

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的售票系统的数据库设计 | | |
| 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 | 甘雨凡 | 学号 | 201810414201 |
| 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](#_Toc28447)

[基于Oracle的售票系统的数据库设计 3](#_Toc23513)

[实验要求 3](#_Toc18210)

[实验选题: 火车票在线售票系统 3](#_Toc18923)

[实验内容 3](#_Toc23696)

[1. 自行设计一个信息系统的数据库项目 3](#_Toc22738)

[1.1 E-R图设计 4](#_Toc30697)

[1.2 类图设计 4](#_Toc3520)

[2. 设计项目涉及的表及表空间使用方案 5](#_Toc9944)

[2.1 设计数据表 5](#_Toc28654)

[2.2 创建表空间 pdbtest\_users02\_1.dbf 与 pdbtest\_users02\_2.dbf 6](#_Toc11008)

[2.3创建用户YUKINO与DEEPSNOW 7](#_Toc31864)

[2.4创建表 8](#_Toc28405)

[2.5 在数据库中建立一个程序包 23](#_Toc22021)

[3. 备份与回复 25](#_Toc19376)

[3.1备份./rman\_level0.sh 25](#_Toc30414)

[3.2 查看备份内容 25](#_Toc27402)

[3.3 恢复备份 26](#_Toc8128)

[3.4 恢复完成 26](#_Toc14029)

[4.实验总结 27](#_Toc21965)

