

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的网上售票的数据库设计 | | |
| 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 | 朱喜倩 | 学号 | 201810414205 |
| 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

[期末项目设计报告 1](#_Toc5391)

[2021 年 6 月 1 日 1](#_Toc9863)

[基于Oracle的网上售票管理系统数据库设计 3](#_Toc17579)

[实验要求 3](#_Toc29580)

[实验选题: 网上售票管理系统 3](#_Toc2095)

[实验内容 3](#_Toc28385)

[（一）自行设计一个信息系统的数据库项目，自拟 某项目 名称 3](#_Toc29613)

[（二）设计项目涉及的表及表空间使用方案 6](#_Toc1682)

[（三）备份恢复 27](#_Toc17976)

[实验总结 28](#_Toc18872)

**基于Oracle的网上售票管理系统数据库设计**

## 实验要求

1. 自行设计一个信息系统的数据库项目，自拟 某项目 名称。

2. 设计项目涉及的表及表空间使用方案。至少5张表和5万条数据，两个表空间。

3. 设计权限及用户分配方案。至少两类角色，两个用户。

4. 在数据库中建立一个程序包，在包中用PL/SQL语言设计一些存储过程和函数，

5. 实现比较复杂的业务逻辑，用模拟数据进行执行计划分析。

6. 设计自动备份方案或则手工备份方案。

7. 设计容灾方案。使用两台主机，通过DataGuard实现数据库整体的异地备份(可选)。

## 实验选题: 网上售票管理系统

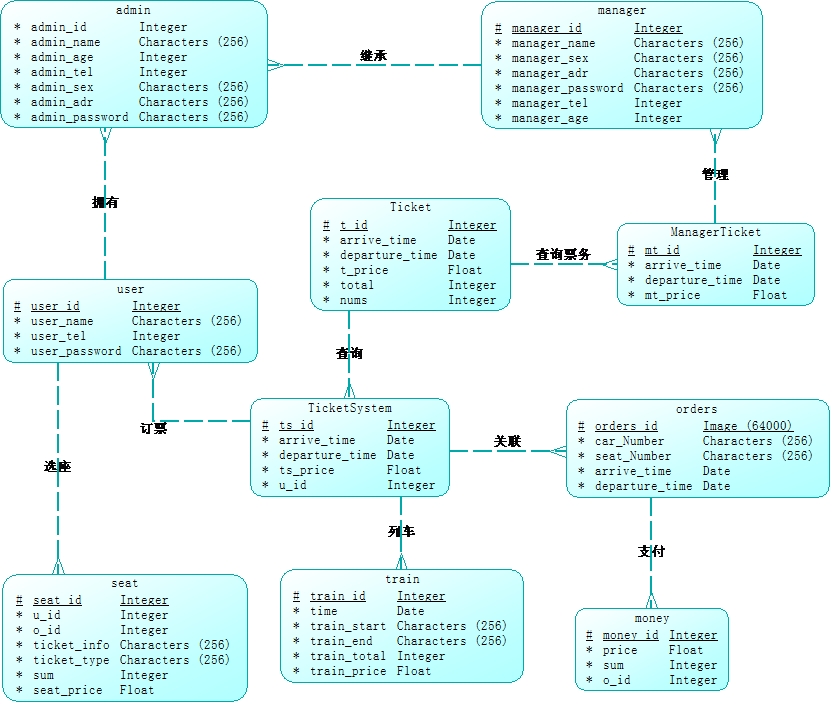
## 实验内容

### （一）自行设计一个信息系统的数据库项目，自拟 某项目 名称

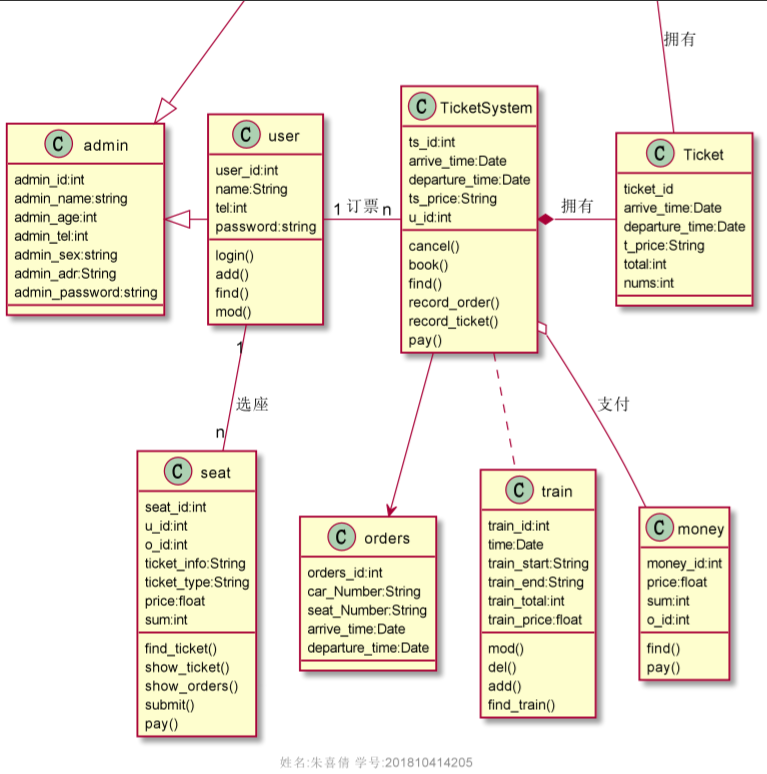
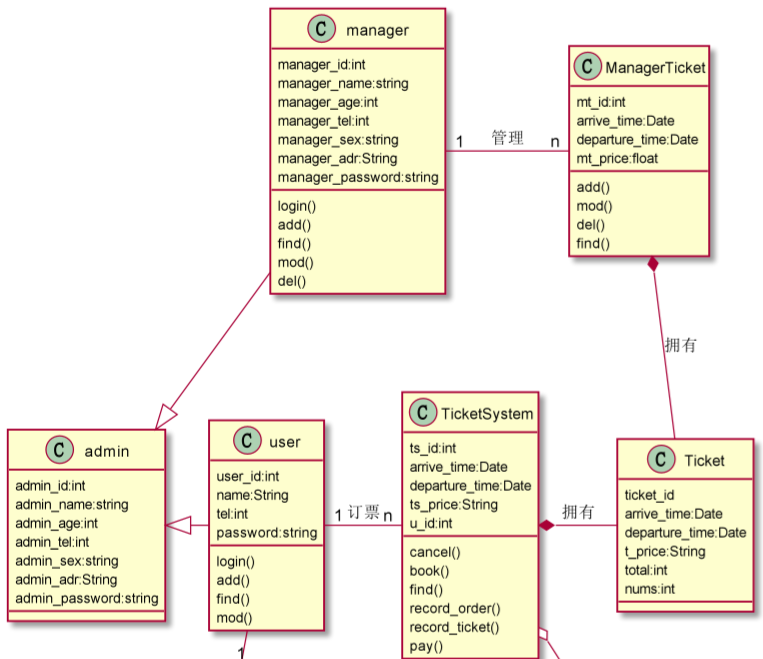
1. **描述**

网上售票管理系统，用户能在该系统完成查询、订票、改票、退票等操作。实验部分信息：admin，manager，user，ManagerTicket，Ticket，TicketSystem，seat，orders，train，money。

**2. E-R图设计**

****

1. **类图设计**

****

### （二）设计项目涉及的表及表空间使用方案

**1.设计数据表**

Admin

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **可以为空** | **注释** |
| ADMIN\_ID | NUMBER(6,0 | NO | adminID,主键 |
| ADMIN\_NAME | VARCHAR2(40 BYTE) | NO | admin名称，非空 |
| ADMIN\_AGE | NUMBER(6,0) | NO | admin年龄，非空 |
