

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题目 | 基于Oracle的果蔬电商系统数据库的设计 | | |
| 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 | 仝若凡 | 学号 | 201710414417 |
| 指导教师 | 赵卫东 | 职称 | 副教授 |

|  |  |  |  |
| --- | --- | --- | --- |
| **评分项** | **评分标准** | **满分** | **得分** |
| 文档整体 | 文档内容详实、规范，美观大方 | 10 |  |
| 表设计 | 表，表空间设计合理，数据合理 | 20 |  |
| 用户管理 | 权限及用户分配方案设计正确 | 10 |  |
| PL/SQL设计 | 存储过程和函数设计正确 | 25 |  |
| 备份方案 | 备份方案设计正确 | 25 |  |
| 容灾方案 | DataGuard设计正确 | 10 |  |
| **得分合计** | | |  |

2021年6月12日

1 总述

* 1. 编写目的

随着科学的不断进步,计算机应用己经遍布整个社会的每个角落。计算机在商业管理上的应用,使其逐步系统化,完善化。利用计算机帮助企业高效率完成销售管理的日常事务,是适应现代企业制度要求,也是推动企业管理走向科学化,规范化的必要条件。由于销售管理是一项琐碎、复杂而又十分细致的工作,在销售的各个环节中不允许出错,实行手工操作,就会浪费大量的人力资源和资金、时间,而销售系统的使用将可以避免以上的难题,不但能够保证单价,账目核算准确无误,而且还可以利用该系统对有关销售的各种信息进行统计,服务于财务部门和销售部门的业务处理流程的本文档通过对果蔬批发进销存工作流程的深入了解及进行了可行性分析后,对果蔬电商管理系统做了需求分析,功能模块划分,数据库的设计与实现。

* 1. 背景

现在网络购物现象发展迅速,受网上购买衣服、玩具等没有保质期限制的物品影响,蔬菜果蔬这类有保质期限制的物品也开始紧跟潮流,逐渐的向着网上进行购买发展,基于此网上购买瓜果蔬菜便成了人们向往的事情。这样的创新尝试将推进网络购物的一个新领域的发展,本系统正是为了应对这一需求而设计的。果蔬网购是近年才在中国流行,其中最具代表地位的果蔬电商商是飞果网、百森果蔬网等进口品牌企业。在电子商务网站中,网上购买瓜果蔬菜是目前应用最人性化的范例之一。给人们的日常生活尤其是现在工作忙碌的年轻人提供了便利。本系统就是针对网络购买蔬菜果蔬进行设计的,一个以义乌农贸城为背景,进行网络销售蔬菜果蔬的一个系统,这个系统以Oracle 12c为工具,使用SQLDeveloper数据库进行数据的管理,通过Hibemet与数据库进行交互,通过需求分析设计实现了用户注册、登录、身份验证及用户数据的采集、瓜果蔬菜的浏览,通过浏览进行选择,也可以有针对性的进行查询和选择性的购买等功能,基本上能够满足对蔬菜果蔬网络的购买的需求。

1. 外部设计
   1. 环境说明

Oracle数据库系统是美国Oracle公司（甲骨文）提供的以分布式数据库为核心的一组软件产品，是目前最流行的客户/服务器（client/server）或B/S体系结构的数据库之一，比如SilverStream就是基于数据库的一种中间件。Oracle数据库是目前世界上使用最为广泛的数据库管理系统，作为一个通用的数据库系统，它具有完整的数据管理功能；作为一个关系型数据库，它是一个完备关系的产品；作为分布式数据库它实现了分布式处理功能，但它的所有知识，只要在一种机型上学习了Oracle知识，便能在各种类型的机器上使用它。

Oracle SQL Developer是Oracle公司出品的一个免费的集成开发环境。是一个免费非开源的用以开发数据库应用程序的图形化工具，使用 SQL Developer 可以浏览数据库对象、运行 SQL 语句和脚本、编辑和调试 PL/SQL 语句。另外还可以创建执行和保存报表。该工具可以连接任何 Oracle 9.2.0.1 或者以上版本的 Oracle 数据库，支持 Windows、Linux 和 Mac OS X 系统。Oracle SQL Developer是针对Oracle数据库的交互式开发环境（IDE）。

Oracle SQL Developer简化了Oracle数据库的开发和管理。 SQL Developer提供了PL/SQL程序的端到端开发，运行查询工作表的脚本，管理数据库的DBA控制台，报表接口，完整的数据建模的解决方案，并且能够支持将你的第三方数据库迁移至Oracle。

SQL Developer可以连接到任何Oracle 10g及其后续版本的数据库，并且能在Windows，Linux和Mac OSX上运行。

最新版本的Oracle SQL Developer提供了PL / SQL单元测试，集成了数据模型浏览器和Subversion源代码版本控制系统，CVS（并行版本系统）此外，2.1版包括了许多更新功能，如SQL格式化，模式比较，复制、导出向导和迁移支持。

1. 数据库逻辑设计

本系统的应用场景是模拟一个企业销售其商品的过程。该企业的销售活动通过订单进行,企业员工要销售产品给客户,必须要新开一张订单,在订单中记录销售过程的关键信息。比如哪个客户购买了哪些产品,订单的经手员工是谁,何时销售的,订单的应收款是多少等等。

* 1. 实体模型

根据应用场景分析,共有3个原始的实体 Entity),它们是部门、员工和产品部门( DEPARTMENTS):部门包括部门D( DEPARTMENT ID)和部门名称DEPARTMENT NAME),如图3-1。

