

**期末项目设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | 基于Oracle的员工管理数据库设计 | | |
| 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 | 吴夏煜 | 学号 | 201810414304 |
| 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

[第一章 系统需求简介 3](#_Toc7476)

[1.1 总体需求简单介绍 3](#_Toc2571)

[1.2 用户总体业务构造 3](#_Toc16896)

[1.3 其他要求 4](#_Toc12446)

[1.4 系统功能设想 4](#_Toc11134)

[第二章 需求分析 5](#_Toc2081)

[2.1 数据流图 5](#_Toc26144)

[2.1.1 系统的全局数据流图 5](#_Toc7519)

[2.1.2 系统的局部数据流图 6](#_Toc15644)

[2.2 数据字典 9](#_Toc8677)

[2.2.1数据流 10](#_Toc10808)

[2.2.2 数据存储 10](#_Toc9513)

[2.2.3 处理过程逻辑 11](#_Toc4301)

[2.2.4 数据项 11](#_Toc12268)

[第三章 概念设计 12](#_Toc1566)

[3.1 实体 12](#_Toc7112)

[3.2 系统局部E-R图 13](#_Toc14650)

[3.3系统全局 E-R图 15](#_Toc4099)

[第四章 逻辑设计 19](#_Toc24312)

[4.1 E-R图到关系模式的转换 19](#_Toc5661)

[4.2 关系模式的规范及调整 19](#_Toc27345)

[4.3 各个数据表的表结构设计 21](#_Toc3000)

[第五章 物理设计 23](#_Toc25516)

[5.1表空间管理 23](#_Toc9412)

[5.2 安全管理 24](#_Toc23504)

[5.3 各个基本对象的建立 25](#_Toc2077)

[5.3.1建立数据表 25](#_Toc3480)

[5.3.2 序列设计 27](#_Toc8693)

[5.3.3 索引设计 29](#_Toc27296)

[5.3.4视图设计 30](#_Toc25288)

[5.4用户管理 31](#_Toc6532)

[5.5 PL/SQL编程设计 32](#_Toc5800)

[5.5.1 游标管理 32](#_Toc32357)

[5.6 命名块对象管理 33](#_Toc8222)

[5.6.1 存储过程设计 33](#_Toc3284)

[5.6.2函数设计 34](#_Toc31485)

[5.6.3 触发器设计 35](#_Toc21971)

[5.7 数据库服务器性能优化 37](#_Toc20178)

[5.8 备份 38](#_Toc6914)

# 系统需求简介

## 总体需求简单介绍

需求分析阶段是数据库应用系统开发的最重要阶段。需求分析要求应用系统的开发人员按照系统的思想，根据收集的资料，对系统目标进行分析，对业务的信息需求、功能需求以及管理中存在的问题等进行分析，抽取本质的、整体的需求，为设计一个结构良好的数据库应用系统的逻辑模型奠定坚实的基础。

职工信息工资管理是公司管理的一项重要内容。随着公司员工数量增加，企业管理工作也变得越来越复杂。企业职工信息工资管理是一项琐碎、复杂而又十分细致的工作，一般不允许发生差错。最初的职工信息和工资统计都是用人工方式处理，工作量大的时候，出现错误的机率也随之升高，不仅花费大量的时间，而且往往由于抄写不慎，或者由于计算的疏忽，出现职工信息错误和工资统计错误的现象。对企业而言，全面开发和应用计算机管理信息系统就是近期不能回避的问题。如果职工的信息和工资情况能够实现计算机系统的管理，可以让人力资源管理人员从繁重琐碎的案头工作解脱出来，去完成更重要的工作。本职工信息工资管理系统可以减轻比较繁琐的手工职工信息和工资管理。

## 用户总体业务构造

职工信息工资系统主要为了方便企业管理职工的信息以及工资情况，提高人事信息的管理水平，简化，规范各机构的日常操作，记录职工的考勤情况，推算出职工的工资，并可作出相应的调整。职工信息工资管理系统要具有如下的功能：

1. 登录注册功能：使用数据库实现管理员的登录和添加
2. 部门管理功能：使用数据库实现部门基本信息的录入和修改
3. 职务管理功能：使用数据库实现职务基本信息的录入和修改
4. 职工管理功能：向数据库中添加职工的基本信息，职工的所在部门和职务名称只能添加部门管理模块和职务管理模块中存在的部门和职务。同时可对职工的基本信息进行查询。
5. 工资管理功能：实现职工工资的录入并判断输入的职工编号是否存在，修改职工的工资情况，查询指定员工的工资情况，并可从数据库中将职工的工资情况导出到本地文件以及上传到服务器上。
6. 考勤管理功能：实现职工考勤信息的录入、考勤信息的修改和查询指定职工的考勤信息。将考勤信息向工资管理模块反馈，实现工资的智能推荐。
7. 辅助管理模块：实现系统的退出和查看帮助