# 基于Oracle的售票系统的数据库设计

# 实验要求

·自行设计一个信息系统的数据库项目，自拟 某项目 名称。

·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。

·设计权限及用户分配方案。至少两类角色，两个用户。

·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。

·设计自动备份方案或则手工备份方案。

·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)。

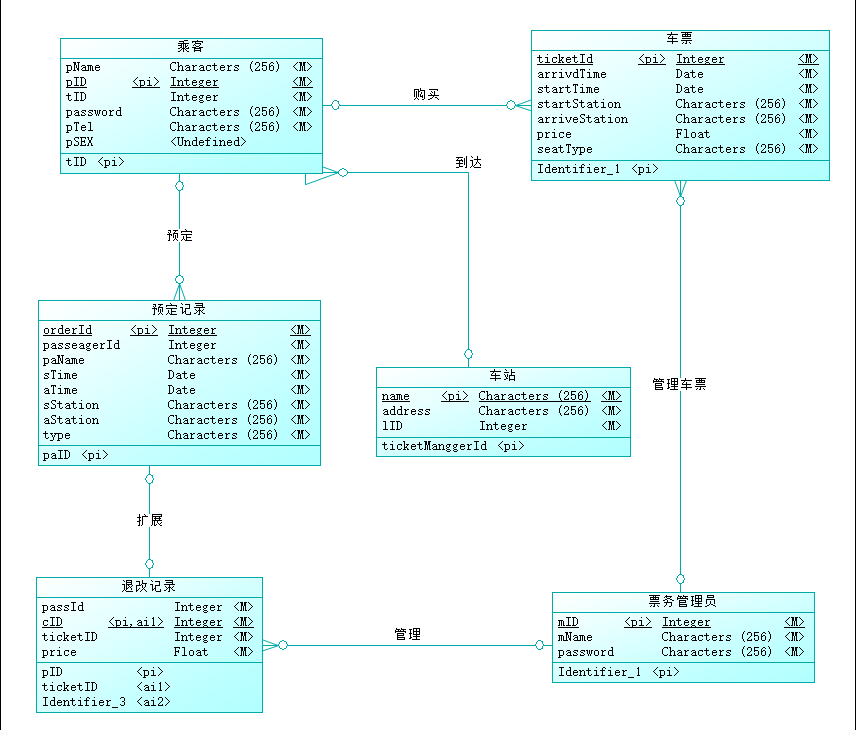
# 实验选题: 火车票在线售票系统

# 实验内容

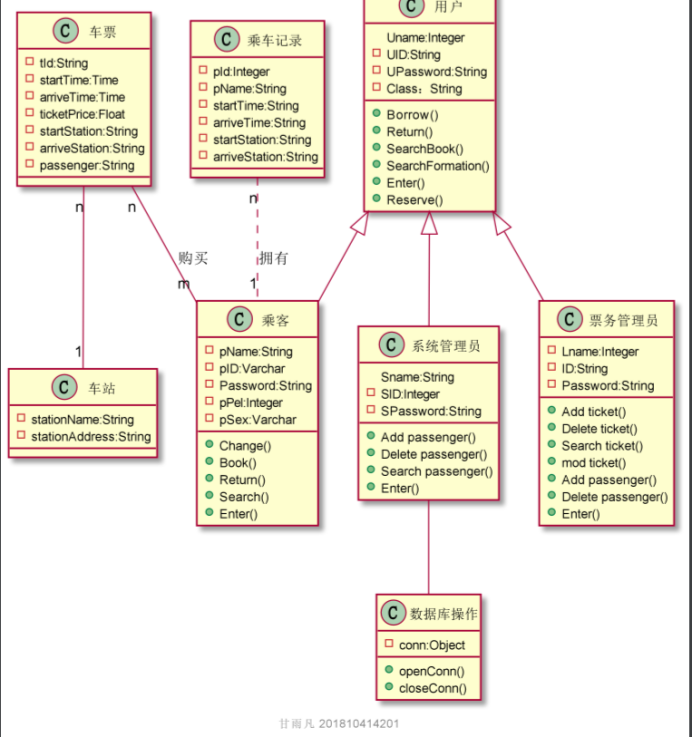
## 自行设计一个信息系统的数据库项目

​火车票代售系统，可以在线预定火车票。实验部分信息：乘客表，车票表，订单表，订单详单表，退改表。

### 1.1 E-R图设计

****

### 1.2 类图设计



## 设计项目涉及的表及表空间使用方案

### 2.1 设计数据表

1 车站信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表中列名 | 数据类型 | 是否为空 | 主键/外键 |
| Stationstyleno | varchar | 否 | 主键 |
| Stationstyle | varchar | 否 |  |

2 乘客信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表中列名 | 数据类型 | 是否为空 | 主键/外键 |
| Passerid | varchar | 否 | 主键 |
| Passname | varchar | 否 | 外键 |
| Pasersex | varchar | 否 |  |
| Pastype | varchar | 是 |  |

3车票信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表中列名 | 数据类型 | 是否为空 | 主键/外键 |
| Ticketid | Varchar | 否 | 主键 |
| Ticketname | Varchar | 否 |  |
| Ticketstyle | Varchar | 否 |  |
| Ticketauthor | Varchar | 否 |  |
| Ticketpub | Varchar | 是 |  |
| Ticketpubdate | Date | 是 |  |
| Ticketindate | Date | 是 |  |

4订单信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表中列名 | 数据类型 | 是否为空 | 主键/外键 |
| Orderid | Varchar | 否 | 外主键 |
| Orderid | Varchar | 否 | 外主键 |

5订单详情信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表中列名 | 数据类型 | 是否为空 | 主键/外键 |
| Ordername | Varchar | 否 | 外主键 |
| Orderid | Varchar | 否 | 外主键 |
| Orderdate | Date | 否 |  |

6退改记录信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 表中列名 | 数据类型 | 是否为空 | 主键/外键 |
| Orderid | varchar | 否 |  |
| Ordername | varchar | 否 |  |
| Ticketid | varchar | 否 | 外主键 |
| Price | varchar | 否 |  |

### 2.2 创建表空间 pdbtest\_users02\_1.dbf 与 pdbtest\_users02\_2.dbf

    CREATE TABLESPACE Users02

    DATAFILE

    '/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_users02\_1.dbf'

    SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED,

    '/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_users02\_2.dbf'

    SIZE 100M AUTOEXTEND ON NEXT 256M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

### 2.3创建用户YUKINO与DEEPSNOW

分配表空间pdbtest\_users02\_1.dbf 与 pdbtest\_users02\_2.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;