| ADMIN\_TEL | NUMBER(13,0) | NO | admin电话，非空 |
| ADMIN\_SEX | VARCHAR2(40 BYTE) | NO | admin性别，非空 |
| ADMIN\_ADR | VARCHAR2(40 BYTE) | NO | admin地址，非空 |
| ADMIN\_PASSWORD | VARCHAR2(40 BYTE) | NO | admin密码，非空 |

Manager

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **可以为空** | **注释** |
| MANAGER\_ID | NUMBER(6,0 | NO | 员工ID,主键 |
| MANAGER\_MANE | VARCHAR2(40 BYTE) | NO | 员工名字,不能为空，创建不唯一B树索引 |
| MANAGER\_AGE | NUMBER(6,0) | YES | 员工年龄 |
| MANAGER\_TEL | NUMBER(13,0) | YES | 员工电话号码 |
| MANAGER\_SEX | VARCHAR2(40 BYTE) | NO | 员工性别 |
| MANAGER\_ADR | VARCHAR2(40 BYTE) | NO | 员工地址 |
| MANAGER\_PASSWORD | VARCHAR2(40 BYTE) | NO | adminID |

User

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **可以为空** | **注释** |
| USER\_ID | NUMBER(6,0) | NO | 用户ID,主键 |
| USER\_MANE | VARCHAR2(40 BYTE) | NO | 用户名字,不能为空，创建不唯一B树索引 |
| USER\_AGE | NUMBER(6,0) | YES | 用户年龄 |
| USER\_TEL | NUMBER(13,0) | YES | 用户电话号码 |
| USER\_SEX | VARCHAR2(40 BYTE) | NO | 用户性别 |
| USER\_ADR | VARCHAR2(40 BYTE) | NO | 用户地址 |
| USER\_PASSWORD | VARCHAR2(40 BYTE) | NO | 用户密码 |

ManagerTicket

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **可以为空** | **注释** |
| MANAGERTICKET\_ID | NUMBER(6,0) | NO | 票务ID,主键 |
| MANAGERTICKET\_arrive\_time | VARCHAR2(40 BYTE) | NO | 列表到达时间 |
| MANAGERTICKET\_departure\_time | VARCHAR2(40 BYTE) | NO | 列车出发时间 |
| MANAGERTICKET\_PRICE | NUMBER(13,0) | NO | 票务票价,主键 |