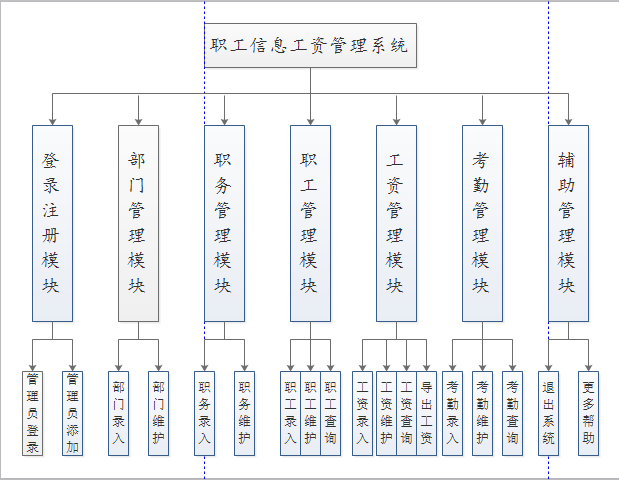
## 其他要求

如安全性，系统环境要求(根据现有的设备情况进行系统运行)等，这些不是本次作业的核心内容，所以就不再进一步叙述。很多中小型企业规模在逐步扩大，一个完善职工信息工资管理系统是十分有必要的，系统数据库设计的前提是公司需要这个系统，这个系统需要进行完善的数据库设计，这个系统能给该公司带来便捷的处理数据，系统数据库设计前提可行。

## 系统功能设想

这里的功能划分，是根据第一阶段需求调查基础上进行的初步划分。随着需求调查的深入，功能模块随着对需求了解的明确得到调整。

该系统设计了7大模块，分别是登录注册模块、部门管理模块、职务管理模块、职工管理模块、工资管理模块、考勤管理模块和辅助管理模块。根据各业务子系统所包括业务内容,还可以将各个子系统继续细化划分为更小的功能模块。划分的准则主要遵循模块的内聚性要求和模块间的低聚合性。如图所示表示一个职工信息工资管理系统功能模块结构图。



# 需求分析

## 2.1 数据流图

### 2.1.1 系统的全局数据流图

系统的全局数据流图，在具体的设计工具中往往也称为第0层或顶层数据流图，主要是从整体上描述系统的数据流，反映系统中数据的整体流向，是设计者针对用户和开发者表达出来的一个总体描述。

我们经过对职工信息工资管理业务的调查、数据的收集和信息流程分析处理，明确了该系统的主要功能，分别为：

我们经过对职工信息工资管理业务的调查、数据的收集和信息流程分析处理，明确了该系统的主要功能，分别为：公司管理员对部门信息、职务信息、职工信息、工资标准、考勤信息、职工的工资信息和用户信息进行进行维护管理。会计依据各个职务的工资标准和考勤管理生成工资清单，对职工工资进行发放处理。