### 2.4创建表

创建DEPARTMENTS表

    CREATE TABLE DEPARTMENTS

    (

    DEPARTMENT\_ID NUMBER(6, 0) NOT NULL

    , DEPARTMENT\_NAME VARCHAR2(40 BYTE) NOT NULL

    , CONSTRAINT DEPARTMENTS\_PK PRIMARY KEY

    (

        DEPARTMENT\_ID

    )

    USING INDEX

    (

        CREATE UNIQUE INDEX DEPARTMENTS\_PK ON DEPARTMENTS (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EMPLOYEES表

    CREATE TABLE EMPLOYEES

    (

    EMPLOYEE\_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)

    , DEPARTMENT\_ID NUMBER(6, 0)

    , PHOTO BLOB

    , CONSTRAINT EMPLOYEES\_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 EMPLOYEES\_INDEX1\_NAME ON EMPLOYEES (NAME ASC)

    NOLOGGING

    TABLESPACE USERS

    PCTFREE 10

    INITRANS 2

    STORAGE

    (

    INITIAL 65536

    NEXT 1048576

    MINEXTENTS 1

    MAXEXTENTS UNLIMITED

    BUFFER\_POOL DEFAULT

    )

    NOPARALLEL;

    ALTER TABLE EMPLOYEES

    ADD CONSTRAINT EMPLOYEES\_FK1 FOREIGN KEY

    (

    DEPARTMENT\_ID

    )

    REFERENCES DEPARTMENTS

    (

    DEPARTMENT\_ID

    )

    ENABLE;

    ALTER TABLE EMPLOYEES

    ADD CONSTRAINT EMPLOYEES\_FK2 FOREIGN KEY

    (

    MANAGER\_ID

    )

    REFERENCES EMPLOYEES

    (

    EMPLOYEE\_ID

    )

    ON DELETE SET NULL ENABLE;

    ALTER TABLE EMPLOYEES

    ADD CONSTRAINT EMPLOYEES\_CHK1 CHECK

    (SALARY>0)

    ENABLE;

    ALTER TABLE EMPLOYEES

    ADD CONSTRAINT EMPLOYEES\_CHK2 CHECK

    (EMPLOYEE\_ID<>MANAGER\_ID)

    ENABLE;

    ALTER TABLE EMPLOYEES

    ADD CONSTRAINT EMPLOYEES\_EMPLOYEE\_MANAGER\_ID CHECK

    (MANAGER\_ID<>EMPLOYEE\_ID)

    ENABLE;

    ALTER TABLE EMPLOYEES

    ADD CONSTRAINT EMPLOYEES\_SALARY CHECK

    (SALARY>0)

    ENABLE;

创建PRODUCTS表

    CREATE TABLE PRODUCTS

    (

    PRODUCT\_NAME VARCHAR2(40 BYTE) NOT NULL

    , PRODUCT\_TYPE VARCHAR2(40 BYTE) NOT NULL

    , CONSTRAINT PRODUCTS\_PK PRIMARY KEY

    (

        PRODUCT\_NAME

    )

    ENABLE

    )

    LOGGING

    TABLESPACE "USERS"

    PCTFREE 10

    INITRANS 1

    STORAGE

    (

    INITIAL 65536

    NEXT 1048576

    MINEXTENTS 1

    MAXEXTENTS 2147483645

    BUFFER\_POOL DEFAULT

    );

    ALTER TABLE PRODUCTS

    ADD CONSTRAINT PRODUCTS\_CHK1 CHECK

    (PRODUCT\_TYPE IN ('车票'))

    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 ORDER\_DETAILS

    (

    ID NUMBER(10, 0) NOT NULL

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

    , PRODUCT\_NAME VARCHAR2(40 BYTE) NOT NULL

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

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

    , CONSTRAINT ORDER\_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DEPARTMENTS，TICKET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,'管理员',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,'管理员',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,'管理员',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,'管理员',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,'管理员',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,'管理员',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,'管理员',NULL,NULL,to\_date('2012-1-1','yyyy-mm-dd'),50000,12,12);

   insert into products (product\_name,product\_type) values ('ticket','车票');

insert into products (product\_name,product\_type) values ('ticket','车票');

insert into products (product\_name,product\_type) values ('ticket','车票');

insert into products (product\_name,product\_type) values ('ticket','车票');

insert into products (product\_name,product\_type) values ('ticket','车票');

insert into products (product\_name,product\_type) values ('ticket','车票');

insert into products (product\_name,product\_type) values ('articles of daily use1');

