

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的职工信息的数据库设计 | | |
| 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 | 2017级 |
| 学生姓名 | 王志翔 | 学号 | 201710414420 |
| 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** | | |  |

2019 年 11 月 18 日

目录

[1. 引言 3](#_Toc1204)

[1.1编写目的 3](#_Toc18151)

[1.2术语表 3](#_Toc31959)

[1.3参考资料 3](#_Toc18782)

[2. 数据库环境 3](#_Toc11508)

[3. 逻辑设计 4](#_Toc10787)

[3.1实体模型 4](#_Toc28784)

[3.2实体联系模型 6](#_Toc745)

[4. 物理设计 8](#_Toc32402)

[4.1表汇总 8](#_Toc2254)

[4.2数据库的设计 8](#_Toc28452)

[4.3用户创建与空间分配 10](#_Toc30452)

[4.4创建表，约束和索引 12](#_Toc4411)

[4.5创建触发器、插入数据 16](#_Toc18097)

[4.6创建包、函数 20](#_Toc3362)

[5. 安全型设计 21](#_Toc27200)

[5.1数据库的备份与恢复 21](#_Toc21183)

# 引言

## 1.1编写目的

《职工考勤管理信息系统》是为实现职工考勤信息管理的现代化，运用Oracle数据库管理系统开发的应用软件。它旨在通过职工基本信息、职工加班信息、职工出勤信息、职工出差信息和职工请假信息这五方面完成对职工的考勤。利用计算机技术做出这样一个系统就节省了很多人力资源，实现了科学、高效的职工考勤信息管理目标。

## 1.2术语表

|  |  |  |
| --- | --- | --- |
| 序号 | 术语或缩略语 | 说明性定义 |
| 1 | ATTENDANCE\_NUM | 该考勤项的天数 |
| 2 | SALARY | 职工的月薪 |
| 3 | ATTENDANCE\_PRICE | 考勤所得金额 |
| 4 | F\_SALARY | 最后实际的薪资 |

## 1.3参考资料

Oracle 12c数据库基础教程 科学出版社 主编赵卫东、刘永红、于曦

# 数据库环境

|  |  |  |
| --- | --- | --- |
| 数据库系统 | 部署环境 | 设计工具 |
| oracle | linux | sqldeveloper |

# 逻辑设计

## 3.1实体模型

根据应用场景分析，共有3个原始的实体，分别是部门、职工和考勤项。

部门（DEPARTMENTS）:部门包括部门ID（DEPARTMENT\_ID）和部门名称（DEPARTMENT\_NAME），如图。

DEPARTMENTS(部门)

DEPARTMENT\_ID

DEPARTMENT\_NAME

职工（STAFF）:职工包括职工ID（STAFF\_ID），姓名（STAFF\_NAME），性别（SEX），电话（PHONE\_NUMBER），月薪（SALARY）等。职工的属性中还应该包括职工所属的部门ID（DEPARTMENT\_ID），部门ID不能为空，表示职工必须属于某一个部门，此外应该还有职工的上司（MANAGER\_ID），实际工资（F\_SALARY），实际工资需要经过月薪加减考勤项的金额得出。职工的实体如图。

STAFFS(职工)

STAFF\_ID

F\_SALARY

DEPARTMENT\_ID

STAFF\_NAME

SALARY

SEX

MANAGER\_ID

PHONE\_NUMBER

考勤项（ATTENDANCES）:考勤项包括考勤ID（出勤ID（ATTENDANCE\_ID）或加班ID（OVER\_ID）或出差ID（LEAVE\_ID）或请假ID（TAKEOFF\_ID）），考勤名称ATTENDANCE\_NAME

（出勤（ATTENDANCE）或加班（OVER）或出差（LEAVE）或请假（TAKEOFF）），见图。

ATTENDANCES(考勤项)

ATTENDANCE\_NAME

ATTENDANCE\_ID

薪资（WAGES）：职工ID（STAFF\_ID），姓名（STAFF\_NAME），月薪（SALARY），考勤金额（ATTENDANCE\_PRICE），实际工资（F\_SALARY）等，如图。

WAGES(薪资)