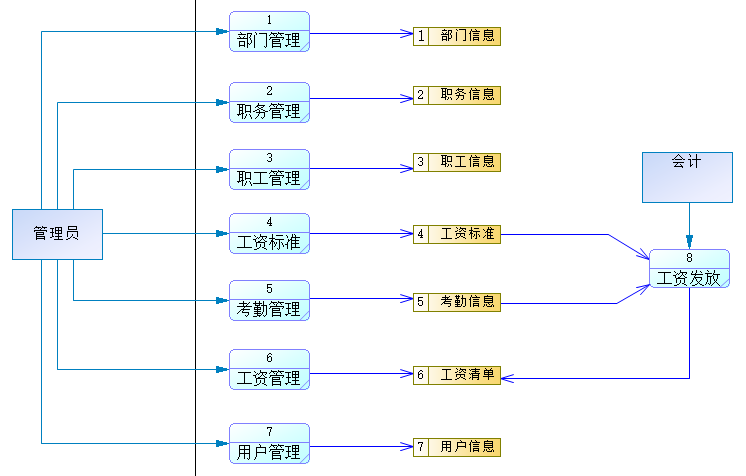


图2.1 顶层数据流图

### 2.1.2 系统的局部数据流图

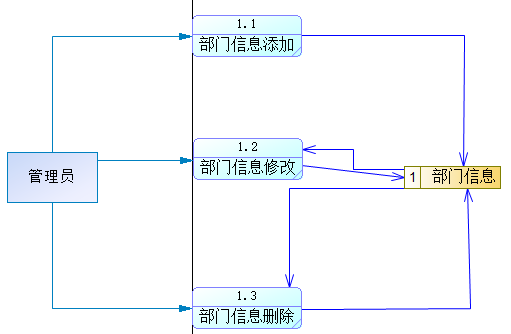


图2.2部门管理模块数据流图

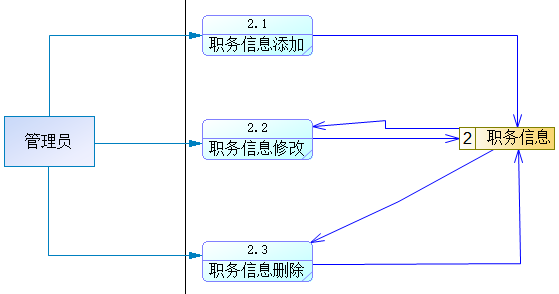


图2.3职务管理模块数据流图

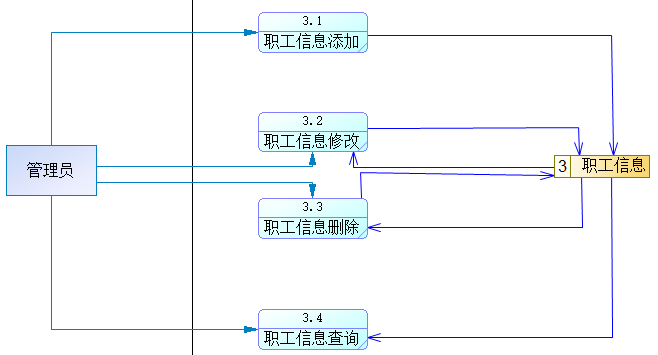


图2.4职工管理模块数据流图

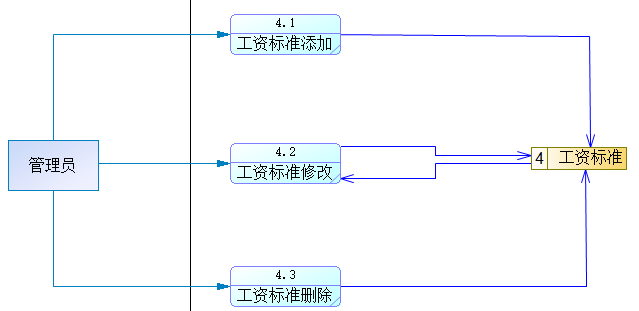


图2.5标准工资管理模块数据流图

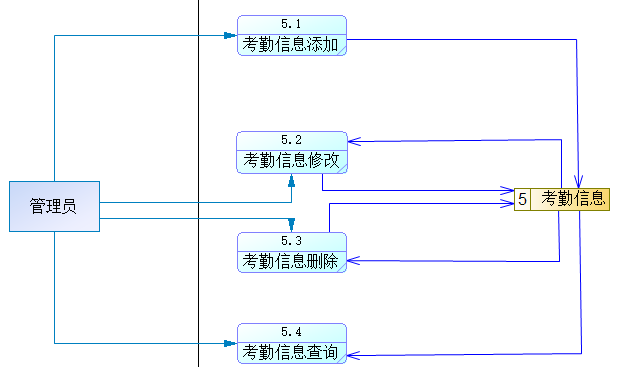


图2.6考勤管理模块数据流图

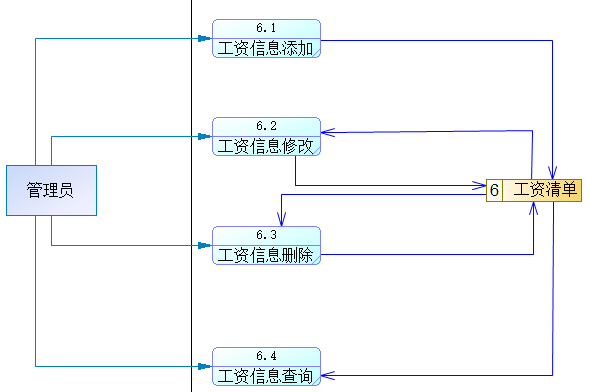


图2.7工资管理模块数据流图

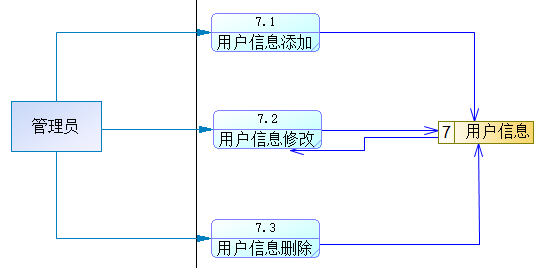


图2.8用户管理模块数据流图

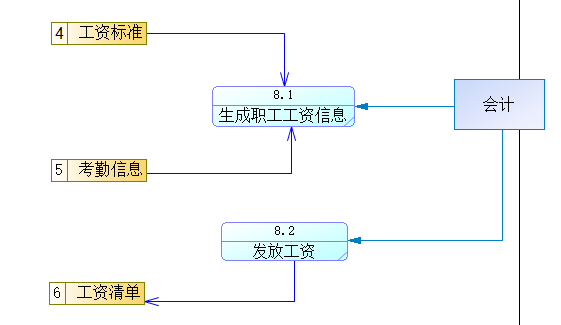


图2.9工资发放数据流图

## 2.2 数据字典

上面我们画的数据流图清楚的表达了数据也处理的关系，但是并没有清楚的描述各类数据的细节，我们这里通过数据字典来说明数据流图中出现的元素的详细定义和描述，包括数据流、加工处理、数据存储、数据的起点和终点或外部实体等。