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

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

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;

### 2.5 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;

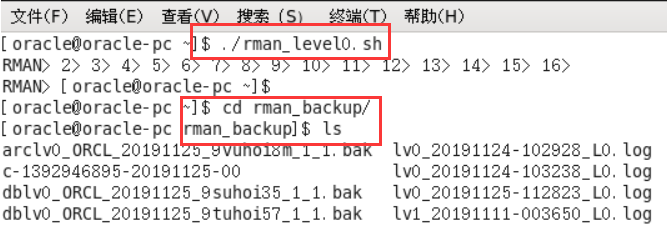
        END;

    END SPM\_Pack;

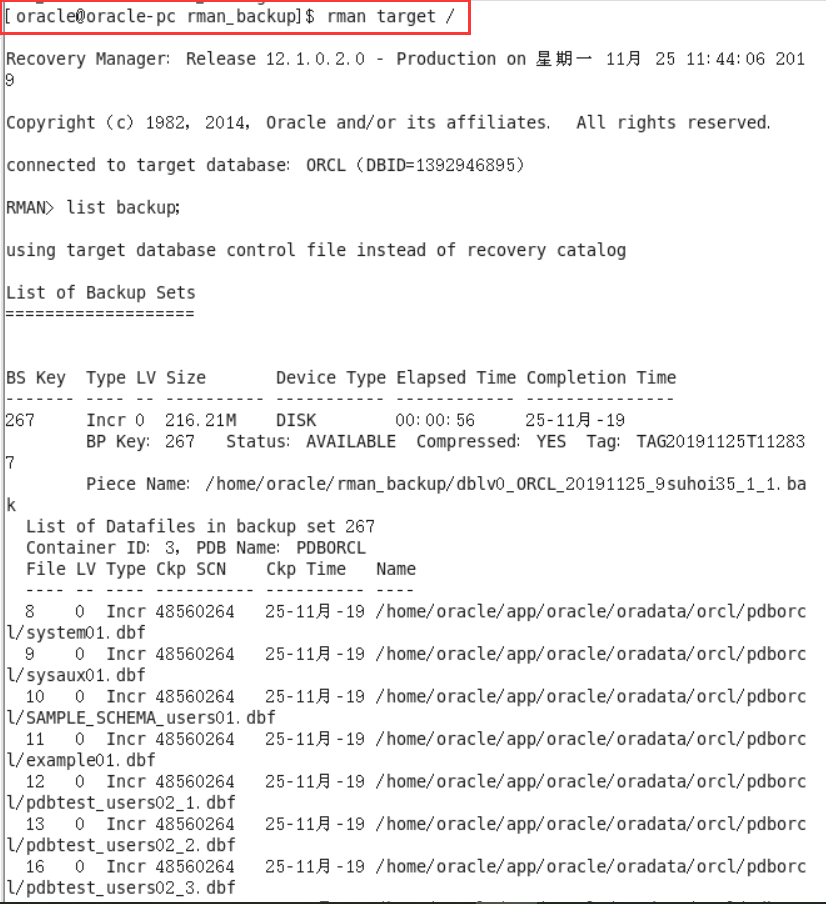
    /

## 3. 备份与回复

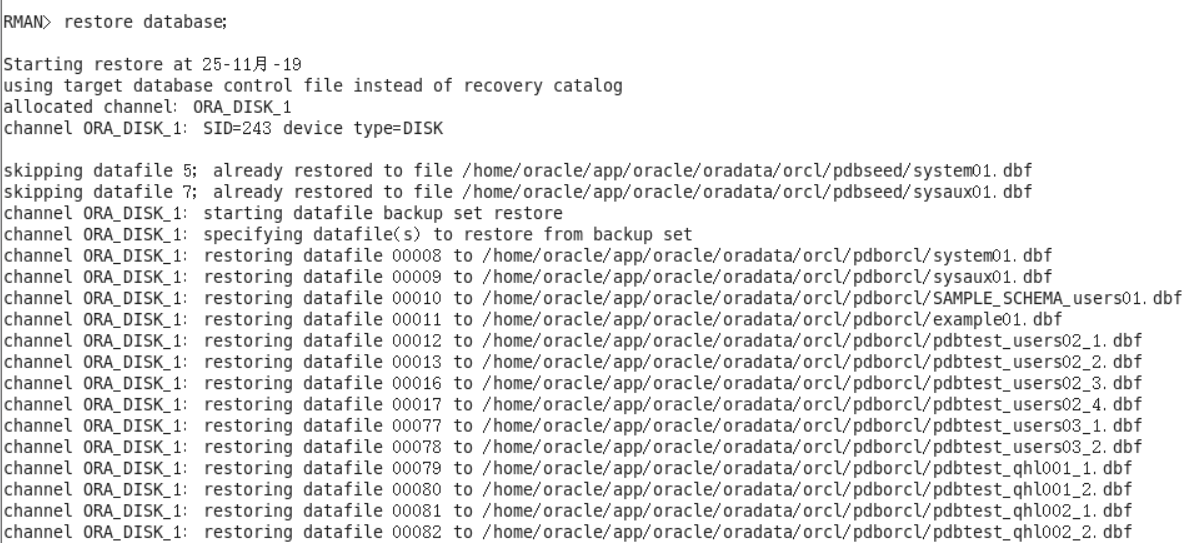
