

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的餐厅管理系统的数据库设计 | | |
| 课程 | Oracle数据库应用 | | |
| 学 院 | 信息科学与工程学院 | | |
| 专 业 | 软件工程 | 年级 | 2018级 |
| 学生姓名 | 胡西凌 | 学号 | 201810414210 |
| 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

**实验选题：餐厅管理系统**

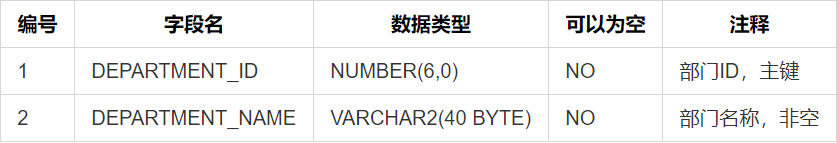
随着计算机的迅速发展，计算机被广泛的应用到各个领域，餐厅行业对软件的功能需求也日益增高。本系统就是针对此种情况和客户需求而开发。

**实验步骤**:

**设计项目涉及的表及表空间使用方案**

表结构

部门表DEPARTMENTS



产品表PRODUCTS



员工表EMPLOYEES



订单表ORDERS



订单详单表ORDER\_DETAILS



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

```sql

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;

```

创建用户USER01与USER02

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

```sql

    --创建用户USER01

    CREATE USER USER01 IDENTIFIED BY 123

    DEFAULT TABLESPACE "USERS"

    TEMPORARY TABLESPACE "TEMP";

    -- QUOTAS

    ALTER USER USER01 QUOTA UNLIMITED ON USERS;

    ALTER USER USER01 QUOTA UNLIMITED ON USERS02;

    -- ROLES

    GRANT "CONNECT" TO USER01 WITH ADMIN OPTION;

    GRANT "RESOURCE" TO USER01 WITH ADMIN OPTION;

    ALTER USER USER01 DEFAULT ROLE "CONNECT","RESOURCE";

    -- SYSTEM PRIVILEGES

    GRANT CREATE VIEW TO USER01 WITH ADMIN OPTION;

    --创建用户USER02

    CREATE USER USER02 IDENTIFIED BY 123

    DEFAULT TABLESPACE "USERS"

    TEMPORARY TABLESPACE "TEMP";

    -- QUOTAS

    ALTER USER USER02 QUOTA UNLIMITED ON USERS;

    ALTER USER USER02 QUOTA UNLIMITED ON USERS02;

    -- ROLES

    GRANT "CONNECT" TO USER02 WITH ADMIN OPTION;

    GRANT "RESOURCE" TO USER02 WITH ADMIN OPTION;

    ALTER USER USER02 DEFAULT ROLE "CONNECT","RESOURCE";

    -- SYSTEM PRIVILEGES

    GRANT CREATE VIEW TO USER02 WITH ADMIN OPTION;

```

创建表DEPARTMENTS和EMPLOYEES

```sql

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

--  DDL for Table DEPARTMENTS

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

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;

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

--  DDL for Table EMPLOYEES

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

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

  )

);

```

为EMPLOYEES表创建索引

```sql

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;

```

为EMPLOYEES表的DEPARTMENT\_ID字段添加外键约束EMPLOYEES\_FK1，引用DEPARTMENTS部门表的DEPARTMENT\_ID字段

```sql

ALTER TABLE EMPLOYEES

ADD CONSTRAINT EMPLOYEES\_FK1 FOREIGN KEY

(

  DEPARTMENT\_ID

)

REFERENCES DEPARTMENTS

(

  DEPARTMENT\_ID

)

ENABLE;

```

为EMPLOYEES表的MANAGER\_ID字段添加外键约束EMPLOYEES\_FK2，引用EMPLOYEES部门表的EMPLOYEE\_ID字段，在删除EMPLOYEES表中的数据时，引用表EMPLOYEES的EMPLOYEE\_ID字段置为null

```sql

ALTER TABLE EMPLOYEES

ADD CONSTRAINT EMPLOYEES\_FK2 FOREIGN KEY

(

  MANAGER\_ID

)

REFERENCES EMPLOYEES

(

  EMPLOYEE\_ID

)

ON DELETE SET NULL ENABLE;

```

 为EMPLOYEES添加检查约束

```sql

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;

```

创建表PRODUCTS，并且添加检查约束

```sql

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 ('主食', '凉菜', '小吃'))

ENABLE;