数据字典包括的项目有：数据项、数据结构、数据流、数据存储、加工逻辑和外部实体。

因为整个系统的数据对象太多，所以这里我们列举订单管理模块中客户创建订单，管理员审核的处理功能中包含的几个对象加以描述。

### 2.2.1数据流

表2-1 职工信息工资管理模块数据流的描述

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 数据流名 | 来源 | 流向 | 组成 | 说明 |
| 1 | 管理员登录请求 | 需要登录的管理员 | 7 | 管理员登录名+管理员登录密码 |  |
| 2 | 管理员部门信息管理请求 | 管理员 | 1 | 部门编号+部门名称+部门电话+部门经理 |  |
| 3 | 管理员职务信息管理请求 | 管理员 | 2 | 职务编号+职务名称+所属部门 |  |
| 4 | 管理员职工信息管理请求 | 管理员 | 3 | 职工编号+职工姓名+所在部门+职务名称+职工年龄+职工电话+职工性别 |  |
| 5 | 管理员工资标准信息管理请求 | 管理员 | 4 | 职务编号+岗位工资+薪资等级+职务补贴 |  |
| 6 | 管理员考勤信息管理请求 | 管理员 | 5 | 职工编号+出勤天数+加班次数+出勤奖金 |  |
| 7 | 管理员工资信息管理请求 | 管理员 | 6 | 职工编号+职工姓名+推荐工资+实际工资 |  |

### 2.2.2 数据存储

表2-2职工信息工资管理模块数据存储的描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据文件 | 文件组成 | 关键标识 | 组织 |
| 1 | 部门信息 | 部门编号+部门名称+部门电话+部门经理 | 部门编号 | 按部门编号排序 |
| 2 | 职务信息 | 职务编号+职务名称+所属部门 | 职务编号 | 按职务编号排序 |
| 3 | 职工信息 | 职工编号+职工姓名+所在部门+职务名称+职工年龄+职工电话+职工性别 | 职工编号 | 按职工编号排序 |
| 4 | 工资标准信息 | 职务编号+岗位工资+薪资等级+职务补贴 | 职务编号 | 按职务编号排序 |
| 5 | 考勤信息 | 职工编号+出勤天数+加班次数+出勤奖金 | 职工编号 | 按职工编号排序 |
| 6 | 工资管理信息 | 职工编号+职工姓名+推荐工资+实际工资 | 职工编号 | 按职工编号排序 |
| 7 | 用户信息 | 用户名+手机号+密码 | 用户编号 | 按用户名排序 |

### 2.2.3 处理过程逻辑

表2-3职工工资信息管理模块数据处理过程逻辑的描述

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 处理过程 | 编号 | 输入 | 输出 | 处理逻辑 |
| 1 | 部门管理 | 1 | 部门信息 | 部门功能+无效部门信息 | 根据部门信息正确性 |
| 2 | 职务管理 | 2 | 职务信息 | 职务管理+无效职务信息 | 根据职务信息合理性 |
| 3 | 职工管理 | 3 | 职工信息 | 职工管理+无效职工信息 | 根据职工信息合理性 |
| 4 | 工资标准 | 4 | 工资标准清单 | 工资标准+无效工资信息清单 | 根据工资标准清单的合理性 |
| 5 | 考勤管理 | 5 | 考勤信息 | 考勤管理+无效考勤信息 | 根据考勤信息的存在性 |
| 6 | 工资管理 | 6 | 工资信息 | 工资管理+无效工资信息 | 根据工薪信息的合理性 |
| 7 | 用户管理 | 7 | 用户信息 | 用户管理+无效用户信息 | 根据用户信息的合理性 |

### 2.2.4 数据项

表2-4 职工工资信息管理模块数据项的说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 数据项 | 数据对象说明 | 数据构成 |
| 1 | 部门编号 | 1{英文|数字}5 |  |
| 2 | 部门名称 | {汉字}6 |  |
| 3 | 部门经理 | {汉字}6 |  |
| 4 | 部门电话 | {数字}11 |  |
| 5 | 职务编号 | 1{英文|数字}5 |  |
| 6 | 职务名称 | {汉字}6 |  |
| 7 | 职工编号 | 1{英文|数字}5 |  |
| 8 | 职工名称 | {汉字}6 |  |
| 9 | 职工姓名 | {汉字}6 |  |
| 10 | 职工性别 | {汉字}1 |  |
| 11 | 职工年龄 | {数字}3 |  |
| 12 | 职工电话 | {数字}11 |  |
| 13 | 出勤天数 | {数字}3 |  |
| 14 | 加班次数 | {数字}3 |  |
| 15 | 出勤奖金 | {数字}12，2 |  |
| 16 | 推荐工资 | {数字}12,2 |  |
| 17 | 实际工资 | {数字}12,2 |  |
| 英文=[‘a’…’z’|’A’…’Z’] | | | |
| 数字=[‘0’…’9’] | | | |

