓名），求出员工年薪

**--声明包**

create or replace package pk is

procedure xgkc(bno varchar2,sign char,num number);

procedure lxcx(c1species varchar2);

procedure jhtx;

function y\_sal (name varchar2) return number;

end;

/

**--创建包体**

create or replace package body pk is

procedure xgkc(bno varchar2,sign char,num number)is

v\_bno boss.book.bookno%type;

begin

select bookno into v\_bno from boss.book where bookno=bno;

if sign='+' then

update boss.book set numbers=numbers+num where bookno=v\_bno;

dbms\_output.put\_line(v\_bno||'号图书成功增加库存'||num);

elsif sign='-' then

update boss.book set numbers=numbers-num where bookno=v\_bno;

dbms\_output.put\_line(v\_bno||'号图书成功减少库存'||num);

else

dbms\_output.put\_line('第二个参数只能为+/-');

end if;

exception

when no\_data\_found then

dbms\_output.put\_line('库存表中不存在该图书，请修改图书编号或增加该图书信息到库存中');

end;

procedure lxcx(c1species varchar2)is

type book\_record\_type is record(

v\_no boss.book.bookno%type,

v\_name boss.book.bookname%type,

v\_price boss.book.bookprice%type);

book\_no\_name\_price book\_record\_type;

cursor c1(c1species varchar2) return book\_no\_name\_price is

select bookno,bookname,bookprice from boss.book where species=c1species;

begin

open c1 (c1species);

loop

fetch c1 into book\_no\_name\_price;

if c1%found then

dbms\_output.put\_line('类型为'||c1species||'，编号为'||book\_no\_name\_price.v\_no||'，名称为'||book\_no\_name\_price.v\_name||'，价格为'||book\_no\_name\_price.v\_price);

else

dbms\_output.put\_line('查询结束！');

exit;

end if;

end loop;

close c1;

end;

procedure jhtx is

type book\_record\_type is record(

v\_bno boss.book.bookno%type,

v\_bname boss.book.bookname%type,

v\_num boss.book.numbers%type,

v\_sum boss.book.numbers%type);

v\_no\_name\_num\_sum book\_record\_type;

cursor c1 return v\_no\_name\_num\_sum is select b.bookno,b.bookname,b.numbers,s.sumnumber from boss.book b,(select 图书编号,sum(售卖数量) sumnumber from boss.sold\_infor group by 图书编号) s where b.bookno=图书编号 and b.numbers<=5;

begin

open c1;

loop

fetch c1 into v\_no\_name\_num\_sum;

if c1%found then

dbms\_output.put\_line('图书编号'||v\_no\_name\_num\_sum.v\_bno||'图书名称'||v\_no\_name\_num\_sum.v\_bname||

'现库存量--'||v\_no\_name\_num\_sum.v\_num||

'--已售卖--'||v\_no\_name\_num\_sum.v\_sum);

else

dbms\_output.put\_line('查询结束！');

exit;

end if;

end loop;

close c1;

end;

function y\_sal (name varchar2) return number is ssal number;

begin

select (sal+nvl(comm,0))\*12 into ssal from boss.users where username=name;

return ssal;

end;

end;

/

执行：exec pk.xgkc('001002','-',1);

exec pk.xgkc('001002','+',1);

exec pk.lxcx('IT');

exec pk.jhtx;

var yearsal number

call pk.y\_sal('staff1') into:yearsal;

print yearsal

1. **用户创建**

**1.创建用户**

**创建boss用户并授予最高权限**

create user boss identified by boss password expire;

grant dba,connect,resource to boss;

**创建其他角色**

create user staff1 identified by staff1;--进货经理

create user staff2 identified by staff2;--售货经理

create user staff3 identified by staff3;--进货员

create user staff4 identified by staff4;--售货员

**2.创建角色**

create role bmanager;--进货人员

create role smanager;--售货人员

**3.授予角色权限**

**----授予bmanager角色权限**

**授予bmanager登陆，查询，删除，更新boos的book表，stock表，stock\_infor视图的权限，和执行boss的所有过程的权利。**

grant connect to bmanager with admin option;--登陆

grant execute any procedure to bmanager;--执行所有过程

grant select,delete,update,insert on boss.book to bmanager;--操作book表

grant select,delete,update,insert on boss.stock to bmanager;--操作stock表

grant select,delete,update,insert on boss.stock\_infor to bmanager; --操作stock\_infor视图

**----授予smanager角色权限**

**授予smanager登陆，查询，删除，更新boos的book表，sold表，sold\_infor视图的权限**

**和执行boss的所有过程的权利。**

grant create session to smanager with admin option;--登陆

grant execute any procedure to smanager;--执行所有过程

grant select,delete,update,insert on boss.book to smanager;--操作book表

grant select,delete,update,insert on boss.sold to bmanager;--操作stock表

grant select,delete,update,insert on boss.sold\_infor to bmanager; --操作stock\_infor视图