Ticket

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **可以为空** | **注释** |
| TICKET\_ID | NUMBER(6,0) | NO | 票务详情ID,主键 |
| TICKET\_PRICE | VARCHAR2(40 BYTE) | NO | 票务价格 |
| ICKET\_arrive\_time | NUMBER(6,0) | YES | 列表到达时间 |
| TICKET\_departure\_time | NUMBER(13,0) | YES | 车出发时间 |
| TICKET\_TOTAL | VARCHAR2(40 BYTE) | NO | 票务总数量 |
| TICKET\_NUMS | NUMBER(8,2) | NO | 每类票务的数量 |

Orders

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **可以为空** | **注释** |
| ORDER\_ID | NUMBER(6,0) | NO | 订单ID,主键 |
| USER\_NAME | VARCHAR2(40 BYTE) | NO | 用户名称，B数索引 |
| USER\_TEL | NUMBER(6,0) | YES | 用户电话 |
| ORDER\_DATE | DATE | YES | 车出发时间 |
| MANAGER\_ID | VARCHAR2(40 BYTE) | NO | 订单经手人，员工表MANAGER的外键 |
| DISCOUNT | NUMBER(8,2) | NO | 订单整体优惠金额，默认为0 |
| TRADE\_RECEIVABLE | Number(8,2) | YES | 订单应收货款，默认为0 |

Train

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **可以为空** | **注释** |
| TRAIN\_ID | NUMBER(6,0) | NO | 列车详情ID,主键 |
| TRAIN\_TIME | DATE | NO | 列车每趟运行时长 |
| TRAIN\_start | DATE | YES | 列车出发时间 |
| TRAIN\_end | DATE | YES | 列车到达时间 |
| TRAIN\_total | NUMBER(8,2) | NO | 每趟列车的总数量 |
| TRAIN\_price | VARCHAR2(40 BYTE) | NO | 票价 |

Money

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **可以为空** | **注释** |
| MONEY\_ID | NUMBER(6,0) | NO | 支付详情ID,主键 |
| MONEY\_PRICE | NUMBER(10,0) | NO | 每次支付的价格，用户购买的票的票价 |
| MONEY\_SUM | NUMBER(10,0) | NO | 每次购买的票的数量 |
| ORDERS\_ID | NUMBER(6,0) | NO | 订单的ID作为外键 |

1. **创建表空间 pdbtest\_users1.dbf 与 pdbtest\_users2.dbf**

    CREATE TABLESPACE Users2

    DATAFILE

    '/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_users1.dbf'

    SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED,

    '/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_users2.dbf'

    SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

1. **创建用户YUKINO与DEEPSNOW**

分配表空间pdbtest\_users1.dbf 与 pdbtest\_users2.dbf的使用配额，再分配角色CONNECT和RESOURCE,便于用户可以连接到数据库来创建资源，最后再分配一个系统权限："CREATE VIEW"，便于用户可以创建视图

    --创建用户YUKINO

    CREATE USER YUKINO IDENTIFIED BY 123

    DEFAULT TABLESPACE "USERS"

    TEMPORARY TABLESPACE "TEMP";

    -- QUOTAS

    ALTER USER YUKINO QUOTA UNLIMITED ON USERS;

    ALTER USER YUKINO QUOTA UNLIMITED ON USERS02;

    -- ROLES

    GRANT "CONNECT" TO YUKINO WITH ADMIN OPTION;

    GRANT "RESOURCE" TO YUKINO WITH ADMIN OPTION;

    ALTER USER YUKINO DEFAULT ROLE "CONNECT","RESOURCE";

    -- SYSTEM PRIVILEGES

    GRANT CREATE VIEW TO YUKINO WITH ADMIN OPTION;

    --创建用户DEEPSNOW

    CREATE USER DEEPSNOW IDENTIFIED BY 123

    DEFAULT TABLESPACE "USERS"

    TEMPORARY TABLESPACE "TEMP";

    -- QUOTAS

    ALTER USER DEEPSNOW QUOTA UNLIMITED ON USERS;

    ALTER USER DEEPSNOW QUOTA UNLIMITED ON USERS02;

    -- ROLES

    GRANT "CONNECT" TO DEEPSNOW WITH ADMIN OPTION;

    GRANT "RESOURCE" TO DEEPSNOW WITH ADMIN OPTION;

    ALTER USER DEEPSNOW DEFAULT ROLE "CONNECT","RESOURCE";

    -- SYSTEM PRIVILEGES

    GRANT CREATE VIEW TO DEEPSNOW WITH ADMIN OPTION;

**4. 创建表**

创建ADMIN表

CREATE TABLE ADMIN

(

ADMIN\_ID NUMBER(6, 0) NOT NULL

, ADMIN\_NAME VARCHAR2(40 BYTE) NOT NULL

, CONSTRAINT ADMIN\_PK PRIMARY KEY

(

ADMIN\_ID

)

USING INDEX

(

CREATE UNIQUE INDEX DEPARTMENTS\_PK ON ADMIN (DEPARTMENT\_ID ASC)

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 2

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

)

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY NOPARALLEL;

