

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的网上超市的数据库设计 | | |
| 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 | 阳天昊 | 学号 | 201810414226 |
| 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](#_Toc13740)

[2021 年 6 月 1 日 1](#_Toc5430)

[实验要求 3](#_Toc1206)

[实验选题 3](#_Toc19473)

[实验内容 3](#_Toc18288)

[1.自行设计一个信息系统的数据库项目 3](#_Toc7792)

[2. 设计项目涉及的表及表空间使用方案 5](#_Toc17862)

[3. 设计备份方案:物理备份(使用rman工具备份及恢复,全程在命令行执行) 25](#_Toc29619)

[实验总结 27](#_Toc17230)

**基于Oracle的某项目数据库设计**

**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: 网上超市管理系统**

**实验内容**

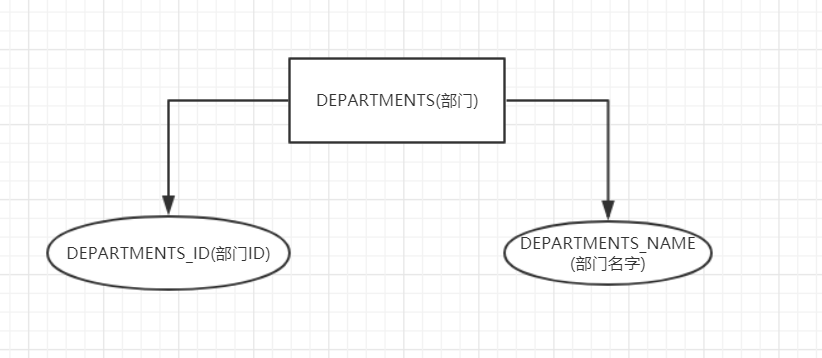
**1.自行设计一个信息系统的数据库项目**

​某网上超市， 该超市能接受网上订单进行产品的销售。实验部分信息：员工表，部门表，订单表，订单详单表，商品表。

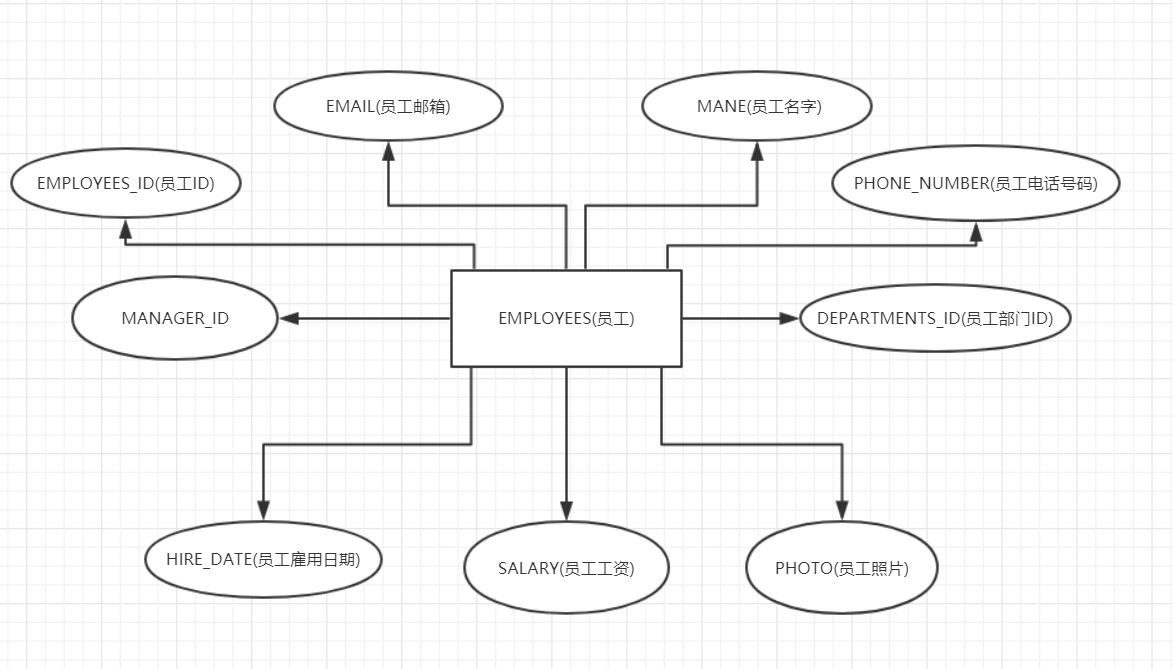
**1.1 E-R图设计**

根据场景分析一共有3个实体：部门、员工、商品

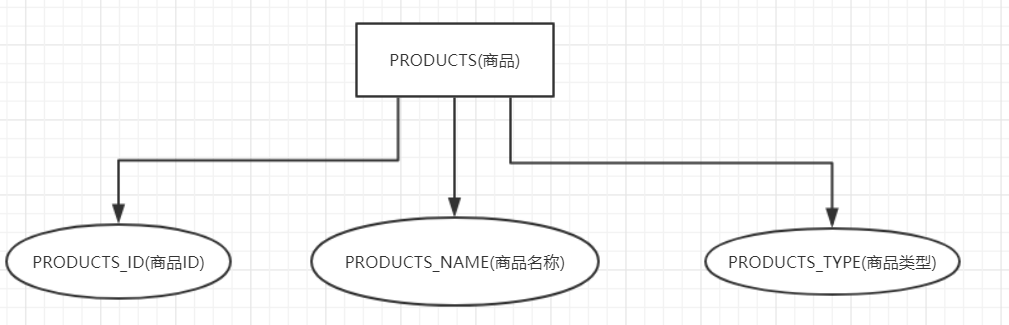
部门



员工

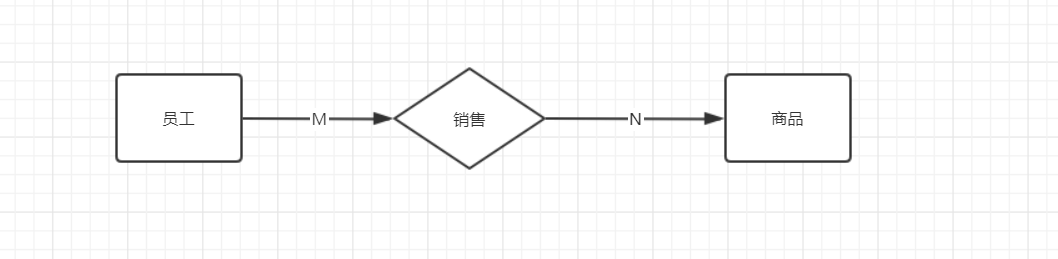


商品

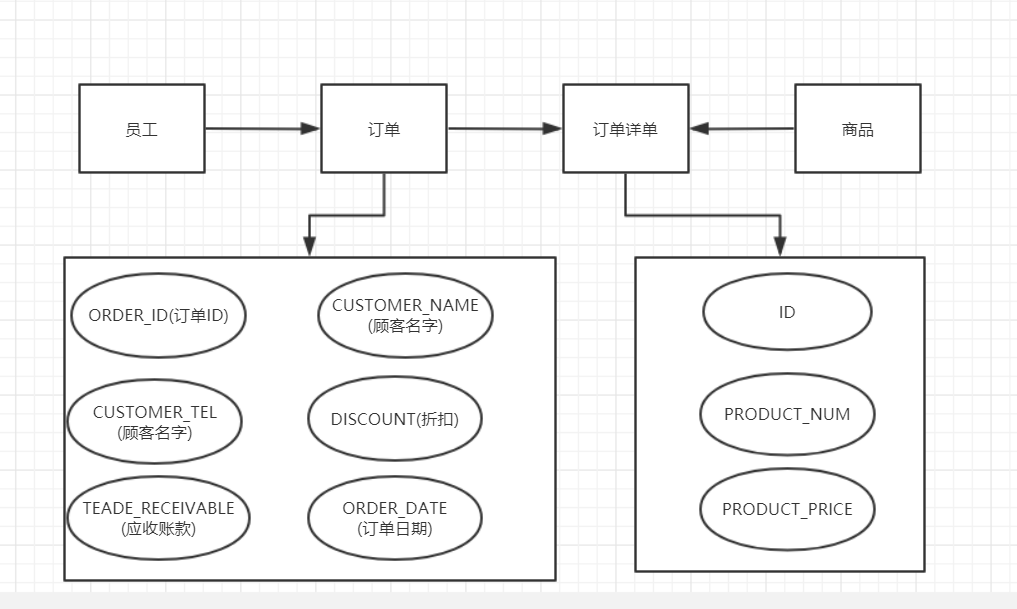


联系模型：订单表、订单详表

1

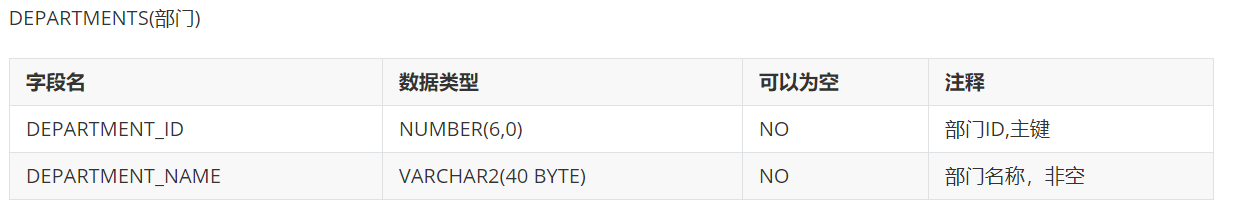


2



1. **设计项目涉及的表及表空间使用方案**

**2.1设计数据表**







![32YDX8DLL82]0CR](@]KBTB](data:image/png;base64,)