**----将bmanager角色授予staff1和staff3**

**:<将bmanager角色授予进货经理staff1且可传递>:**

grant bmanager to staff1 with admin option;

**:<将bmanager角色授予进货员staff3不可传递>:**

grant bmanager to staff3;

**----将smanager角色授予staff2和staff4**

**:<将smanager角色授予售货经理staff2且可传递>:**

grant smanager to staff2 with admin option;

**:<将smanager角色授予售货员staff4不可传递>:**

grant smanager to staff4;

1. **复杂功能实现**
2. **查询book表并让其有序显示--表book**

**-------该语句可被所有用户使用**

set linesize 130

select \* from boss.book;

1. **顾客在购买某一本书之后，让店员推荐相同类型的图书（名称，类型，作者，单价）--表book**

**-------单行子查询--保存到d:\oracle\replace\_sql\slcx\_book.sql**

**-------该语句可被所有用户使用**

accept v\_bname prompt '输入书名后将显示与该书类型相同的书籍信息，请输入书名：'

select bookname,species,bookprice,author from book where species in(select distinct species from boss.book where bookname='《&v\_bname》');

执行：start d:\oracle\replace\_sql\slcx\_book.sql

1. **五周年店庆老板打算搞一个打折促销活动，其中，工具类九折，文学八折，IT七折，需要给销售人员折后价格表（编号，书名，单价，活动价，备注）--表book**

**-------decode**

**-------该语句可被所有用户使用**

select bookno,bookname,bookprice,species,decode(species,'工具类',bookprice\*0.9,'文学',bookprice\*0.8,'IT',bookprice\*0.7) "活动价",decode(species,'工具类','9折','文学','8折','IT','7折') "备注"

from boss.book

order by "备注";

1. **查询图书售卖情况（未售卖的图书售卖数量为0）--表book，表sold**

**-------该语句可被boss,seller manager,seller用户使用**

select b.bookno,b.bookname,b.species,b.bookprice,nvl(s.numbers,0),s.out\_time

from boss.book b,boss.sold s

where b.bookno=s.bookno(+)

order by s.out\_time;

1. **查询某年，月，日进货时每种类型的书花费多少钱？--视图stock\_infor**

**-------该语句可被boss,buyer manager,buyer用户使用**

select 类别,sum(总价)"进货总价"

from boss.stock\_infor

where to\_char("进货日期",'&v\_datetype')='&v\_date'

group by 类别;

1. **查询年，月，日的销售总额--内嵌视图（sold\_infor）**

**--保存d:\oracle\replace\_sql\se1\_sold\_infor.sql**

**-------该语句可被boss,seller manager,seller用户使用**

accept v\_datetype prompt '请输入您要查询的时间段类型年（yyyy）,月（yyyy-mm）,日（yyyy-mm-dd）：'

select sum(总价)"销售总额",售卖日期 from (select 总价,to\_char(售卖日期,'&v\_datetype') 售卖日期 from boss.sold\_infor) group by 售卖日期;

执行：start d:\oracle\replace\_sql\se1\_sold\_infor.sql

**7.查询按要求的属性求销售总额**

**--保存d:\oracle\replace\_sql\se2\_sold\_infor.sql**

**-------该语句可被boss,seller manager,seller用户使用**

accept v\_col prompt'请输入您要按(书名or类别or单价or售卖日期)查看销售总量：'

select sum("售卖数量")"总数量",sum("总价")"销售额",&&v\_col from boss.sold\_infor group by &v\_col;

执行：start d:\oracle\replace\_sql\se2\_sold\_infor.sql

**8.查询某天，某月，某年的盈利**

**--保存start d:\oracle\replace\_sql\se\_profit.sql**

**-------该语句可被boss用户使用**

accept v\_datetype prompt '请输入您要查询的盈利的时间段类型（yyyy）,月（yyyy-mm）,日（yyyy-mm-dd）：'

select sum(盈利),售卖日期 from (select 盈利,to\_char(售卖日期,'&v\_datetype') 售卖日期 from profit) group by 售卖日期 order by 售卖日期；

执行：start d:\oracle\replace\_sql\se\_profit.sql

**9.使用包功能：**

**-------该语句可被所有用户使用**

set serveroutput on;--打开控制台输出

**1)输入（图书编号，增/减，数量）实现增减库存**

exec boss.pk.xgkc('001002','-',1);

exec boss.pk.xgkc('001002','+',1);

**2)查询某类型所有图书的图书编号，图书名称和价格**

exec boss.pk.lxcx('IT');

**3)执行过程，输出提示库存少于5的图书信息方便进货**

exec boss.pk.jhtx;