# 概念设计

上述的数据流图和数据字典共同构成了对用户需求的表达，它们是系统分析员(数据库管理员)在需求调查过程中和用户反复交互得到的。建设系统实际要处理的数据基本上已经在数据流图中得到体现，整个设计过程的后续步骤提供基础和依据。

概念设计就是通过对需求分析阶段所得到的信息需求进行综合、归纳与抽象，形成一个独立于具体数据库管理系统的概念模型，主要的手段为ER图。

在概念设计阶段，主要采用的设计手段目前还是实体联系模型(E-R Model)。绘制E-R图的关键是确定E-R图的各种结构，包括实体、属性和联系。大部分的流行建模工具(Power Designer、Oracle Designer、ERwin等)也都包含了对E-R设计手段的支持。

## 3.1 实体

要建立系统的E-R模型的描述，需进一步从数据流图和数据字典中提取系统所有的实体及其属性。这种提出实体的指导原则如下：

1.属性必须是不可分的数据项，即属性中不能包含其它的属性或实体

2.E-R图中的关联必须是实体之间的关联，属性不能和其它实体之间有关联

由前面分析得到的数据流图和数据字典，可以抽象得到实体主要有7个：部门、职工、职务、管理员、工资标准、考勤信息、工资情况。

* 部门实体属性有：部门编号、部门名称、部门经理、部门电话。
* 职工实体属性有：职工编号， 职工姓名，所在部门，职务名称，职工年龄，职工电话，职工性别。
* 职务实体属性有：职务编号，职务名称，所属部门。
* 考勤信息实体属性有：职工编号， 出勤天数，加班次数，出勤奖金。
* 工资标准实体属性有：职务编号，岗位工资，薪资等级，职务补贴。
* 工资信息实体属性有：职工编号，职工姓名，推荐工资，实际工资。
* 管理员实体属性有：手机号，用户名，密码 。

## 3.2 系统局部E-R图

在需求分析阶段我们采用的是自上而下的分析方法，那么要在其基础上进一步作概念设计我们面临的是细化的分析数据流图以及数据字典，分析得到实体及其属性后，进一步可分析各实体之间的联系。

* 部门实体和职工实体存在归属的联系，一个部门可以有多个职工，而每个职工只归属于一个部门，所以它们之间是一对多的联系(1:n)。
* 职工实体和职务实体存在归属的联系，一个职务可以有多个职工担任，但一个职工只能担任一个职务，所以它们之间是一对多的联系(1:n)。
* 工资标准实体和职工实体存在归属的联系，一个工资标准可以对应多个职工，但每个职工只对应一个工资标准，所以它们之间是一对多的联系(1:n)。
* 职工和考勤信息存在一对一的关系，一个职工对应一份考勤信息，一份考勤信息对应一份职工，它们之间的联系为(1:1)。