创建MANAGER表

CREATE TABLE MANAGER

(

MANAGER\_ID NUMBER(6, 0) NOT NULL

, NAME VARCHAR2(40 BYTE) NOT NULL

, EMAIL VARCHAR2(40 BYTE)

, PHONE\_NUMBER VARCHAR2(40 BYTE)

, HIRE\_DATE DATE NOT NULL

, SALARY NUMBER(8, 2)

, MANAGER\_ID NUMBER(6, 0)

, MANAGER\_ID NUMBER(6, 0)

, PHOTO BLOB

, CONSTRAINT MANAGER\_PK PRIMARY KEY

(

EMPLOYEE\_ID

)

USING INDEX

(

CREATE UNIQUE INDEX EMPLOYEES\_PK ON EMPLOYEES (EMPLOYEE\_ID ASC)

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 2

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOPARALLEL

)

ENABLE

)

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NO INMEMORY

NOPARALLEL

LOB (PHOTO) STORE AS SYS\_LOB0000092017C00009$$

(

ENABLE STORAGE IN ROW

CHUNK 8192

NOCACHE

NOLOGGING

TABLESPACE USERS

STORAGE

(

INITIAL 106496

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

);

CREATE INDEX MANAGER\_INDEX1\_NAME ON MANAGER (NAME ASC)

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 2

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOPARALLEL;

ALTER TABLE MANAGER

ADD CONSTRAINT MANAGER\_FK1 FOREIGN KEY

(

DEPARTMENT\_ID

)

REFERENCES DEPARTMENTS

(

DEPARTMENT\_ID

)

ENABLE;

ALTER TABLE MANAGER

ADD CONSTRAINT MANAGER\_FK2 FOREIGN KEY

(

MANAGER\_ID

)

REFERENCES MANAGER

(

EMPLOYEE\_ID

)

ON DELETE SET NULL ENABLE;

ALTER TABLE MANAGER

ADD CONSTRAINT MANAGER\_CHK1 CHECK

(SALARY>0)

ENABLE;

ALTER TABLE MANAGER

ADD CONSTRAINT MANAGER\_CHK2 CHECK

(MANAGER\_ID<>MANAGER\_ID)

ENABLE;

ALTER TABLE MANAGER

ADD CONSTRAINT MANAGER\_MANAGER\_MANAGER\_ID CHECK

(USER\_ID<>MANAGER\_ID)

ENABLE;

ALTER TABLE MANAGER

ADD CONSTRAINTMANAGER\_SALARY CHECK

(SALARY>0)

ENABLE;

创建TICKET表

CREATE TABLE TICKET

(

TICKET\_NAME VARCHAR2(40 BYTE) NOT NULL

, TICKET\_TYPE VARCHAR2(40 BYTE) NOT NULL

, CONSTRAINT TICKET\_PK PRIMARY KEY

(

TICKET\_NAME

)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

ALTER TABLE TICKET

ADD CONSTRAINT TICKET\_CHK1 CHECK

(TICKET\_TYPE IN ('食品', '饮料', '生活用品'))

ENABLE;

创建ORDERS表及相关表, 表空间：分区表：USERS,USERS02

CREATE GLOBAL TEMPORARY TABLE "ORDER\_ID\_TEMP"

("ORDER\_ID" NUMBER(10,0) NOT NULL ENABLE,

CONSTRAINT "ORDER\_ID\_TEMP\_PK" PRIMARY KEY ("ORDER\_ID") ENABLE

) ON COMMIT DELETE ROWS ;

COMMENT ON TABLE "ORDER\_ID\_TEMP" IS '用于触发器存储临时ORDER\_ID';

CREATE TABLE ORDERS

(

ORDER\_ID NUMBER(10, 0) NOT NULL

, CUSTOMER\_NAME VARCHAR2(40 BYTE) NOT NULL

, CUSTOMER\_TEL VARCHAR2(40 BYTE) NOT NULL

, ORDER\_DATE DATE NOT NULL

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

, DISCOUNT NUMBER(8, 2) DEFAULT 0

, TRADE\_RECEIVABLE NUMBER(8, 2) DEFAULT 0

)

TABLESPACE USERS

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NOPARALLEL

PARTITION BY RANGE (ORDER\_DATE)

(

PARTITION PARTITION\_BEFORE\_2016 VALUES LESS THAN (TO\_DATE(' 2016-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 8388608

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION PARTITION\_BEFORE\_2017 VALUES LESS THAN (TO\_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 8388608

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

);

--创建本地分区索引ORDERS\_INDEX\_DATE：

CREATE INDEX ORDERS\_INDEX\_DATE ON ORDERS (ORDER\_DATE ASC)

LOCAL

(

PARTITION PARTITION\_BEFORE\_2016

TABLESPACE USERS

PCTFREE 10

INITRANS 2

STORAGE

(

INITIAL 8388608

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

, PARTITION PARTITION\_BEFORE\_2017

TABLESPACE USERS02

PCTFREE 10

INITRANS 2

STORAGE

(

INITIAL 8388608

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

)

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOPARALLEL;