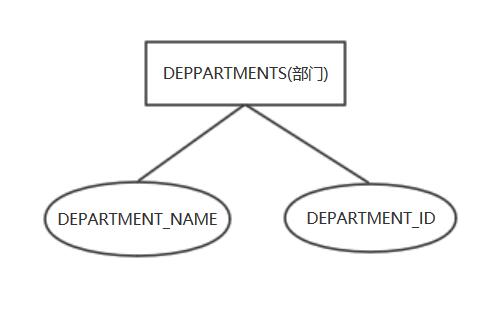


图 3-1 DEPARTMENTS实体

员工 EMPLOYEES):员工包括员工 ID(EMPLOYEE ID),姓名（NAME),照片（POTO）工资 (SALARY)等。员工的属性中还应包括员工所属的部门 ID(DEPARTMENT\_ID),部门ID不能为空,表示员工必须属于某一个部门。员工的属性中还有员工的上司( MANAGER\_ ID)该属性可以为空,表示没有上司。员工的实体如图3-2。

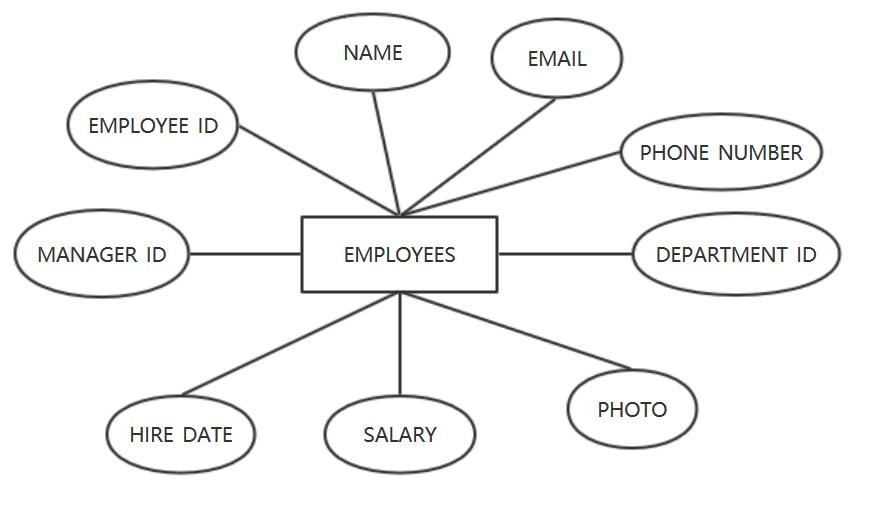


图 3-2 EMPLOYEES实体

产品( PRODUCTS):产品包括3个属性:产品1D( PRODUCT\_ID),产品名称PRODUCT\_NAME)和产品类型( PRODUCT\_TYPE),见图3-3

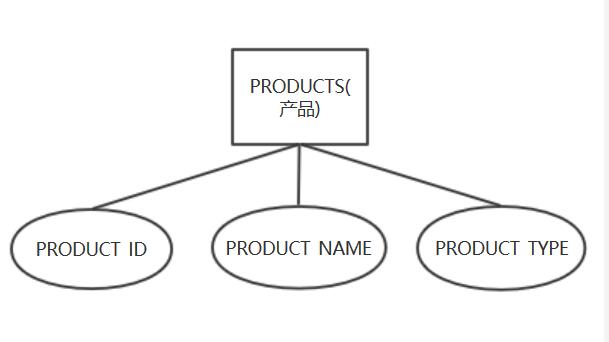


图3-3 PRODUCTS实体

* 1. 实体联系模型

企业员工的工作是销售产品,因此员工和产品之间就有一个“销售”的联系,如图3-4所示,员工与产品之间的关系是多对多的关系,如图3-4。

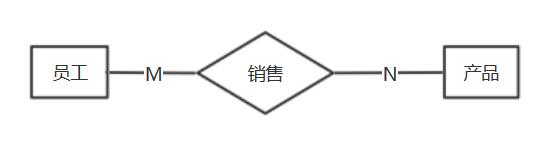


图3-4员工与产品销售关系简图

考虑到销售活动中有一些重要属性,比如折扣,客户信息,销售时间,产品的销售数量和销售价格,我们把销售关系细分为订单和订单详单两个实体,订单中存储:订单ID( ORDER\_ID)、折扣 (DISCOUNT)、客户信息( CUSTOMER\_NAME, CUSTOMER\_TEL)订单时间( ORDER\_DATE)以及应收货款总额( TRADE\_RECEIVABLE),订单详单中存储订单详单的ID,以及订单中的全部产品信息,包括:销售数量( PRODUCT\_NUM)和销售价格( PRODUCT\_PRICE),见图3-5。