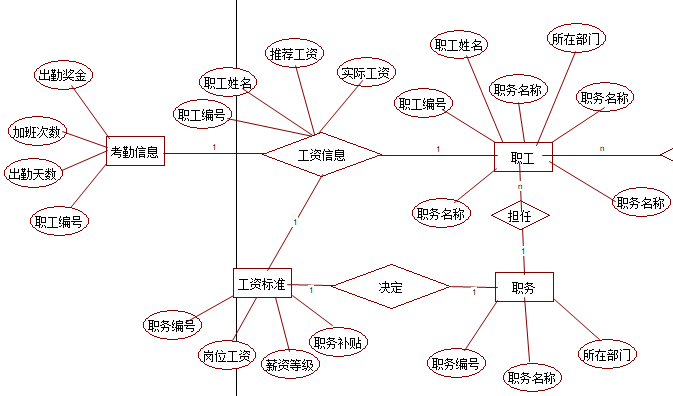


图3.1 工资局部E-R图

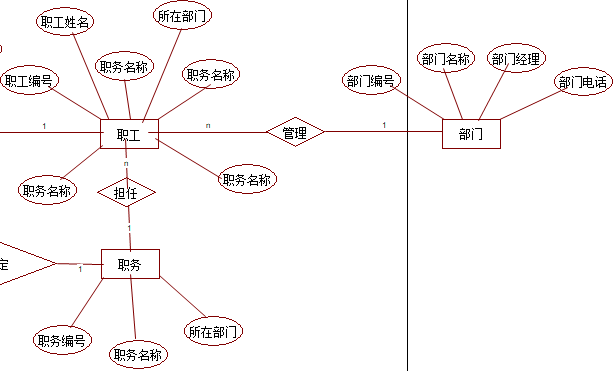


图3.2 职工信息局部E-R图

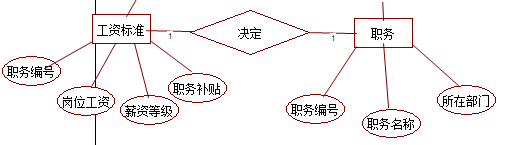


图3.3 工资标准局部E-R图

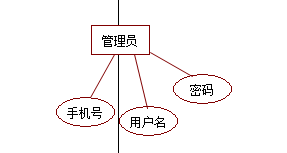


图3.4 管理员局部E-R图

## 3.3系统全局 E-R图

系统的局部E-R图，仅反映系统局部实体之间的联系，但无法反映系统在整体上实体间的相互联系。而对于一个比较复杂的应用系统来说，这些局部的E-R图往往有多人各自分析完成的，只反映局部的独立应用的状况，在系统整体的运作需要时，他们之间有可能存在重复的部分或冲突的情况，如实体的划分、实体或属性的命名不一致等，属性的具体含义(包括数据类型以及取值范围等不一致)问题，都可能造成上述提到的现象。