CREATE INDEX ORDERS\_INDEX\_CUSTOMER\_NAME ON ORDERS (CUSTOMER\_NAME ASC)

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 2

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOPARALLEL;

CREATE UNIQUE INDEX ORDERS\_PK ON ORDERS (ORDER\_ID ASC)

GLOBAL PARTITION BY HASH (ORDER\_ID)

(

PARTITION INDEX\_PARTITION1 TABLESPACE USERS

NOCOMPRESS

, PARTITION INDEX\_PARTITION2 TABLESPACE USERS02

NOCOMPRESS

)

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 2

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOPARALLEL;

ALTER TABLE ORDERS

ADD CONSTRAINT ORDERS\_PK PRIMARY KEY

(

ORDER\_ID

)

USING INDEX ORDERS\_PK

ENABLE;

ALTER TABLE ORDERS

ADD CONSTRAINT ORDERS\_FK1 FOREIGN KEY

(

EMPLOYEE\_ID

)

REFERENCES EMPLOYEES

(

EMPLOYEE\_ID

)

ENABLE;

CREATE TABLE MANAGERTICKET

(

ID NUMBER(10, 0) NOT NULL

, MANAGERTICKET\_ID NUMBER(10, 0) NOT NULL

, MANAGERTICKET\_NAME VARCHAR2(40 BYTE) NOT NULL

, MANAGERTICKET\_NUM NUMBER(8, 2) NOT NULL

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

, CONSTRAINT MANAGERTICKET \_DETAILS\_FK1 FOREIGN KEY

(

ORDER\_ID

)

REFERENCES ORDERS

(

ORDER\_ID

)

ENABLE

)

TABLESPACE USERS

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS

NOPARALLEL

PARTITION BY REFERENCE (ORDER\_DETAILS\_FK1)

(

PARTITION PARTITION\_BEFORE\_2016

NOLOGGING

TABLESPACE USERS

--必须指定表空间，否则会将分区存储在用户的默认表空间中

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 8388608

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY,

PARTITION PARTITION\_BEFORE\_2017

NOLOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 8388608

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

)

;

CREATE UNIQUE INDEX ORDER\_DETAILS\_PK ON ORDER\_DETAILS (ID ASC)

NOLOGGING

TABLESPACE USERS

PCTFREE 10

INITRANS 2

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS UNLIMITED

BUFFER\_POOL DEFAULT

)

NOPARALLEL;

ALTER TABLE ORDER\_DETAILS

ADD CONSTRAINT ORDER\_DETAILS\_PK PRIMARY KEY

(

ID

)

USING INDEX ORDER\_DETAILS\_PK

ENABLE;

--这个索引可以使整个订单的详单存放在一起

CREATE INDEX ORDER\_DETAILS\_ORDER\_ID ON ORDER\_DETAILS (ORDER\_ID)

GLOBAL PARTITION BY HASH (ORDER\_ID)

(

PARTITION INDEX\_PARTITION1 TABLESPACE USERS

NOCOMPRESS

, PARTITION INDEX\_PARTITION2 TABLESPACE USERS02

NOCOMPRESS

);

ALTER TABLE ORDER\_DETAILS

ADD CONSTRAINT ORDER\_DETAILS\_PRODUCT\_NUM CHECK

(Product\_Num>0)

ENABLE;

创建3个触发器

CREATE OR REPLACE EDITIONABLE TRIGGER "ORDERS\_TRIG\_ROW\_LEVEL"

BEFORE INSERT OR UPDATE OF DISCOUNT ON "ORDERS"

FOR EACH ROW --行级触发器

declare

m number(8,2);

BEGIN

if inserting then

:new.TRADE\_RECEIVABLE := - :new.discount;

else

select sum(PRODUCT\_NUM\*PRODUCT\_PRICE) into m from ORDER\_DETAILS where ORDER\_ID=:old.ORDER\_ID;

if m is null then

m:=0;

end if;

:new.TRADE\_RECEIVABLE := m - :new.discount;

end if;

END;

/

--批量插入订单数据之前，禁用触发器

ALTER TRIGGER "ORDERS\_TRIG\_ROW\_LEVEL" DISABLE;

CREATE OR REPLACE EDITIONABLE TRIGGER "ORDER\_DETAILS\_ROW\_TRIG"

AFTER DELETE OR INSERT OR UPDATE ON ORDER\_DETAILS

FOR EACH ROW

BEGIN

--DBMS\_OUTPUT.PUT\_LINE(:NEW.ORDER\_ID);

IF :NEW.ORDER\_ID IS NOT NULL THEN

MERGE INTO ORDER\_ID\_TEMP A

USING (SELECT 1 FROM DUAL) B

ON (A.ORDER\_ID=:NEW.ORDER\_ID)

WHEN NOT MATCHED THEN

INSERT (ORDER\_ID) VALUES(:NEW.ORDER\_ID);

END IF;

IF :OLD.ORDER\_ID IS NOT NULL THEN

MERGE INTO ORDER\_ID\_TEMP A

USING (SELECT 1 FROM DUAL) B

ON (A.ORDER\_ID=:OLD.ORDER\_ID)