STAFF\_NAME

F\_SALARY

STAFF\_ID

SALARY

ATTENDANCE\_PRICE

DATE

薪资详单（WAGE\_DETAILS）薪资详单的ID，职工ID（STAFF\_ID），考勤ID（出勤ID（ATTENDANCE\_ID）或加班ID（OVER\_ID）或出差ID（LEAVE\_ID）或请假ID（TAKEOFF\_ID）），考勤天数DAY\_NUM（加班天数（OVER\_NUM）或请假天数（TAKEOFF\_NUM）），以及加班金额（OVER\_PRICE），缺勤金额（TAKEOFF\_PRICE），奖金（BONUS）等，如图。

WAGE\_DETAILS(薪资详单)

详单ID

STAFF\_ID

考勤ID

DAY\_NUM

BONUS

OVER\_PRICE

TAKEOFF\_PRICE

DATE

## 3.2实体联系模型

职工的实际薪资是需要对该月的考勤情况进行统计的，因此职工与考勤项存在一个“考勤”的联系，职工与考勤项之间的关系是多对多的关系，如图。

职工

考勤

考勤项

M

N

由于职工表中的部门不为空，即他们也存才“从属”关系，一个职工只可以属于一个部门，而一个部门可以包含多个职工，职工与部门之间的关系是多对一的关系，如图。

职工

从属

部门

N

1

考虑到对职工薪资的计算属性太多，存在一个表中过于繁杂，于是将薪资关系细分为薪资与薪资详单两个实体，薪资中存储：职工ID（STAFF\_ID），姓名（STAFF\_NAME），月薪（SALARY），考勤金额（ATTENDANCE\_PRICE），实际工资（F\_SALARY），薪资详单中存储薪资详单的ID，职工ID（STAFF\_ID），考勤ID及天数包括职工的所有考勤情况：加班天数（OVER\_NUM），请假天数（TAKEOFF\_NUM），以及加班金额（OVER\_PRICE），超出规定请假后的缺勤金额（TAKEOFF\_PRICE），全勤后的奖金（BONUS），考勤金额=奖金+加班金额\*加班天数-缺勤金额\*请假天数。实体间关系如图，职工每月都会有薪资，所以是多对多的关系，而一个职工每月薪资都有多个薪资详单，每个详单存储一类考勤项。

职工

薪资

薪资详单

考勤项

M

N

1

M

M

1

# 物理设计

## 4.1表汇总

|  |  |
| --- | --- |
| 表名 | 功能说明 |
| 部门表 | 职能的划分 |
| 职工表 | 员工的详细信息 |
| 考勤项表 | 职工的考勤项目 |
| 薪资表 | 职工每月的薪资说明 |
| 薪资详单表 | 职工薪资的详细说明 |

## 4.2数据库的设计

部门表（DEPARTMENTS）包括DEPARTMENT\_ID和DEPARTMENT\_NAME两个属性，见表。

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

职工表（STAFFS）包括职工ID（STAFF\_ID），姓名（STAFF\_NAME），性别（SEX），电话（PHONE\_NUMBER），月薪（SALARY），所属的部门ID（DEPARTMENT\_ID），职工的上司（MANAGER\_ID），实际工资（F\_SALARY），见表。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| STAFF\_ID | NUMBER(10,0) | NO | 职工ID,职工表主键 |
| STAFF\_NAME | VARCHAR2(40 BYTE) | NO | 职工姓名，不能为空 |
| SEX | VARCHAR2(4 BYTE) | YES | 性别 |
| PHONE\_NUMBER | VARCHAR2(40 BYTE) | YES | 电话 |
| SALARY | NUMBER(8,2) | YES | 月薪，必须>0 |
| DEPARTMENT\_ID | NUMBER(6,0) | YES | 职工所在部门 |
| MANAGER\_ID | NUMBER(6,0) | YES | 职工的上司 |
| F\_SALARY | NUMBER(8,2) | YES | 职工所得实际薪资 |

考勤项表（ATTENDANCES）考勤项包括考勤ID（ATTENDANCE\_ID），考勤名称ATTENDANCE\_NAME（出勤（ATTENDANCE）或加班（OVER）或出差（LEAVE）或请假（TAKEOFF）），见表。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ATTENDANCE\_ID | NUMBER(2,0) | NO | 考勤ID号，考勤项表主键 |
| ATTENDANCE\_NAME | VARCHAR2(20 BYTE) | NO | 考勤项目名称 |