```

创建临时表ORDER\_ID\_TEMP和表ORDERS

```sql

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

--  DDL for Table ORDER\_ID\_TEMP

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';

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

--  DDL for Table ORDERS

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

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

```sql

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;

```

 创建索引ORDERS\_INDEX\_CUSTOMER\_NAME

```sql

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;

```

创建唯一索引ORDERS\_PK

```sql

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;

```

为ORDERS表添加主键约束和外键约束

```sql

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;

```

 创建ORDER\_DETAILS表

```sql

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

)

;

```

为ORDER\_DETAILS表的ID字段创建唯一索引ORDER\_DETAILS\_PK

```sql

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;

```

为ORDER\_DETAILS添加主键约束ORDER\_DETAILS\_PK

```sql

ALTER TABLE ORDER\_DETAILS

ADD CONSTRAINT ORDER\_DETAILS\_PK PRIMARY KEY

(

  ID

)

USING INDEX ORDER\_DETAILS\_PK

ENABLE;

```

为ORDER\_DETAILS创建全局索引ORDER\_DETAILS\_ORDER\_ID

```sql

--这个索引是必须的，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

);

```

为ORDER\_DETAILS添加检查约束Product\_Num>0

```sql

ALTER TABLE ORDER\_DETAILS

ADD CONSTRAINT ORDER\_DETAILS\_PRODUCT\_NUM CHECK

(Product\_Num>0)

ENABLE;

```

创建触发器

```sql

--创建3个触发器

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

--  DDL for Trigger ORDERS\_TRIG\_ROW\_LEVEL

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

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;

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

--  DDL for Trigger ORDER\_DETAILS\_ROW\_TRIG

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

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;

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

--  DDL for Trigger ORDER\_DETAILS\_SNTNS\_TRIG

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

  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; --这句话很重要，否则可能一直不释放空间，后继插入会非常慢。

END;

/

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

```

创建两个序列号SEQ\_ORDER\_ID、SEQ\_ORDER\_DETAILS\_ID

```sql

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

--  DDL for Sequence

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

CREATE SEQUENCE SEQ\_ORDER\_ID INCREMENT BY 1 START WITH 1 MAXVALUE 999999999 MINVALUE 1 CACHE 20 ORDER;

CREATE SEQUENCE SEQ\_ORDER\_DETAILS\_ID INCREMENT BY 1 START WITH 1 MAXVALUE 999999999 MINVALUE 1 CACHE 20 ORDER;

```

创建视图VIEW\_ORDER\_DETAILS

```sql

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

--  DDL for View VIEW\_ORDER\_DETAILS

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

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

```sql

--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,'李董事长',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,NULL,1);

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (11,'销售部1');

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

  VALUES (11,'张总',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,1,1);

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

  VALUES (111,'吴经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,11,11);

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

  VALUES (112,'白经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,11,11);

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (12,'销售部2');

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

  VALUES (12,'王总',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,1,1);

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

  VALUES (121,'赵经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,12,12);

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

  VALUES (122,'刘经理',NULL,NULL,to\_date('2010-1-1','yyyy-mm-dd'),50000,12,12);

insert into products (product\_name,product\_type) values ('computer1','电脑');

insert into products (product\_name,product\_type) values ('computer2','电脑');

insert into products (product\_name,product\_type) values ('computer3','电脑');

insert into products (product\_name,product\_type) values ('phone1','手机');

insert into products (product\_name,product\_type) values ('phone2','手机');

insert into products (product\_name,product\_type) values ('phone3','手机');

insert into products (product\_name,product\_type) values ('paper1','耗材');

insert into products (product\_name,product\_type) values ('paper2','耗材');

insert into products (product\_name,product\_type) values ('paper3','耗材');

```

批量插入订单数据

```sql

--批量插入订单数据，注意ORDERS.TRADE\_RECEIVABLE（订单应收款）的自动计算,注意插入数据的速度

--2千万条记录，插入的时间是：18100秒（约5小时）

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..10000

  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; --应该将SEQ\_ORDER\_ID.nextval保存到变量中。

    v\_name := 'aa'|| 'aa';

    v\_name := 'zhang' || i;

    v\_tel := '139888883' || 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 (SEQ\_ORDER\_DETAILS\_ID.NEXTVAL,v\_order\_id,v\_name,2,v);

    v:=dbms\_random.value(1000,50);

    v\_name:='paper'|| (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:='phone'|| (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);

    --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;

  --统计用户的所有表，所需时间很长：2千万行数据，需要1600秒，该语句可选

  --dbms\_stats.gather\_schema\_stats(User,estimate\_percent=>100,cascade=> TRUE); --estimate\_percent采样行的百分比

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\_2021 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\_2021

NOCOMPRESS;