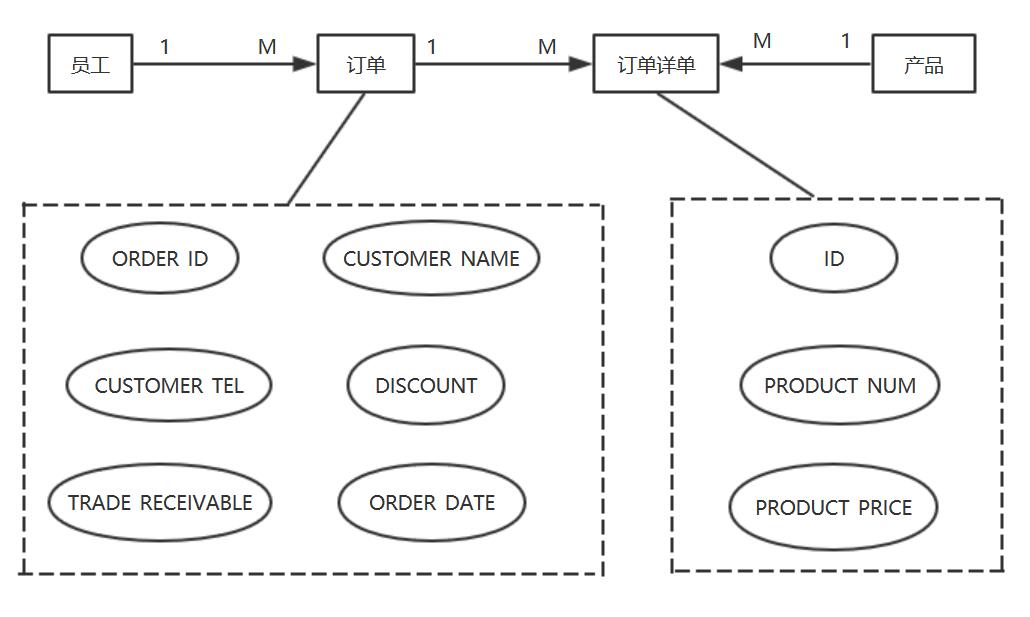


图3-5员工的订单关系

1. 表结构设计

E.R模型建立好以后,就可以设计 Oracle的关系表了,在独立实体中找出主要属性设置为主键,比如在产品表中,产品名称( PRODUCT\_NAME)是主键。由关系派生出的实体中要加入外键关系,比如在图3-5中有两个一对多的关系,需要增加外键属性,即在订单表中增加员工D属性( EMPLOYEE\_ID),在订单详单中增加产品ID属性(PRODUCT \_ID)。部门表( DEPARTMENTS)包括(DEPARTMENT\_ID)和(DEPARTMENT\_NAME)两个属性,见表4-1。

表4\_1部门表 DEPARTNTS

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| DEPARTMENT\_ID | NUMBER(6,0) | NO | 部门ID，主键 |
| DEPARTMENT\_NAME | VARCHAR2(40 BYTE) | NO | 部门名称，非空 |

产品表 PRODUCTS包括产品DD( PRODUCT\_ID)、产品名称( PRODUCT\_NAME)和产品类型( PRODUCT\_TYPE),见表4-2。注意产品表中的产品类型只能取值:西瓜，猕猴桃，菠萝。

表4\_2 产品表 PRODUCTS

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| PRODUCT\_ID | VARCHAR2(40 BYTE) | NO | 产品ID号，产品表的主键 |
| PRODUCT\_NAME | VARCHAR2(40 BYTE) | NO | 产品名称 |
| PRODUCT\_TYPE | VARCHAR2(40 BYTE) | NO | 产品类型只能取值:西瓜，猕猴桃，菠萝 |

员工表 EMPLOYEES包括员工的属性,要注意（MANAGER ID）是员工的上司,是员工表(EMPOLYEE\_ID)的外键, (MANAGER\_ID)不能等于(EMPLOYEE\_ID）,即员工的领导不能是自己。主键删除时(MANAGER\_ID)设置为空值。见表5-3。订单表 ORDERS是订货信息,见表5-4.表中(TRADE\_RECEIVABLE)是订单的应收货款,是从订单详单表中自动汇总过来的。计算公式是: Trade\_Receivables=sum(订单详单表.Product\_Num\*订单详单表.Product\_Price)- Discount。

表4\_3员工表 EMPLOYEES

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| EMPLOYEE\_ID | NUMBER(6,0) | NO | 员工ID，员工表的主键 |
| NAME | VARCHAR2(40 BYTE) | NO | 员工姓名，不能为空，创建不唯一B树索引 |
| EMAIL | VARCHAR2(40 BYTE) | YES | 电子信箱 |
| PHONE\_NUMBER | VARCHAR2(40 BYTE) | YES | 电话 |
| HIRE\_DATE | DATE | NO | 雇佣日期 |
| SALARY | NUMBER(8,2) | YES | 月薪，必须大于零 |
| MANAGER\_ID | NUMBER(6,0) | YES | 员工的上司，员工表EMPLOYEE\_ID的外键，MANAGER\_ID不能等于EMPLOYEE\_ID |
| DEPARTMENT\_ID | NUMBER(6,0) | YES | 员工所在部门，是部门表DEPARTMENTS的外键 |
| PHOTO | BLOB | YES | 员工照片 |

表 4-4 订单表ORDERS

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ORDER\_ID | NUMBER(10,0) | NO | 订单编号,主键。值来自于序列:SEQ\_ORDER\_ID |
| CUSTOMER\_NAME | VARCHAR2(40 BYTE) | NO | 客户名称,B树索引 |
| CUSTOMER\_TEL | VARCHAR2(40 BYTE) | NO | 客户电话 |
| ORDER\_DATE | DATE | NO | 订货日期,应该采用分区存储方式 |
| EMPLOYEE\_ID | NUMBER(6,0) | NO | 订单经手人，员工表EMPLOYEES的外键 |
| DISCOUNT | NUMBER(8,2) | YES | 订单整体优惠金额。默认值为0 |
| Trade\_Receivable | NUMBER(8,2) | YES | 订单应收货款，默认为0 |

订单详单表（ORDER\_DETAILS）包含订单中全部产品的信息,见表5-5。要注意,件系统要通过触发器将产品的金额汇总到订单表中的(TRADE\_RECEIVABLE)属性。

表 4-5 订单详单表ORDER\_DETAILS

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ID | NUMBER(10,0) | NO | 本表的主键，值来自于序列：SEQ\_ORDER\_DETAILS\_ID |
| ORDER\_ID | NUMBER(10,0) | NO | 所属的订单号，订单表ORDERS的外键 |
| PRODUCT\_ID | VARCHAR2(40 BYTE) | NO | 产品ID，是产品表PRODUCTS的外键 |
| PRODUCT\_NUM | NUMBER(8,2) | NO | 产品销售数量，必须大于0 |
| PRODUCT\_PRICE | NUMBER(8,2) | NO | 产品销售价格 |

为了使用触发器计算订单的应收货款，需要创建一个临时表ORDER\_ID\_TEMP，这个表的目的是暂时储存ORDER\_ID。见表4-6。

表 4-6 订单ID临时表ORDER\_ID\_TEMP

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ORDER\_ID | NUMBER(10,0) | NO | 主键 |