薪资表（WAGES）包括职工ID（STAFF\_ID），姓名（STAFF\_NAME），月薪（SALARY），考勤金额（ATTENDANCE\_PRICE），实际工资（F\_SALARY），日期（DATE），见表。

薪资表中的实际工资即为职工表的实际工资，经过薪资表中的信息汇总计算得来，计算公式是：实际工资（F\_SALARY）=月薪（SALARY）-考勤金额（ATTENDANCE\_PRICE），而考勤金额是经过薪资详单中的信息汇总计算得来，计算公式是：考勤金额（ATTENDANCE\_PRICE）=奖金（BONUS）+加班金额（OVER\_PRICE）\*加班天数（OVER\_NUM）-缺勤金额（TAKEOFF\_PRICE）\*请假天数（TAKEOFF\_NUM）。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| STAFF\_ID | NUMBER(10,0) | NO | 职工ID |
| STAFF\_NAME | VARCHAR2(40 BYTE) | NO | 职工姓名 |
| SALARY | NUMBER(8,2) | NO | 月薪 |
| ATTENDANCE\_PRICE | NUMBER(8,2) | NO | 考勤金额 |
| F\_SALARY | NUMBER(8,2) | NO | 职工所得实际薪资 |
| DATE | DATE | NO | 日期，只到年月 |

薪资详单表（WAGE\_DETAILS）包含薪资详单的ID，职工ID（STAFF\_ID），出勤ID（ATTENDANCE\_ID），考勤名称（ATTENDANCE\_NAME），加班ID（OVER\_ID），加班天数（OVER\_NUM），出差ID（LEAVE\_ID），请假ID（TAKEOFF\_ID），请假天数（TAKEOFF\_NUM），以及加班金额（OVER\_PRICE），缺勤金额（TAKEOFF\_PRICE），奖金（BONUS），日期（DATE），见表。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 可以为空 | 注释 |
| ID | NUMBER(12,0) | NO | 本表的主键，值来自序列 |
| ATTENDANCE\_ID | NUMBER(2,0) | NO | 考勤项的ID |
| ATTENDANCE\_NAME | VARCHAR2(20 BYTE) | YES | 考勤项名称 |
| OVER\_NUM | NUMBER(2,0) | NO | 加班的天数 |
| OVER\_PRICE | NUMBER(8,2) | YES | 加班金额/天 |
| TAKEOFF\_PRICE | NUMBER(8,2) | YES | 缺勤金额/天 |
| BONUS | NUMBER(8,2) | YES | 满勤的奖金 |
| DATE | DATE | NO | 日期，只到年月 |
| TAKEOFF\_NUM | NUMBER(2,0) | NO | 请假的天数 |
| STAFF\_ID | NUMBER(10,0) | NO | 职工的ID |

## 4.3用户创建与空间分配

设计好表的结构后，需要考虑用户与空间的分配问题。该系统的角色有两类，分别为管理员与职工，只有管理员同时拥有读写权限，职工只能进行读的权限。此外还新建了两个表空间USERS02，USERS03用于存储薪资记录。

使用SYSTEM（管理员）创建表空间USERS02，USERS03，表空间由两个数据文件组成，每个文件的初始大小都是100M，自动增长，每次扩展50M，文件大小无限制。

$ sqlplus system/123@pdborcl

SQL>create tablespace users02 datafile

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

SIZE 100M AUTOEXTEND ON NEXT 50M MAXSIZE UNLIMITED,

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

SIZE 100M AUTOEXTEND ON NEXT 50M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

SQL>create tablespace users03 datafile

‘/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_users03\_1.dbf’

SIZE 100M AUTOEXTEND ON NEXT 50M MAXSIZE UNLIMITED,

‘/home/oracle/app/oracle/oradata/orcl/pdborcl/pdbtest\_users03\_2.dbf’

SIZE 100M AUTOEXTEND ON NEXT 50M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

表空间USERS02，USERS03创建完成后仍然使用SYSTEM创建角色staff\_view和用户staff1，由于用户staff1仅为普通用户，所以授予connect，resource权限就足够了。