为解决这些问题，必须理清系统在应用环境中的具体语义，进行综合统一，通过调整消除那些问题，得到系统的全局E-R图。

各局部E-R存在不少的重复的实体，经过上述聚集分析和合并得到系统全局的E-R图如图3.5所示。该全局E-R图基本上不存在关系的冗余状况，因此它已经是一个优化的。

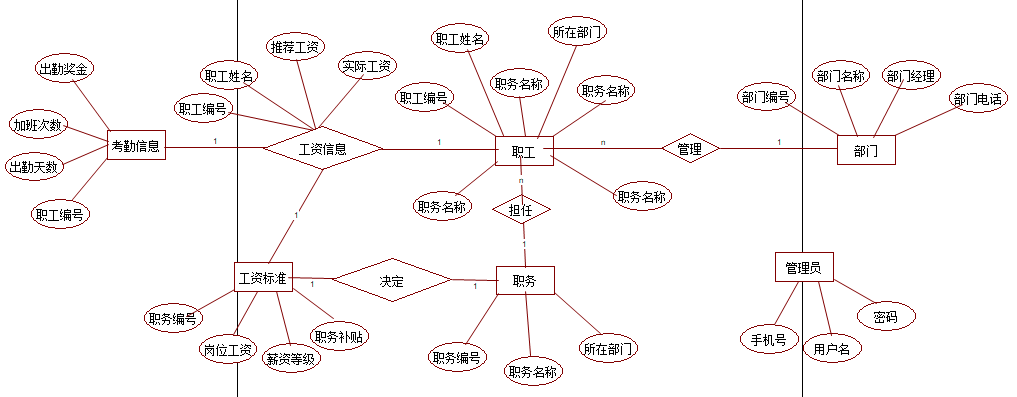


图3.5 职工信息工资管理系统的全局E-R图

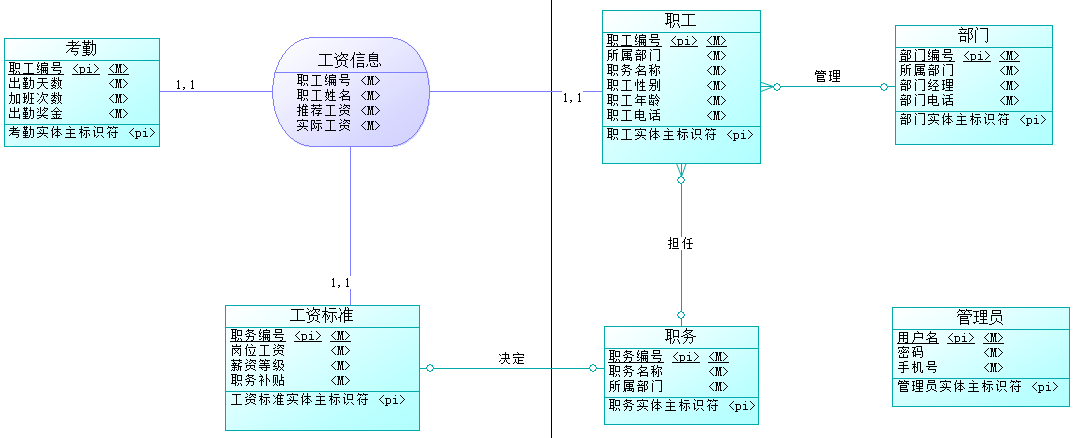


图3-6 职工信息工资系统的全局概念图

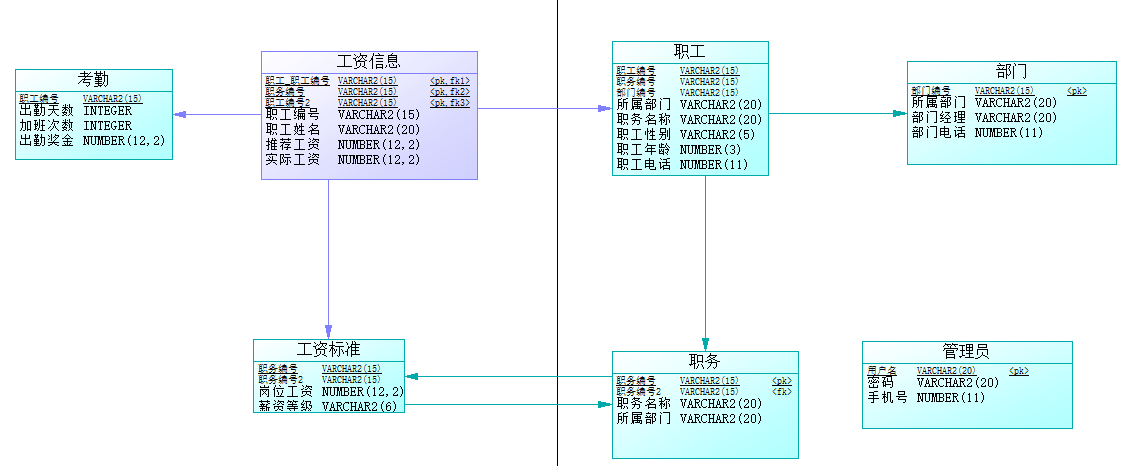


图3-7 职工信息工资系统数据库物理模型设计图

# 逻辑设计

## 4.1 E-R图到关系模式的转换

本系统功能庞大，实体众多，这里列举重要的关系模式的转换过程：

部门实体和职工实体存在归属的联系，一个部门可以有多个职工，而每个职工只归属于一个部门，所以它们之间是一对多的联系(1:n)。设计成如下的的关系模式：

* 部门实体属性有：部门编号、部门名称、部门经理、部门电话。
* 职工实体属性有：职工编号， 职工姓名，所在部门，职务名称，职工年龄，职工电话，职工性别。

 职工实体和职务实体存在归属的联系，一个职务可以有多个职工担任，但一个职工只能担任一个职务，所以它们之间是一对多的联系(1:n)。

* 职工实体属性有：职工编号， 职工姓名，所在部门，职务名称，职工年龄，职工电话，职工性别。
* 职务实体属性有：职务编号，职务名称，所属部门。

 工资标准实体和职工实体存在归属的联系，一个工资标准可以对应多个职工，但每个职工只对应一个工资标准，所以它们之间是一对多的联系(1:n)。

* 工资标准实体属性有：职务编号，岗位工资，薪资等级，职务补贴。
* 职工实体属性有：职工编号， 职工姓名，所在部门，职务名称，职工年龄，职工电话，职工性别。

 职工和考勤信息存在一对一的关系，一个职工对应一份考勤信息，一份考勤信息对应一份职工，它们之间的联系为(1:1)。

* 职工实体属性有：职工编号， 职工姓名，所在部门，职务名称，职工年龄，职工电话，职工性别。
* 考勤信息实体属性有：职工编号， 出勤天数，加班次数，出勤奖金。

## 4.2 关系模式的规范及调整

在提出关系模式后，我们必须在规范化和实际要求进行优化，这实际上是一个权衡的过程。如果设计没有完全规范化，如可能用于决策支持(与需要大量更新的事务处理相对)的数据库(如数据仓库)则可能没有冗余更新，而且可能对查询更易于理解和更高效。不过，在数据库应用程序内，未规范化的数据在设计过程更需要注意。一般的策略是以规范化设计为出发点，然后出于特定因素有条件地非规范化某些表,以达到系统总体的优化目的。

首先，需要我们确定上面建立的关系模式中的函数依赖，一般在作需求分析时就了解到一些数据项的依赖关系，如教师的编号决定了教师的姓名和其它的数据项信息，而实体间的联系本身也是反映了一种函数依赖关系，但是这不是研究的对象，我们针对的是在一个关系模式中的函数依赖对象。

其次，对上一步确立的所有函数依赖进行检查，判别是否存在部分函数依赖以及传递函数依赖，针对有的依赖通过投影分解，消除在一个关系模式中存在的部分函数依赖和传递函数依赖。

大部分数据库系统只要满足第三关系范式就可以，这也是我们这里规范化的基本要求。由于需求分析阶段的方法得当，经过简单的分析可以看出，上述所有关系中每个数据项都是基本的，任何非主属性都不存在对主码的部分依赖，也不存在非主属性存在着对主码的传递依赖。可见，以上所有的关系模式都属于3NF。

在实际的应用中，关系模式的规范化程度并不是越高越好，因为在关系模式的规范化提升过程中，必须进行着将一个关系模式分解成为多个关系模式的过程。这样，在以后执行查询时，如果需要相关的信息，就必须作多个表的连接方能达到查询的目的，这无疑给系统增加一定的开销，特别存在很多用户同时访问或者关系中存在许多元组等因素其负担会越加明显。为了兼顾性能的需要，在适当的时候可能需要对相关程度比较高的一些关系模式进行合并处理，或者在关系模式中增加相关程度比较高的属性等。这是有可能选择第二范式甚至第一范式。