1. 用户创建与空间分配
   1. 创建表空间

表的结构设计好之后,还要考虑用户和空间分配问题。我们需要为系统新建一个用户( STUDY),在磁盘空间方面,考虑将数据存储在PDB上,这里选择 Oracle12c安装时默认的 PDBORCL数据库,另外,还需要为销售系统创建一个新表空间 USERS02用于存储订单记录，下面是 SYSTEM用户创建的一个表空间 USERSI02的命令,注意给 USERS02表空间分配了两个数据文件: pdbtest usen021dbf和 subtest users022dbf,这两个数据文件初始大小都是100M,所以表空间的初始大小是200M。

表空间创建代码如下：

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 UNLIMITE

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

* 1. 创建用户

表空间 USERS02创建完成之后创建用户.我们为本系统设计的用户名称是 STUDY.用户创建之后,给用户 STUDY分配表空间 USERS和 USERS02的使用配额再分配角色 CONNECT和 RESOURCE,以便该用户可以连接数据库,可以创建资源(表,过程,序列等资源对象),最后再分配一个系统权限:“ CREATE VIEW”,以便该用户可以创建视图。

创建角色并分配权限代码如下：

CREATE USER STUDY IDENTIFIED BY 123

DEFAULT TABLESPACE “USERS”

TEMPORARY TABLESPACE "TEMP”;

--QUOTAS

ALTER USER STUDY QUOTA UNLIMITED ON USERS;

ALTER USER STUDY QUOTA UNLIMITED ON USERS02;

--ROLES

GRANT "CONNECT" TO STUDY WITH ADMIN OPTION;

GRANT "RESOURCE" TO STUDY WITH ADMIN OPTION;

ALTER USER STUDY DEFAULT ROLE “CONNECT”, “RESOURCE”;

--SYSTEM PRIVILEGES

GRANT CREATE VTEW TO STUDY WITH ADMIN OPTION;

* 1. 存储空间的分配

到现在,我们有了两个表空间 USERS(原有的)和 USERS02(新建的)。这里我们假定企业的销售订单非常多,订单表和订单详单表的记录数量非常大,可能上千万条记录,所以我们把 ORDERS和 ORDE\_DETAILS两张表设计为基于 ORDER\_DATE的分区表,以加快局部时间范围的查询速度。存储空间的分配规划如表5-1所示。

表5-1 存储空间分配

|  |  |  |
| --- | --- | --- |
| 表 | 表空间USERS | 表空间USERS02 |
| DEPARTMENTS | 存储全部数据 |  |
| EMPLOYEES | 存储全部数据 |  |
| PRODUCTS | 存储全部数据 |  |
| ORDERS | 存储2016年之前（不含）的数据 | 存储2016年之后（不含）的数据 |
| ORDER\_DETAILS | 存储2016年之前（不含）的数据 | 存储2016年之后（不含）的数据 |

1. 创建表，约束和索引

本系统包括部门表，员工表，订单表，订单详单表，产品表。以订单表和订单详单表为例。ORDERS表按分区存储，分区类型选择“RANG”范围分区，按订单时间ORDER\_DATE的范围进行分区,分区PARTITION\_BEFORE\_2016存储订单时间小于2016年的订单记录,这个分区存储在表空间 USERS中,而分区PARTITION\_BEFORE2017存储的是订单时间小于2017年的订单记录,由于有分区PARTITION\_BEFORE\_2016的存在, PARTITION\_BEFORE\_2017分区实际只存储2016 年一年的记录。

创建ORDERS表的部分语句如下：

|  |
| --- |
| 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 |
|  |

|  |
| --- |
| , CONSTRAINT ORDERS\_PK PRIMARY KEY |
|  |

|  |
| --- |
| ( |
|  |

|  |
| --- |
| ORDER\_ID |
|  |

|  |
| --- |
| ) |
|  |

|  |
| --- |
| USING INDEX |
|  |

|  |
| --- |
| ( |
|  |

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

|  |
| --- |
| LOGGING |
|  |

|  |
| --- |
| TABLESPACE USERS |
|  |

|  |
| --- |
| PCTFREE 10 |
|  |

|  |
| --- |
| INITRANS 2 |
|  |

|  |
| --- |
| STORAGE |
|  |

|  |
| --- |
| ( |
|  |

|  |
| --- |
| BUFFER\_POOL DEFAULT |
|  |

|  |
| --- |
| ) |
|  |

|  |
| --- |
| NOPARALLEL |
|  |

|  |
| --- |
| ) |
|  |

|  |
| --- |
| ENABLE |
|  |

|  |
| --- |
| ) |
|  |

|  |
| --- |
| TABLESPACE USERS |
|  |

|  |
| --- |
| PCTFREE 10 |
|  |

|  |
| --- |
| INITRANS 1 |
|  |

|  |
| --- |
| STORAGE |
|  |

|  |
| --- |
| ( |
|  |

|  |
| --- |
| BUFFER\_POOL DEFAULT |
|  |

|  |
| --- |
| ) |
|  |

|  |
| --- |
| NOCOMPRESS |
|  |

|  |
| --- |
| NOPARALLEL |
|  |

|  |
| --- |
| PARTITION BY RANGE (ORDER\_DATE) |
|  |

|  |
| --- |
| ( |
|  |

|  |
| --- |
| PARTITION PARTITION\_2015 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\_2016 VALUES LESS THAN (TO\_DATE(' 2017-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS\_CALENDAR=GREGORIAN')) |
|  |

|  |
| --- |
| NOLOGGING |
|  |

|  |
| --- |
| TABLESPACE USERS02 |
|  |
|  |
|  |

ORDER\_DETAILS是 ORDERS的从表, ORDER\_DETAILS的记录数量比ORDERS还多,所以 ORDER\_DETAILS表也必须分区存储,但由于 ORDER\_DETAILS表中没有订单时间,所以不能按时间分区进行存储。ORDER\_DETAILS使用引用分区进行存储。

创建 ORDER\_DETAILS表的部分语句如下：

CREATE TABLE order\_details

(

id NUMBER(10, 0) NOT NULL

, order\_id NUMBER(10, 0) NOT NULL

, product\_id 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

...