WHEN NOT MATCHED THEN

INSERT (ORDER\_ID) VALUES(:OLD.ORDER\_ID);

END IF;

END;

/

ALTER TRIGGER "ORDER\_DETAILS\_ROW\_TRIG" DISABLE;

CREATE OR REPLACE EDITIONABLE TRIGGER "ORDER\_DETAILS\_SNTNS\_TRIG"

AFTER DELETE OR INSERT OR UPDATE ON ORDER\_DETAILS

declare

m number(8,2);

BEGIN

FOR R IN (SELECT ORDER\_ID FROM ORDER\_ID\_TEMP)

LOOP

--DBMS\_OUTPUT.PUT\_LINE(R.ORDER\_ID);

select sum(PRODUCT\_NUM\*PRODUCT\_PRICE) into m from ORDER\_DETAILS

where ORDER\_ID=R.ORDER\_ID;

if m is null then

m:=0;

end if;

UPDATE ORDERS SET TRADE\_RECEIVABLE = m - discount

WHERE ORDER\_ID=R.ORDER\_ID;

END LOOP;

--delete from ORDER\_ID\_TEMP;

END;

/

ALTER TRIGGER "ORDER\_DETAILS\_SNTNS\_TRIG" DISABLE;

CREATE SEQUENCE "SEQ\_ORDER\_ID" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 1 CACHE 2000 ORDER NOCYCLE NOPARTITION ;

CREATE SEQUENCE "SEQ\_ORDER\_DETAILS\_ID" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 1 CACHE 2000 ORDER NOCYCLE NOPARTITION ;

CREATE OR REPLACE FORCE EDITIONABLE VIEW "VIEW\_ORDER\_DETAILS" ("ID", "ORDER\_ID", "CUSTOMER\_NAME", "CUSTOMER\_TEL", "ORDER\_DATE", "PRODUCT\_TYPE", "PRODUCT\_NAME", "PRODUCT\_NUM", "PRODUCT\_PRICE") AS

SELECT

d.ID,

o.ORDER\_ID,

o.CUSTOMER\_NAME,o.CUSTOMER\_TEL,o.ORDER\_DATE,

p.PRODUCT\_TYPE,

d.PRODUCT\_NAME,

d.PRODUCT\_NUM,

d.PRODUCT\_PRICE

FROM ORDERS o,ORDER\_DETAILS d,PRODUCTS p where d.ORDER\_ID=o.ORDER\_ID and d.PRODUCT\_NAME=p.PRODUCT\_NAME;

/

插入ADMIN，MANAGER数据

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (1,'总经办');

INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)

VALUES (1,'阳董事长',NULL,NULL,to\_date('2012-1-1','yyyy-mm-dd'),50000,NULL,1);

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (11,'销售部');

INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)

VALUES (11,'张总',NULL,NULL,to\_date('2012-1','yyyy-mm-dd'),50000,1,1);

INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)

VALUES (111,'陈经理',NULL,NULL,to\_date('2012-1','yyyy-mm-dd'),50000,11,11);

INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)

VALUES (112,'刘经理',NULL,NULL,to\_date('2012-1-1','yyyy-mm-dd'),50000,11,11);

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (12,'主管部');

INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)

VALUES (12,'赵总',NULL,NULL,to\_date('2012-1-1','yyyy-mm-dd'),50000,1,1);

INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)

VALUES (121,'朱经理',NULL,NULL,to\_date('2012-1-1','yyyy-mm-dd'),50000,12,12);

INSERT INTO EMPLOYEES(EMPLOYEE\_ID,NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)

VALUES (122,'李经理',NULL,NULL,to\_date('2012-1-1','yyyy-mm-dd'),50000,12,12);

insert into TICKET (TICKET\_name,TICKET\_type) values ('food1','火车');

insert into TICKET (TICKET\_name,TICKET\_type) values ('food2','高铁');

insert into TICKET (TICKET\_name,TICKET\_type) values ('food3','飞机');

insert into TICKET (TICKET\_name,TICKET\_type) values ('drinks1','北京');

insert into TICKET (TICKET\_name,TICKET\_type) values ('drinks2','杭州');

insert into TICKET (TICKET\_name,TICKET\_type) values ('drinks3','深圳');

insert into TRAIN (TRAIN\_name,TRAIN\_type) values ('articles of daily use1','G111');

insert into products (TRAIN\_name,TRAIN\_type) values ('articles of daily use2','T111');

insert into products (TRAIN\_name,TRAIN\_type) values ('articles of daily use3','G111');

批量插入订单数据，注意ORDERS.TRADE\_RECEIVABLE（订单应收款）的自动计算

declare

dt date;

m number(8,2);

V\_EMPLOYEE\_ID NUMBER(6);

v\_order\_id number(10);

v\_name varchar2(100);

v\_tel varchar2(100);

v number(10,2);

begin

for i in 1..50000

loop

if i mod 2 =0 then

dt:=to\_date('2015-3-2','yyyy-mm-dd')+(i mod 60);

else

dt:=to\_date('2016-3-2','yyyy-mm-dd')+(i mod 60);