$ sqlplus system/123@pdborcl

SQL> CREATE ROLE staff\_view;

SQL> GRANT connect,resource,CREATE VIEW TO staff\_view;

SQL> CREATE USER staff1 IDENTIFIED BY 123 DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;

SQL> ALTER USER staff1 QUOTA UNLIMITED ON users;

SQL> ALTER USER staff1 QUOTA UNLIMITED ON users02;

SQL> ALTER USER staff1 QUOTA UNLIMITED ON users03;

SQL> GRANT con\_res\_view TO staff1;

SQL> exit

由于企业职工多且每个职工的薪资记录也需要多条数据记录计算，数量非常大，所以将薪资表与薪资详单表设计为基于日期的分区表，以加快局部时间范围的查询速度。存储空间的分配规划如下表。

|  |  |  |  |
| --- | --- | --- | --- |
| 表 | USERS | USERS02 | USERS03 |
| DEPARTMENTS | 存储全部数据 |  |  |
| STAFFS | 存储全部数据 |  |  |
| ATTENDANCES | 存储全部数据 |  |  |
| WAGES | 存储2017年前（含）的数据 | 存储2018年的数据 | 存储2019年后（含）的数据 |
| WAGE\_DETAILS | 存储2017年前（含）的数据 | 存储2018年的数据 | 存储2019年后（含）的数据 |

## 4.4创建表，约束和索引

在用户和空间分配完成后就可以创建表了，创建表的命令为CREATE TABLE，在进行薪资表与薪资详单表的创建时需要按分区存储，分区类型选择“RANG”范围分区，核心的语句为“PARTITION BY RANGE(DATE)”，表示按日期DATE的范围进行分区，下面是薪资表的创建。

CREATE TABLE WAGES

(

STAFF\_ID NUMBER(10, 0) NOT NULL

, STAFF\_NAME VARCHAR2(40 BYTE) NOT NULL

, DATE DATE NOT NULL

, SALARY NUMBER(8, 2) NOT NULL

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

, F\_SALARY NUMBER(8, 2) NOT NULL

, CONSTRAINT WAGES\_PK PRIMARY KEY

(

STAFF\_ID , DATE

)

TABLESPACE USERS

PCTFREE 10 INITRANS 1

STORAGE

( BUFFER\_POOL DEFAULT )

NOCOMPRESS NOPARALLEL

PARTITION BY RANGE (DATE)

(

PARTITION PARTITION\_BEFORE\_2018 VALUES LESS THAN (TO\_DATE(' 2018-01 ', 'SYYYY-MM', '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\_2019 VALUES LESS THAN (TO\_DATE(' 2019-01', 'SYYYY-MM', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION PARTITION\_AFTER\_2019 VALUES MORE THAN (TO\_DATE(' 2019-01', 'SYYYY-MM', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE USERS03

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

);

分区PARTITION\_BEFORE\_2018存储日期在2018年之前的职工薪资，这个分区储存在表空间USERS中，而分区PARTITION\_BEFORE\_2019存储日期在2018年到2019年的职工薪资，这个分区储存在表空间USERS02中，分区PARTITION\_AFTER\_2019存储日期在2019年之后的职工薪资，这个分区储存在表空间USERS03中。

由于WAGE\_DETAILS是WAGES的从表，WAGE\_DETAILS的记录数量比WAGES要多，所以WAGE\_DETAILS表也要进行分区储存，同样是基于日期进行分区储存。

CREATE TABLE WAGE\_DETALS

(

ID NUMBER(12,0) NOT NULL

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

, ATTENDANCE\_ID NUMBER(2, 0) NOT NULL

, ATTENDANCE\_NAME VARCHAR2(20 BYTE)

, DATE DATE NOT NULL

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

, OVER\_NUM NUMBER(2, 0) NOT NULL

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

, TAKEOFF\_NUM NUMBER(2, 0) NOT NULL

, CONSTRAINT WAGES\_PK PRIMARY KEY

( ID )

TABLESPACE USERS

PCTFREE 10 INITRANS 1

STORAGE

( BUFFER\_POOL DEFAULT )

NOCOMPRESS NOPARALLEL

PARTITION BY RANGE (DATE)