)

NOCOMPRESS NO INMEMORY,

PARTITION PARTITION\_BEFORE\_2017

NOLOGGING

TABLESPACE USERS02

...

)

NOCOMPRESS NO INMEMORY

);

创建全局临时表ORDER\_ID\_TEMP的语句如下：

|  |
| --- |
| -- 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';

结果见图 6-1：

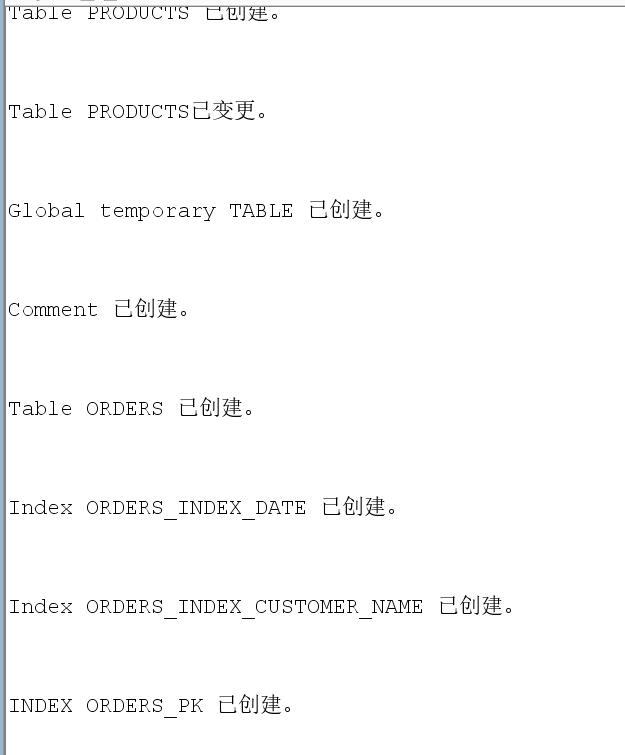


图 6-1

1. 创建触发器，序列和视图
   1. 创建触发器

本销售系统中订单表ORDERS的TRADE\_RECEIVABLE是应收货款,是自动计算字的计算公式是Trade\_Receivable=sum(ORDER\_DETAILS.Product\_Num\*ORDER\_DETAILS.Produet\_ pice)-Discount。从这个公式可以看出,当修改订单表的折扣 Discount或者对订单详单表 ORDER\_DETAILS作任何增加,修改,删除的操作后,都可能需要重新计算Trade\_Receivable。所以我们设计了以下3个触发器来实现这个功能,完整的触发器代码如下：

|  |
| --- |
| -- 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; |
|  |

|  |
| --- |
|  |

|  |
| --- |
| END; |
|  |

结果见图 7-1：



图 7-1

* 1. 创建序列

在插入订单数据的时候，需要用到自增的ID，所以需要设计两个序列SEQ\_ORDER\_ID用于给ORDER表的ORDER\_ID赋值，SEQ\_ORDER\_ID用于给ORDER\_DETAILS表的ORDER\_DETAILS \_ID赋值。

创建序列的语句如下：

|  |
| --- |
| -- DDL for Sequence SEQ\_ORDER\_ID |
|  |

|  |
| --- |
| -------------------------------------------------------- |
|  |

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

|  |
| --- |
| -------------------------------------------------------- |
|  |

|  |
| --- |
| -- DDL for Sequence SEQ\_ORDER\_DETAILS\_ID |
|  |

|  |
| --- |
| -------------------------------------------------------- |
|  |

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

结果见 图 7-2：

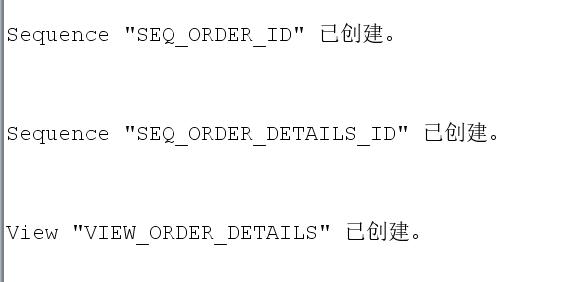


图 7-2

* 1. 创建视图

为了方便的查询订单详单中的产品，需要设计一个视图VIEW\_ORDER\_DETAILS

创建视图VIEW\_ORDER\_DETAILS的语句如下：

|  |
| --- |
| -- 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; |
|  |

/

结果见 图 7-3：

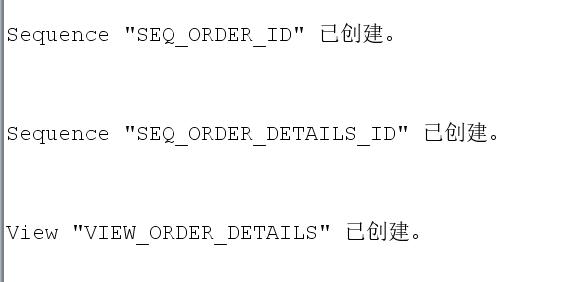


图 7-3

1. 创建程序包，函数和过程

本系统设计了一些函数和过程,放在程序包 MyPack中, Get\_SaleAmount（）函数的作用是计算并返回一个部门的销售总金额。Get\_Employees过程的作用是打印出某个员工的所有下属,含下属的下属, GET EMPLOYEEBYPAGE PO过程的作用是分页查询员工表,返回游标变量. Calc\_All\_TradeReceivable()过程的作用是更新订单表中每个订单的应收款。代码分为包头和包体两部分。