**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 ('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 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，EMPLOYEES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 products (product\_name,product\_type) values ('food1','食品');

    insert into products (product\_name,product\_type) values ('food2','食品');

    insert into products (product\_name,product\_type) values ('food3','食品');

    insert into products (product\_name,product\_type) values ('drinks1','饮料');

    insert into products (product\_name,product\_type) values ('drinks2','饮料');

    insert into products (product\_name,product\_type) values ('drinks3','饮料');

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

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

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

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. 设计备份方案:物理备份(使用rman工具备份及恢复,全程在命令行执行)**

    说明：

    　　1. RMAN是 ORACLE提供的一个备份与恢复的工具，可以用来执行完全或不完全的数据库恢复。

    　　2. RMAN不能用于备份初始化参数文件和口令文件。

    　　3. 与传统工具相比，RMAN具有独特的优势：跳过未使用的数据块。当备份一个RMAN备份集时，RMAN不会备份从未被写入的数据块，而传统的方式无法获知那些是未被使用的数据块。

    　　4. RMAN可以进行增量备份（增量备份是针对于上一次备份（无论是哪种备份）：备份上一次备份后，所有发生变化的文件）

**3.1首先需要在将要备份的数据库服务器上切换服务器的归档模式**

    # su oracle //切换到oracle用户

    $ sqlplus /nolog //启动sqlplus

    SQL> conn / as sysdba //以DBA的身份连接数据库

    SQL> shutdown immediate; //立即关闭数据库

    SQL> startup mount //启动实例并加载数据库，但不打开

    SQL> alter database archivelog; //更改数据库为归档模式

    SQL> alter database open; //打开数据库

    SQL> alter system archive log start; //启用自动归档

    SQL> exit //退出

**3.2 启动并连接到rman，使用sys用户**

    $ rman target=sys/oracle@orcl

**3.3 基本设置**

具体路径可自己安装需求改，但是登录的用户需要对备份的目录有读写权限，目录建立方法可参考上面逻辑备份的备份目录建立

    RMAN> configure default device type to disk; //设置默认的备份设备为磁盘

    RMAN> configure device type disk parallelism 2; //设置备份的并行级别，通道数

    RMAN> configure channel 1 device type disk format '/home/oracle/oracle\_bak/bakup\_%U'; //设置备份的文件格式，只适用于磁盘设备

    RMAN> configure channel 2 device type disk format '/home/oracle/oracle\_bak/bakup2\_%U'; //设置备份的文件格式，只适用于磁盘设备

    RMAN> configure controlfile autobackup on; //打开控制文件与服务器参数文件的自动备份

    RMAN> configure controlfile autobackup format for device type disk to '/home/oracle/oracle\_bak/ctl\_%F'; //设置控制文件与服务器参数文件自动备份的文件格式

    说明：

        format:

        %c：备份片的拷贝数(从1开始编号)；

        %d：数据库名称；

        %D：位于该月中的天数(DD)；

        %M：位于该年中的月份(MM)；

        %F：一个基于DBID唯一的名称，这个格式的形式为c-xxx-YYYYMMDD-QQ,其中xxx位该数据库的DBID，YYYYMMDD为日期，QQ是一个1-256的序列；

        %n：数据库名称，并且会在右侧用x字符进行填充，使其保持长度为8；

        %u：是一个由备份集编号和建立时间压缩后组成的8字符名称。利用%u可以为每个备份集产生一个唯一的名称；

        %p：表示备份集中的备份片的编号，从1开始编号；

        %U：是%u\_%p\_%c的简写形式，利用它可以为每一个备份片段(既磁盘文件)生成一个唯一的名称，这是最常用的命名方式；

        %t：备份集时间戳;

        %T:年月日格式(YYYYMMDD);

**3.4 查看所有配置与数据库方案报表**

    RMAN> show all

    RMAN> report schema;

**3.5 全量备份数据库**

**3.5.1 全量备份全库**

    RMAN> backup database plus archivelog delete input;

**3.5.2 备份表空间**

这里也可以改为备份某个表空间，比如 back tablespace users;就是备份users的表空间

    RMAN> backup tablespace system plus archivelog delete input;

**3.5.3 备份归档日志**

    RMAN> backup archivelog all delete input;

**3.5.4 复制数据文件**

    RMAN> copy datafile 1 to '/home/oracle/oracle\_bak/bak/system.copy';

**3.5.5 查看备份和文件副本、查看复制文件**

    RMAN> list backup;

    RMAN> list copy

**3.5.6 验证备份**

    RMAN> validate backupset 3;

可以在list backup命令的结果中查看备份集的编号

**实验总结**

通过本次实验我复习了表空间、用户的创建，数据表的设计与创建。还复习了数据的插入，与使用PL/SQL语言设计一些存储过程和函数的代码编写。同时还学习到了如何通过物理备份，来备份数据，让我受益匪浅。