**4)输入（员工姓名），求出员工年薪**

var yearsal number

call boss.pk.y\_sal('staff2') into:yearsal;

print yearsal

1. **Oracle体系结构**

**No1.**alter system flush shared\_pool;--清空共享池

**执行语句1：insert into boss.book values('001012','《红与黑》','文学',49.80,'吉林出版集团',to\_date('2011-10','yyyy-mm'),'司汤达',0)；**

已用时间: 00: 00: 00.12

统计信息

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

322 recursive calls 递归调用

9 db block gets

84 consistent gets 逻辑读

2 physical reads 物理读

1076 redo size

668 bytes sent via SQL\*Net to client

658 bytes received via SQL\*Net from client

4 SQL\*Net roundtrips to/from client

5 sorts (memory)

0 sorts (disk)

1 rows processed

**执行语句1： insert into boss.book values('001013','《小王子》','文学',19.80,'西苑出版社',to\_date('2008-09','yyyy-mm'),'吴群芳',0)；**

已用时间: 00: 00: 00.01

统计信息

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

8 recursive calls 递归调用

9 db block gets

3 consistent gets 逻辑读

0 physical reads 物理读

1004 redo size

673 bytes sent via SQL\*Net to client

656 bytes received via SQL\*Net from client

4 SQL\*Net roundtrips to/from client

1 sorts (memory)

0 sorts (disk)

1 rows processed

**解释：**执行上述语句1、2，在清空缓冲池的情况下，执行语句1时oracle需要对语句进行解析，还需从物理存储中读取数据，所以耗时较多；执行语句2时由于第一次对语句一做过解释，并且保存到共享池中，所以第二次执行时省去了对语句的解释过程，速度会相应加快！

**No2.**alter system flush shared\_pool;--清空共享池

**执行语句3：select count(\*) from book;**

已用时间: 00: 00: 00.10

统计信息

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

807 recursive calls

0 db block gets

166 consistent gets

0 physical reads

0 redo size

408 bytes sent via SQL\*Net to client

385 bytes received via SQL\*Net from client

2 SQL\*Net roundtrips to/from client

9 sorts (memory)

0 sorts (disk)

1 rows processed

**执行语句4：select count(\*) from book;**

已用时间: 00: 00: 00.00

统计信息

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

0 recursive calls

0 db block gets

1 consistent gets

0 physical reads

0 redo size

408 bytes sent via SQL\*Net to client

385 bytes received via SQL\*Net from client

2 SQL\*Net roundtrips to/from client

0 sorts (memory)

0 sorts (disk)

1 rows processed

**解释：**执行上述语句1、2，在清空缓冲池的情况下，执行语句1时oracle需要对语句进行解析，还需从物理存储中读取数据，所以耗时较多；执行语句2时由于第一次对语句一做过解释，并且保存到共享池中，而将读出的数据放入数据缓存区中，所以第二次执行时省去了对语句的解释过程，也省去了物理读，而是直接从数据缓存区读取数据，速度会相应加快！

**附录：**

**--进货人员工作流程**

如果所进书籍为新图书则从第1步开始，如果所进书籍为原有图书则从第2步开始：

**1.新增图书信息（新增图书时库存量应为0）**

start d:\oracle\replace\_sql\insert\_book.sql

**2.新增进货信息**

start d:\oracle\replace\_sql\insert\_stock.sql

**3.修改库存**

exec boss.pk.xgkc('图书编号','+',进货量);

**--售货人员工作流程**

**1.新增出货信息**

start d:\oracle\replace\_sql\insert\_sold.sql

**2.修改库存**

exec boss.pk.xgkc('图书编号','-',售卖数量);

### 十二.实验总结

完成该oracle数据实验基本掌握了数据库的建表操作，并能够创建出符合要求的表，将表与表之间建立联系。并且能够建立出表的视图，方便对图的查看。在进行备份和容灾操作的过程中，能够使用linux命令对数据库进行操作，极大地熟悉了linux的操作，同时也对oracle数据库的操作有了一定的了解。备份与容灾的操作是非常重要的，防止在数据库发生错误的时候，造成数据丢失，在进行这一实验过程的时候，利用了两台虚拟机，分别对主库与备库进行操作，在完成备份之后还进行了模拟删除数据，并成功恢复了数据。

同时在操作的过程中也应该更加认真仔细，在出现问题的时候应该要具备解决问题的能力，例如应该分析错误提示，找到错误的原因，并通过查询资料解决该问题。因为在数据库备份的过程中，等待时间较长，我们也应该耐心等待，不能够进行其他不利于数据库备份的操作。

在这次的实验过后，熟悉了oracle的基本命令，但是在oracle的学习中还远远不够，所以在学习oracle数据库的路程中还任重道远。