### 3.1备份./rman\_level0.sh

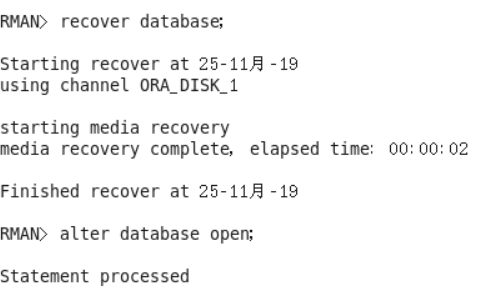


### 3.2 查看备份内容

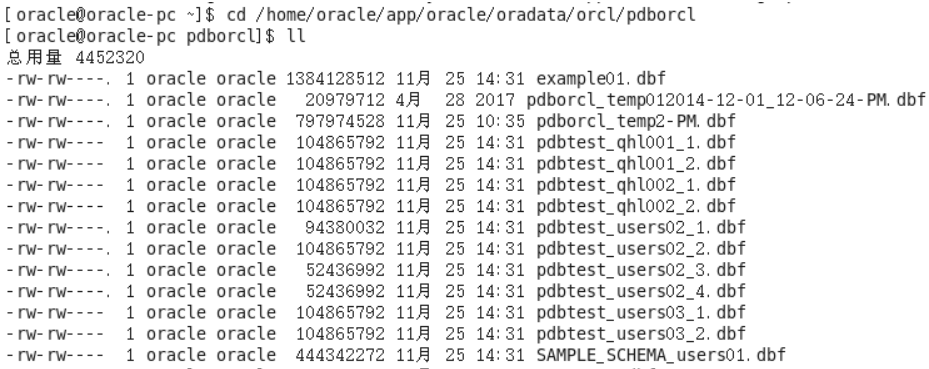


### 3.3 恢复备份



****

### 3.4 恢复完成



## 

## 4.实验总结

通过完成本次实验，我收获了很多。我了解Oracle表和视图的概念，学会了使用SQL语句Create Table创建表以及Select语句插入，修改，删除以及查询数据，还有使用SQL语句创建视图，学习部分存储过程和触发器的使用。以及我了解PL/SQL语言结构和PL/SQL变量和常量的声明和使用方法；学习并掌握条件语句、分支语句、循环语句的使用方法；以及常用的PL/SQL函数，和包，过程，函数的用法。