(

PARTITION PARTITION\_BEFORE\_2018 VALUES LESS THAN (TO\_DATE(' 2018-01 ', 'SYYYY-MM', '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\_2019 VALUES LESS THAN (TO\_DATE(' 2019-01', 'SYYYY-MM', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE USERS02

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

, PARTITION PARTITION\_AFTER\_2019 VALUES MORE THAN (TO\_DATE(' 2019-01', 'SYYYY-MM', 'NLS\_CALENDAR=GREGORIAN'))

NOLOGGING

TABLESPACE USERS03

PCTFREE 10

INITRANS 1

STORAGE

(

BUFFER\_POOL DEFAULT

)

NOCOMPRESS NO INMEMORY

);

## 4.5创建触发器、插入数据

在职工信息系统中的薪资表的F\_SALARY和ATTENDANCE\_PRICE是自动计算的字段，计算公式为：实际工资（F\_SALARY）=月薪（SALARY）+考勤金额（ATTENDANCE\_PRICE），考勤金额（ATTENDANCE\_PRICE）=奖金（BONUS）+加班金额（OVER\_PRICE）\*加班天数（OVER\_NUM）-缺勤金额（TAKEOFF\_PRICE）\*请假天数（TAKEOFF\_NUM）。当修改薪资表的月薪或者薪资详单表的各类金额都会影响最后的实际薪资，所以设计触发器来实现自动计算。

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

BEFORE INSERT OR UPDATE OF DISCOUNT ON "WAGES"

FOR EACH ROW

declare

m number(8,2);

BEGIN

if inserting then

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

else

select sum(BONUS+OVER\_NUM\*OVER\_PRICE-TAKEOFF\_PRICE\*TAKEOFF\_NUM) into m from WAGE\_DETAILS where STAFF\_ID=:old.STAFF\_ID and DATE=:old.DATE;

if m is null then

m:=0;

end if;

:new.TRADE\_RECEIVABLE := m + :new.salary;

end if;

END;

/

现在需要给创建好的表插入数据，先是需要给部门表、职工表以及考勤项表进行数据插入，部门表与考勤项表的数据是较为固定的，插入的数据较少，职工表就要大一些。

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

INSERT INTO STAFFS(STAFF\_ID,STAFF\_NAME,SEX,PHONE\_NUMBER,DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)VALUES (1,'王老板',NULL,NULL,to\_date('2016-01','yyyy-mm'),50000,NULL,1);

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

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

INSERT INTO STAFFS(STAFF\_ID,STAFF\_NAME,SEX,PHONE\_NUMBER,DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)VALUES (11,'张三',NULL,NULL,to\_date('2016-01','yyyy-mm'),7000,NULL,1);

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

INSERT INTO STAFFS(STAFF\_ID,STAFF\_NAME,SEX,PHONE\_NUMBER,DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)VALUES (12,'李四',NULL,NULL,to\_date('2016-01','yyyy-mm'),5000,NULL,1);

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

INSERT INTO STAFFS(STAFF\_ID,STAFF\_NAME,SEX,PHONE\_NUMBER,DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)VALUES (13,'王五',NULL,NULL,to\_date('2016-01','yyyy-mm'),4500,NULL,1);

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

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (3,'开发部');

INSERT INTO STAFFS(STAFF\_ID,STAFF\_NAME,SEX,PHONE\_NUMBER,DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)VALUES (14,'赵六',NULL,NULL,to\_date('2016-01','yyyy-mm'),5500,NULL,1);

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (3,'开发部');

INSERT INTO STAFFS(STAFF\_ID,STAFF\_NAME,SEX,PHONE\_NUMBER,DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)VALUES (15,'琴七',NULL,NULL,to\_date('2016-01','yyyy-mm'),5000,NULL,1);

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (3,'开发部');

INSERT INTO STAFFS(STAFF\_ID,STAFF\_NAME,SEX,PHONE\_NUMBER,DATE,SALARY,MANAGER\_ID,DEPARTMENT\_ID)VALUES (16,'何九',NULL,NULL,to\_date('2016-01','yyyy-mm'),8000,NULL,1);

INSERT INTO DEPARTMENTS(DEPARTMENT\_ID,DEPARTMENT\_NAME) values (3,'开发部');