```

在数据库中建立一个程序包

```sql

create or replace PACKAGE MyPack IS

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

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

END MyPack;

/

create or replace PACKAGE BODY MyPack IS

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

  AS

    N NUMBER(20,2); --注意，订单ORDERS.TRADE\_RECEIVABLE的类型是NUMBER(8,2),汇总之后，数据要大得多。

    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

      --通过LEVEL判断递归的级别

      LEFTSPACE:=' ';

      --使用游标

      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 MyPack;

```

**设计备份方案:物理备份**

 冷备份

冷备份发生在数据库已经正常关闭的情况下，当正常关闭时会提供给我们一个完整的数据库。冷备份时将关键性文件拷贝到另外的位置的一种说法。对于备份Oracle信息而言，冷备份时最快和最安全的方法。冷备份的优点是：

１、 是非常快速的备份方法（只需拷文件）

２、 容易归档（简单拷贝即可）

３、 容易恢复到某个时间点上（只需将文件再拷贝回去）

４、 能与归档方法相结合，做数据库“最佳状态”的恢复。

５、 低度维护，高度安全。

```sql

--（1） 关闭数据库

sqlplus /nolog

sql>connect /as sysdba

sql>shutdown normal;

--（2） 用拷贝命令备份全部的数据文件、重做日志文件、控制文件、初始化参数文件

sql>cp

--（3） 重启Oracle数据库

sql>startup

```

 热备份

热备份是在数据库运行的情况下，采用archivelog mode方式备份数据库的方法。所以，如果你有昨天夜里的一个冷备份而且又有今天的热备份文件，在发生问题时，就可以利用这些资料恢复更多的信息。热备份要求数据库在Archivelog方式下操作，并需要大量的档案空间。一旦数据库运行在archivelog状态下，就可以做备份了。

热备份的优点是：

1． 可在表空间或数据库文件级备份，备份的时间短。

2． 备份时数据库仍可使用。

3． 可达到秒级恢复（恢复到某一时间点上）。

4． 可对几乎所有数据库实体做恢复 。

5． 恢复是快速的，在大多数情况下可以再数据库仍工作时恢复。

① 数据文件一个表空间一个表空间的备份。

（1） 设置表空间为备份状态

（2） 备份表空间的数据文件

（3） 回复表空间为正常状态

② 备份归档log文件

（1） 临时停止归档进程

（2） 备份下那些在archive redo log目录中的文件

（3） 重新启动archive进程

（4） 备份归档的redo log文件

③ 用alter database backup controlfile命令来备份控制文件

**导入和导出**

利用Export可将数据从数据库中提取出来，利用Import则可将提取出来的数据送回到Oracle数据库中去。

支持类型： table, user， tablespace， full database

```sql

exp system/manager@lm\_pdborcl  file=d:\export.dmp full=y--将数据库lm\_pdborcl完全导出（全库备份）,用户名LinMing 密码123456 导出到D:\daochu.dmp中

exp system/manager@lm\_pdborcl file=d:\export.dmp owner=(DEPARTMENTS,EMPLOYEES,PRODUCTS,ORDERS,ORDERS\_DETAILS,ORDERS\_ID\_TEMP)--将数据库lm\_pdborcl下DEPARTMENTS,EMPLOYEES,PRODUCTS,ORDERS,ORDERS\_DETAILS,ORDERS\_ID\_TEMP

的表导出

exp aichannel/aichannel@lm\_pdborcl file= d:export.dmp tables= (DEPARTMENTS,EMPLOYEES,PRODUCTS,ORDERS,ORDERS\_DETAILS,ORDERS\_ID\_TEMP)--将表DEPARTMENTS,EMPLOYEES,PRODUCTS,ORDERS,ORDERS\_DETAILS,ORDERS\_ID\_TEMP导出

```

<!--注：若用户权限不够时需要修改权限-->

<!--grant exp\_full\_database, imp\_full\_database to username;-->

**实验总结**

通过本次实验，我复习了之前学习的对象管理和PL\SQL编程相关知识，同时对这两部分有了新的理解，除此之外还学到了备份的方法，认识到了备份的重要性