包头如下：

|  |
| --- |
| create or replace PACKAGE MyPack IS |
|  |

|  |
| --- |
|  |

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

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

|  |
| --- |
| END MyPack; |
|  |

/

包体如下：

|  |
| --- |
| 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; |
|  |

/

结果见 图 8-1

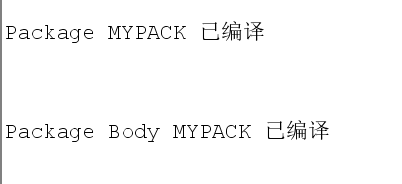


图 8-1

1. 数据库测试
   1. 向表中插入数据

以批量插入订单数据为例，其他表的数据插入语句省略，其中ORDERS.TRADE\_RECEIVABLE（订单应收款）自动计算,语句如下：

|  |
| --- |
| 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; --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;

结果见图 9-1

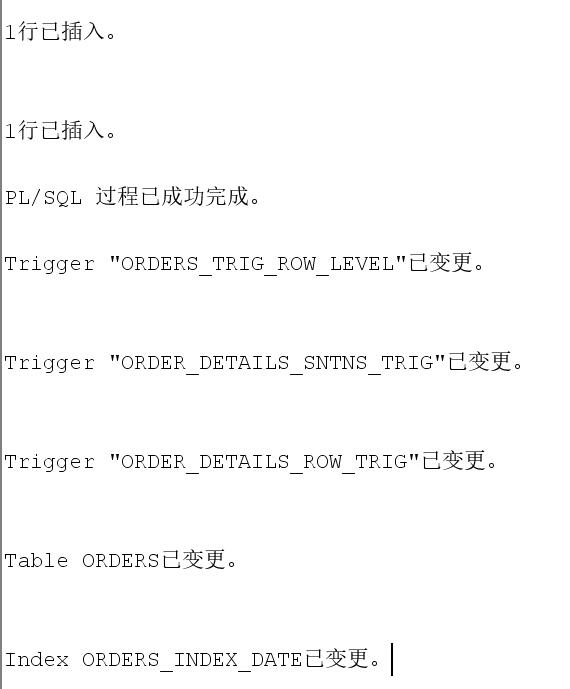


图 9-1

* 1. 调用程序包中的函数

调用MyPack.Get\_SaleAmount()函数，计算部门11的销售总金额，语句如下：

|  |
| --- |
| select count(\*) from orders; |
|  |

select MyPack.Get\_SaleAmount(11) AS 部门11应收金额,MyPack.Get\_SaleAmount(12) AS 部门12应收金额 from dual;

结果如图 9-2：



图 9-2

* 1. 调用程序包中的过程

在Sql developer 中调用MYPACK.Get\_Employees()过程，输出1号员工所有下属及子下属，语句如下：

|  |
| --- |
| set serveroutput on |
|  |

|  |
| --- |
| DECLARE |
|  |

|  |
| --- |
| V\_EMPLOYEE\_ID NUMBER; |
|  |

|  |
| --- |
| BEGIN |
|  |

|  |
| --- |
| V\_EMPLOYEE\_ID := 1; |
|  |

|  |
| --- |
| MYPACK.Get\_Employees ( V\_EMPLOYEE\_ID => V\_EMPLOYEE\_ID) ; |
|  |

|  |
| --- |
| V\_EMPLOYEE\_ID := 11; |
|  |

|  |
| --- |
| MYPACK.Get\_Employees ( V\_EMPLOYEE\_ID => V\_EMPLOYEE\_ID) ; |
|  |

|  |
| --- |
| END; |
|  |

/

结果见图9-3

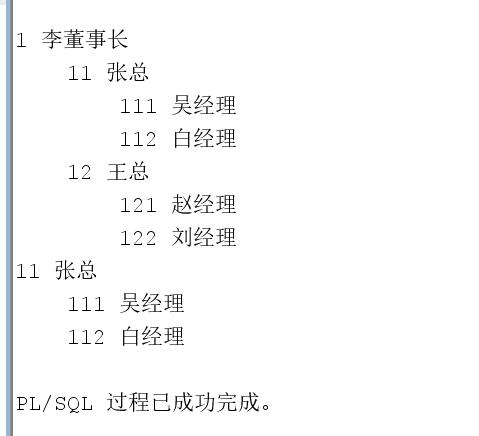


图9-3

1. 自动联机备份方案设计

每周7天为一个循环,每天23点钟都作一个备份:每周日23点钟作一次针对整个数据库的0级完整备份,每周的其余6天(周一到周六)每天都作一次1级增量备份,每次备份之后都要清除过期和失效的备份和归档日志。

先创建一个用存储备份文件和日志的目录: home/oracle/man\_backup,然后新建两个shell文件: /home/oracle/rman\_leve0.sh和 /home/oracle/man\_levell.sh. rman\_level0. sh是0级备份的脚本文件, rman\_levell.sh是一级备份的脚本文件。要保证这两个文件可执行

rman\_level0.sh内容如下:

#rman\_level0.sh

#!/bin/sh

export NLS\_LANG=’SIMPLIFIED CHINESE\_CHINA. AL32UTF8’

export ORACLE\_HOME=/home/oracle/app/oracle/product /12. 1.0/dbhome\_1

export ORACLE\_SID=orcl

export $PATH=SORACLE HOME/bin: $PATH

rman target/ nocatalog msglog=/home/oracle/rman\_backup/lvo0\_date+%Y%m%d-%H%M%S~\_L0.log<<EOF

run{

configure retention policy to redundancy 1;

configure controlfile autobackup on;

configure controlfile autobackup format for device type disk to

‘/home/oracle/rman\_backup/%F’;

configure default device type to disk;

crosscheck archivelog all;

allocate channel cl device type disk;

backup as compressed backupset incremental level 0 database format

‘/home /oracle/rman\_backup/dblv0\_%d\_%T\_ %U. bak’

Plus archivelog format /home /oracle/rman\_backup/dblv0\_%d\_%T\_ %U. bak’;

Report obsolete;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

release channel c1;

}

EOF

exit

rman\_level1.sh内容如下:

#rman\_level1.sh

#!/bin/sh

export NLS\_LANG=’SIMPLIFIED CHINESE\_CHINA. AL32UTF8’

export ORACLE\_HOME=/home/oracle/app/oracle/product /12. 1.0/dbhome\_1

export ORACLE\_SID=orcl

export $PATH=SORACLE HOME/bin: $PATH

rman target/ nocatalog msglog=/home/oracle/rman\_backup/lvo0\_date+%Y%m%d-%H%M%S~\_L0.log<<EOF

run{

configure retention policy to redundancy 1;

configure controlfile autobackup on;

configure controlfile autobackup format for device type disk to

‘/home/oracle/rman\_backup/%F’;

configure default device type to disk;

crosscheck archivelog all;

allocate channel cl device type disk;

backup as compressed backupset incremental level 1 database format

‘/home /oracle/rman\_backup/dblv1\_%d\_%T\_ %U. bak’

Plus archivelog format /home /oracle/rman\_backup/dblv1\_%d\_%T\_ %U. bak’;

Report obsolete;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

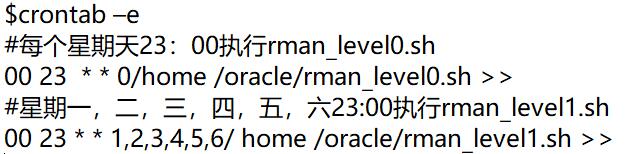
release channel c1;

}

EOF

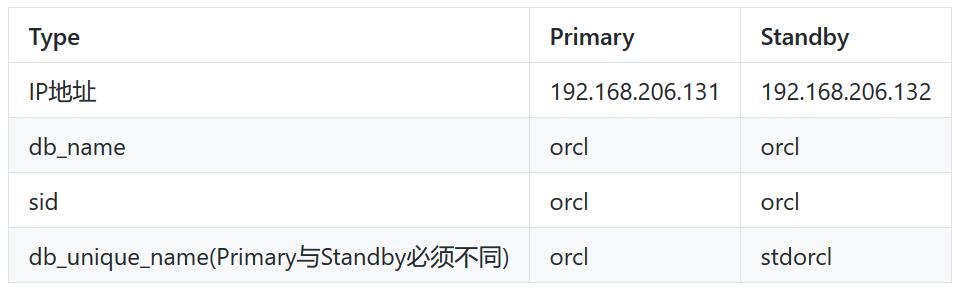
Exit

下面通过linux的crontab命令创建自动执行脚本文件的调度：



1. 容灭方案设计

主机设置如下：



在备库oracle用户创建归档目录，数据目录并设置权限

mkdir -p /home/oracle/app/oracle/diag/orcl

mkdir -p /home/oracle/app/oracle/oradata/stdorcl/

mkdir -p /home/oracle/app/oracle/oradata/stdorcl/pdborcl

mkdir -p /home/oracle/arch

mkdir -p /home/oracle/rman

mkdir -p /home/oracle/app/oracle/oradata/stdorcl/pdbseed/

mkdir -p /home/oracle/app/oracle/oradata/stdorcl/pdb/

删除原有数据库:

$sqlplus / as sysdba

shutdown immediate;

startup mount exclusive restrict;

drop database;

启动到nomount

$sqlplus / as sysdba

startup nomount

$sqlplus / sysdba

select group#,thread#,members,status from v$log;

增加standbylogfile：

alter database add standby logfile group 5 '/home/oracle/app/oracle/oradata/orcl/stdredo1.log' size 50m;

alter database add standby logfile group 6 '/home/oracle/app/oracle/oradata/orcl/stdredo2.log' size 50m;

alter database add standby logfile group 7 '/home/oracle/app/oracle/oradata/orcl/stdredo3.log' size 50m;

alter database add standby logfile group 8 '/home/oracle/app/oracle/oradata/orcl/stdredo4.log' size 50m;

主库环境开启强制归档

ALTER DATABASE FORCE LOGGING;

alter system set LOG\_ARCHIVE\_CONFIG='DG\_CONFIG=(orcl,stdorcl)' scope=both sid='\*';

alter system set log\_archive\_dest\_1='LOCATION=/home/oracle/arch VALID\_FOR=(ALL\_LOGFILES,ALL\_ROLES) DB\_UNIQUE\_NAME=orcl' scope=spfile;

alter system set LOG\_ARCHIVE\_DEST\_2='SERVICE=stdorcl LGWR ASYNC VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=stdorcl' scope=both sid='\*';

alter system set fal\_client='orcl' scope=both sid='\*';

alter system set FAL\_SERVER='stdorcl' scope=both sid='\*';

alter system set standby\_file\_management=AUTO scope=both sid='\*';

alter system set DB\_FILE\_NAME\_CONVERT='/home/oracle/app/oracle/oradata/stdorcl/','/home/oracle/app/oracle/oradata/orcl/' scope=spfile sid='\*';

alter system set LOG\_FILE\_NAME\_CONVERT='/home/oracle/app/oracle/oradata/stdorcl/','/home/oracle/app/oracle/oradata/orcl/' scope=spfile sid='\*';

alter system set log\_archive\_format='%t\_%s\_%r.arc' scope=spfile sid='\*';

alter system set remote\_login\_passwordfile='EXCLUSIVE' scope=spfile;

alter system set PARALLEL\_EXECUTION\_MESSAGE\_SIZE=8192 scope=spfile;

编辑主库以及备库的/home/oracle/app/oracle/product/12.1.0/dbhome\_1/network/admin/tnsnames.ora文件

$gedit /home/oracle/app/oracle/product/12.1.0/dbhome\_1/network/admin/tnsnames.ora

ORCL =

(DESCRIPTION =

(ADDRESS\_LIST =

(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.206.131)(PORT = 1521))

)

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = orcl)

)

)

stdorcl =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.206.132)(PORT = 1521))

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SID = orcl)

)

)

在主库上生成备库的参数文件:

SQL>create pfile from spfile;

生成/home/oracle/app/oracle/product/12.1.0/dbhome\_1/dbs/initorcl.ora

将主库的参数文件，密码文件拷贝到备库:

scp /home/oracle/app/oracle/product/12.1.0/dbhome\_1/dbs/initorcl.ora 192.168.206.132:/home/oracle/app/oracle/product/12.1.0/dbhome\_1/dbs/

scp /home/oracle/app/oracle/product/12.1.0/dbhome\_1/dbs/orapworcl 192.168.206.132:/home/oracle/app/oracle/product/12.1.0/dbhome\_1/dbs/

将主库复制到备库

$rman target sys/123@orcl auxiliary sys/123@stdorcl

执行duplicate:

run{

allocate channel c1 type disk;

allocate channel c2 type disk;

allocate channel c3 type disk;

allocate AUXILIARY channel c4 type disk;

allocate AUXILIARY channel c5 type disk;

allocate AUXILIARY channel c6 type disk;

DUPLICATE TARGET DATABASE

FOR STANDBY

FROM ACTIVE DATABASE

DORECOVER

NOFILENAMECHECK;

release channel c1;

release channel c2;

release channel c3;

release channel c4;

release channel c5;

release channel c6;

}

在备库上更改参数文件

$gedit /home/oracle/app/oracle/product/12.1.0/dbhome\_1/dbs/initorcl.ora

文件内容是：

orcl.\_\_data\_transfer\_cache\_size=0

orcl.\_\_db\_cache\_size=671088640

orcl.\_\_java\_pool\_size=16777216

orcl.\_\_large\_pool\_size=33554432

orcl.\_\_oracle\_base='/home/oracle/app/oracle'#ORACLE\_BASE set from environment

orcl.\_\_pga\_aggregate\_target=536870912

orcl.\_\_sga\_target=1258291200

orcl.\_\_shared\_io\_pool\_size=50331648

orcl.\_\_shared\_pool\_size=301989888

orcl.\_\_streams\_pool\_size=0

\*.\_allow\_resetlogs\_corruption=TRUE

\*.\_catalog\_foreign\_restore=FALSE

\*.audit\_file\_dest='/home/oracle/app/oracle/admin/orcl/adump'

\*.audit\_trail='db'

\*.compatible='12.1.0.2.0'

\*.control\_files='/home/oracle/app/oracle/oradata/orcl/control01.ctl','/home/oracle/app/oracle/fast\_recovery\_area/orcl/control02.ctl','/home/oracle/app/oracle/fast\_recovery\_area/orcl/control03.ctl'

\*.db\_block\_size=8192

\*.db\_domain=''

\*.db\_file\_name\_convert='/home/oracle/app/oracle/oradata/orcl/','/home/oracle/app/oracle/oradata/stdorcl/'

\*.db\_name='orcl'

\*.db\_unique\_name='stdorcl'

\*.db\_recovery\_file\_dest='/home/oracle/app/oracle/fast\_recovery\_area'

\*.db\_recovery\_file\_dest\_size=4823449600

\*.diagnostic\_dest='/home/oracle/app/oracle'

\*.dispatchers='(PROTOCOL=TCP)(dispatchers=1)(pool=on)(ticks=1)(connections=500)(sessions=1000)'

\*.enable\_pluggable\_database=true

\*.fal\_client='stdorcl'

\*.fal\_server='orcl'

\*.inmemory\_max\_populate\_servers=2

\*.inmemory\_size=157286400

\*.local\_listener=''

\*.log\_archive\_config='DG\_CONFIG=(stdorcl,orcl)'

\*.log\_archive\_dest\_1='LOCATION=/home/oracle/arch VALID\_FOR=(ALL\_LOGFILES,ALL\_ROLES) DB\_UNIQUE\_NAME=stdorcl'

\*.log\_archive\_dest\_2='SERVICE=orcl LGWR ASYNC VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=orcl'

\*.log\_archive\_format='%t\_%s\_%r.arc'

\*.log\_file\_name\_convert='/home/oracle/app/oracle/oradata/orcl/','/home/oracle/app/oracle/oradata/stdorcl/'

\*.max\_dispatchers=5

\*.max\_shared\_servers=20

\*.open\_cursors=400

\*.parallel\_execution\_message\_size=8192

\*.pga\_aggregate\_target=511m

\*.processes=300

\*.recovery\_parallelism=0

\*.remote\_login\_passwordfile='EXCLUSIVE'

\*.service\_names='ORCL'

\*.sga\_max\_size=1572864000

\*.sga\_target=1258291200

\*.shared\_server\_sessions=200

\*.standby\_file\_management='AUTO'

\*.undo\_tablespace='UNDOTBS1'

在备库增加静态监听

$gedit /home/oracle/app/oracle/product/12.1.0/dbhome\_1/network/admin/listener.ora

SID\_LIST\_LISTENER =

(SID\_LIST =

(SID\_DESC =

(ORACLE\_HOME = /home/oracle/app/oracle/product/12.1.0/db\_1)

(SID\_NAME = orcl)

)

)

重新启动,备库开启实时应用模式:：

$sqlplus / as sysdba

shutdown immediate

startup

alter database recover managed standby database disconnect;

可以把语句alter database recover managed standby database disconnect;放到触发器中去，开机启动：

在主库修改数据，在备库查询修改。 Select Name,Sequence#,applied,completion\_time From v$archived\_log Order By Sequence# Desc;

1. 总结

该果蔬电商系统的数据库设计包括表设计，用户管理，PL/SQL设计，备份方案设计以及容灭方案设计，难度逐渐增加，同时也将本学期所学的oracle的知识点全部串接起来了，将它们一起运用便完成了一个数据库系统设计。将理论与实际结合了起来，不仅让我明白了可以干什么，还让我知道了该怎么去用。不是学习了轻飘飘了理论知识，而是学习了如何掌握使用这些技能。