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

v\_order\_id:=SEQ\_ORDER\_ID.nextval;

v\_name := 'aa'|| 'aa';

v\_name := 'Miku' || i;

v\_tel := '138923483' || i;

insert /\*+append\*/ into ORDERS (ORDER\_ID,CUSTOMER\_NAME,CUSTOMER\_TEL,ORDER\_DATE,EMPLOYEE\_ID,DISCOUNT)

values (v\_order\_id,v\_name,v\_tel,dt,V\_EMPLOYEE\_ID,dbms\_random.value(100,0));

--插入订单y一个订单包括3个商品

v:=dbms\_random.value(10000,4000);

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

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

values (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,2,v);

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

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

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

values (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,3,v);

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

v\_name:='articles of daily use'|| (i mod 3 + 1);

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

values (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,1,v);

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;

END IF;

end loop;

end;

/

ALTER TRIGGER "ORDERS\_TRIG\_ROW\_LEVEL" ENABLE;

ALTER TRIGGER "ORDER\_DETAILS\_SNTNS\_TRIG" ENABLE;

ALTER TRIGGER "ORDER\_DETAILS\_ROW\_TRIG" ENABLE;

最后动态增加一个PARTITION\_BEFORE\_2021分区：

ALTER TABLE ORDERS

ADD PARTITION PARTITION\_BEFORE\_2018 VALUES LESS THAN (TO\_DATE(' 2021-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN'));

ALTER INDEX ORDERS\_INDEX\_DATE

MODIFY PARTITION PARTITION\_BEFORE\_2018

NOCOMPRESS;

**5. 在数据库中建立一个程序包**

在包中用PL/SQL语言设计一些存储过程和函数，实现比较复杂的业务逻辑，用模拟数据进行执行计划分析。

create or replace PACKAGE SPM\_Pack IS

/\*

包SPM\_Pack中有：

一个函数:Get\_SaleAmount(V\_DEPARTMENT\_ID NUMBER)，

一个过程:Get\_Employees(V\_EMPLOYEE\_ID NUMBER)

\*/

FUNCTION Get\_SaleAmount(V\_DEPARTMENT\_ID NUMBER) RETURN NUMBER;

PROCEDURE Get\_Employees(V\_EMPLOYEE\_ID NUMBER);

END SPM\_Pack;

/

create or replace PACKAGE BODY SPM\_Pack IS

FUNCTION Get\_SaleAmount(V\_DEPARTMENT\_ID NUMBER) RETURN NUMBER

AS

N NUMBER(20,2);

BEGIN

SELECT SUM(O.TRADE\_RECEIVABLE) into N FROM ORDERS O,EMPLOYEES E

WHERE O.EMPLOYEE\_ID=E.EMPLOYEE\_ID AND E.DEPARTMENT\_ID =V\_DEPARTMENT\_ID;

RETURN N;

END;

PROCEDURE GET\_EMPLOYEES(V\_EMPLOYEE\_ID NUMBER)

AS

LEFTSPACE VARCHAR(2000);

begin

LEFTSPACE:=' ';

for v in

(SELECT LEVEL,EMPLOYEE\_ID,NAME,MANAGER\_ID FROM employees

START WITH EMPLOYEE\_ID = V\_EMPLOYEE\_ID

CONNECT BY PRIOR EMPLOYEE\_ID = MANAGER\_ID)

LOOP

DBMS\_OUTPUT.PUT\_LINE(LPAD(LEFTSPACE,(V.LEVEL-1)\*4,' ')||

V.EMPLOYEE\_ID||' '||v.NAME);

END LOOP;

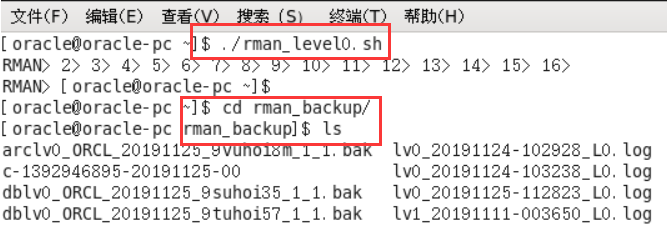
END;

END SPM\_Pack;