INSERT INTO ATTENDANCES (ATTENDANCES\_ID,ATTENDANCES\_NAME) VALUES ('1','出勤');

INSERT INTO ATTENDANCES (ATTENDANCES\_ID,ATTENDANCES\_NAME) VALUES ('2','出差');

INSERT INTO ATTENDANCES (ATTENDANCES\_ID,ATTENDANCES\_NAME) VALUES ('3','请假');

INSERT INTO ATTENDANCES (ATTENDANCES\_ID,ATTENDANCES\_NAME) VALUES ('4','加班');

如果在WAGES表中插入1万条数据，那么WAGE\_DETAILS表将会存在4万条数据，循环进行数据的插入。

declare

dt date;

m number(8,2);

V\_EMPLOYEE\_ID NUMBER(6);

v\_staff\_id number(10);

v\_staff\_name varchar2(100);

v number(10,2);

begin

for i in 1..10000

loop

if i mod 2 =0 then

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

else

dt:=to\_date('2016-3','yyyy-mm')+(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;

v\_staff\_id:=SEQ\_STAFF\_ID.nextval;

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

v\_staff\_name := 'zhang' || i;

insert /\*+append\*/ into WAGES (STAFF\_ID,STAFF\_NAME,DATE,EMPLOYEE\_ID,BONUS)

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

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

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

insert /\*+append\*/ into WAGE\_DETAILS(ID,STAFF\_ID,STAFF\_NAME,OVER\_NUM,OVER\_PRICE,TAKEOFF\_NUM,TAKEOFF\_PRICE)

values (SEQ\_WAGE\_DETAILS\_ID.NEXTVAL,v\_staff\_id,v\_staff\_name,2,v,3,v);

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

select sum(BONUS+OVER\_NUM\*OVER\_PRICE-TAKEOFF\_PRICE\*TAKEOFF\_NUM) into m from WAGE\_DETAILS where STAFF\_ID=v\_staff\_id and DATE=dt;

if m is null then

m:=0;

end if;

UPDATE WAGES SET TRADE\_RECEIVABLE = m + salary WHERE STAFF\_ID=v\_staff\_id;

IF I MOD 1000 =0 THEN

commit;

END IF;

end loop;

end;

/

## 4.6创建包、函数

职工信息系统还设计的有函数，放在程序包MyPack中。函数get\_salary()函数的作用是计算某员工在公司工作所得所有金额。

create or replace PACKAGE MyPack IS

FUNCTION get\_salary(STAFF\_ID NUMBER) RETURN NUMBER;

END MyPack;

/

create or replace PACKAGE BODY MyPack IS

FUNCTION Bonus(STAFF\_ID NUMBER) RETURN NUMBER

AS

N NUMBER(20,2);

BEGIN

SELECT SUM(F\_SALARY) into N FROM WAGES O

WHERE O.STAFF\_ID=STAFF\_ID AND O.DATE =DATE;

RETURN N;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(‘你需要的数据不存在’);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLCODE||’---’SQLERRM);

END;

END MyPack;

/

# 安全型设计

## 5.1数据库的备份与恢复

备份与恢复是数据库设计不可缺失的一环，如果数据库崩溃却无法进行恢复结果可能是毁灭性的。在任何情况下，无论数据库服务器有多稳定，可靠，我们都需要制定备份与恢复方案来备份重要数据。

1. 进行用户管理备份

$ sqlplus system/123@pdborcl

SQL>SELECT file\_name FROM dba\_data\_files where tablespace\_name=’USERS’;

SQL>ALTER tablespace users begin backup;

SQL>!cp /home/oracle/app/oracle/oradata/orcl/pdborcl/SAMPLE\_SCHEMA\_users01.dbf

/home/oracle/SAMPLE\_SCHEMA\_users01.dbf

SQL>!cp /home/oracle/app/oracle/oradata/orcl/control01.ctl

/home/oracle/control01.ctl

1. 恢复数据库

SQL>!cp /home/oracle/SAMPLE\_SCHEMA\_users01.dbf

/home/oracle/app/oracle/oradata/orcl/pdborcl/SAMPLE\_SCHEMA\_users01.dbf

SQL>shutdown immediate

SQL>startup mount

SQL>recover database;