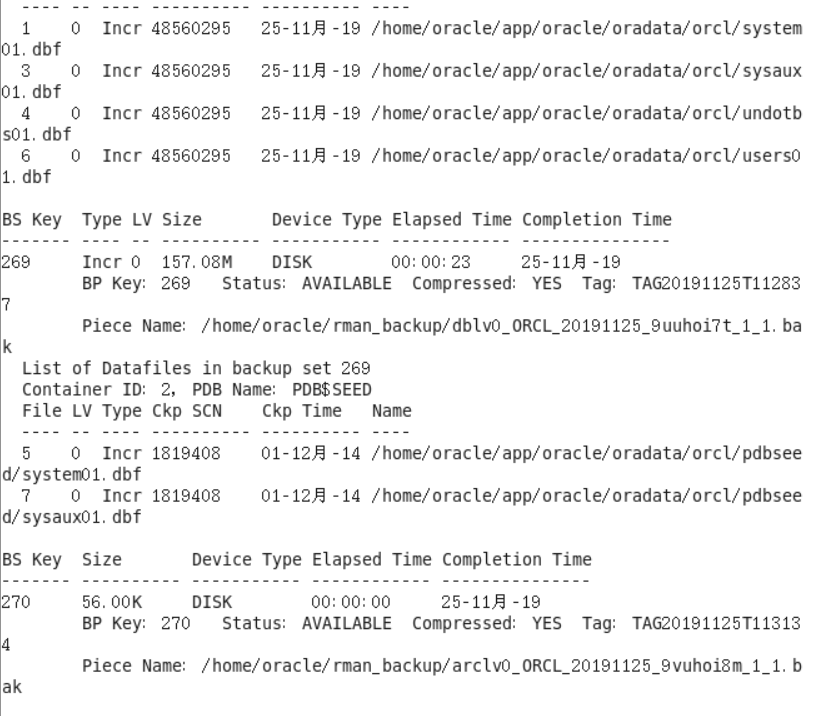
/

### （三）备份恢复

#### **1. 备份./rman\_level0.sh**



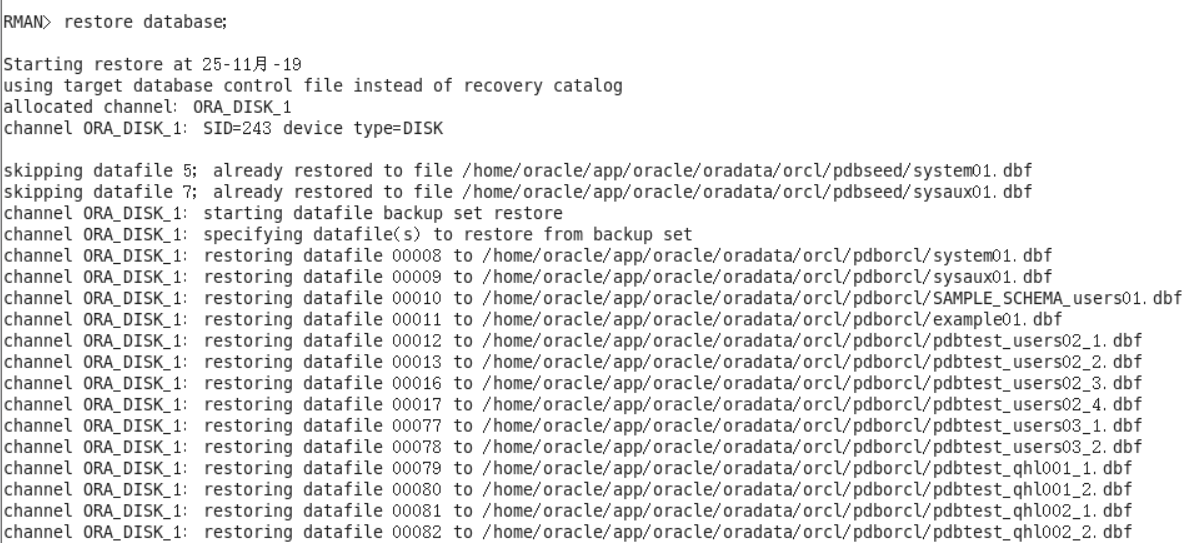
#### **查看备份内容**



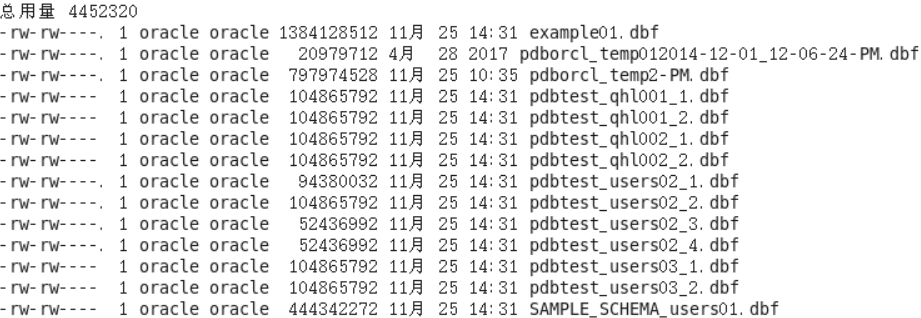
#### **删除数据**

pict15

#### **恢复备份**



#### **恢复完成**



## 实验总结

通过本次实验，我学会了如何分区，知道了分区表的创建，查询等基本操作。也了解了对于分区对象的查询可以仅搜索自己关心的分区，提高检索速度，如果表的某个分区出现故障，表在其他分区的数据仍然可用，因此具有更好的查询性能和可用性。同时，我也掌握分区表的创建方法，并且也熟悉了各种分区方式的使用场景。我了解PL/SQL语言结构，还了解PL/SQL变量和常量的声明和使用方法。同时我也学习了条件语句的使用方法、分支语句的使用方法、循环语句的使用方法，也了解到了常用的PL/SQL函数，并且也对其有更了进一步的认识，通过这次实验的锻炼，我也学习了包，过程，函数的用法。在以后的学习中，要学会学以致用。