如果系统存在很多的元组数(记录数)，特别当记录达到百万甚至千万条记录时，系统的查询效率可能会受到明显的影响，分析关系模式的特点，可以根据某些关键属性不同将关系模式分解为多个关系模式，模式之间通过共同的主键一一对应起来。

## 4.3 各个数据表的表结构设计

在上述经由E-R模型得到关系模式并且得到适当的调整后，我们可以结合在需求表述中数据字典包含的数据项信息，得到数据库的表结构。

表4-1至表4-15给出了数据库中各个数据表的表结构。

数据库表清单，记录各个表，详细信息如表4-1。

表4-1 数据库表清单

|  |  |  |
| --- | --- | --- |
| 数据库表名 | 关系模式名称 | 备注 |
| user | 管理员 | 管理员表 |
| department | 部门 | 部门信息表 |
| staff | 职工 | 职工信息表 |
| job | 职务 | 职务信息表 |
| attcheck | 考勤信息 | 考勤信息表 |
| wagelevel | 工资标准 | 工资标准明细表 |
| wage | 工资信息 | 工资信息表 |

* 管理员基本信息表：用于记录管理员的用户名，密码，手机号，详细信息如表4-2所示。

表4-2 管理员表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 字段大小 | 是否主键 | 是否为空 | 说明 |
| username | varchar2 | 20 | 是 | 否 | 管理员用户名 |
| password | varchar2 | 20 | 否 | 否 | 管理员密码 |
| phone | number(11) | 11 | 否 | 否 | 管理员手机号 |

* 部门信息表：用于记录部门的基本信息，如部门编号，部门名称，部门经理，部门电话等信息，详细信息如表4-3所示。

表4-3部门信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 字段大小 | 是否主键 | 是否为空 | 说明 |
| deptnum | varchar2 | 15 | 是 | 否 | 部门编号 |
| deptname | varchar2 | 20 | 否 | 否 | 部门名称 |
| manager | varchar2 | 20 | 否 | 否 | 部门经理 |
| tel | number(11) | 11 | 否 | 否 | 部门电话 |

* 职务信息表：用于记录职务的基本信息，包括职务编号，职务名称，所属部门，详细信息如表4-3所示。

表4-3职务信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 字段大小 | 是否主键 | 是否为空 | 说明 |
| jobnum | varchar2 | 15 | 是 | 否 | 职务编号 |
| jobname | varchar2 | 20 | 否 | 否 | 职务名称 |
| deptname | varchar2 | 20 | 否 | 否 | 所属部门 |

* 职工信息表：用于记录职工的基本信息，包括职工编号，职工姓名，所在部门，职务名称，职工年龄，职工电话，职工性别。详细信息如表4-4所示。

表4-4职工信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 字段大小 | 是否主键 | 是否为空 | 说明 |
| staffnum | varchar2 | 15 | 是 | 否 | 职工编号 |
| staffname | varchar2 | 20 | 否 | 否 | 职工姓名 |
| deptname | varchar2 | 20 | 否 | 否 | 所在部门 |
| jobname | varchar2 | 20 | 否 | 否 | 职务名称 |
| staffsex | varchar2 | 5 | 否 | 否 | 职工性别 |
| staffage | number(3) | 3 | 否 | 否 | 职工年龄 |
| stafftel | number(11) | 11 | 否 | 否 | 职工电话 |

* 考勤信息表：用于记录考勤的基本信息，如职工编号，出勤天数，加班次数，出勤奖金等信息，详细信息如表4-5所示。

表4-5考勤信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 字段大小 | 是否主键 | 是否为空 | 说明 |
| staffnum | varchar2 | 15 | 是 | 否 | 职工编号 |
| attday | integer |  | 否 | 否 | 出勤天数 |
| overtime | integer |  | 否 | 否 | 加班次数 |
| prize | number(12,2) | 12 | 否 | 否 | 出勤奖金 |

* 工资标准明细表：用于记录工资标准的基本信息，如职务编号，岗位工资，薪资等级，职务补贴等信息，详细信息如表4-6所示。

表4-6工资标准明细表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 字段大小 | 是否主键 | 是否为空 | 说明 |
| jobnum | varchar2 | 15 | 是 | 否 | 职务编号 |
| basicwage | number(12,2) | 12 | 否 | 否 | 岗位工资 |
| wagegrade | varchar2 | 6 | 否 | 否 | 薪资等级 |
| allowance | number(12,2) | 12 | 否 | 否 | 职务补贴 |

* 工资信息表：用于记录职工工资的基本信息，如职工编号，职工姓名，推荐工资，实际工资等信息，详细信息如表4-7所示。

表4-7工资信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 字段大小 | 是否主键 | 是否为空 | 说明 |
| staffnum | varchar2 | 15 | 是 | 否 | 职工编号 |
| staffname | varchar2 | 20 | 否 | 否 | 职工姓名 |
| referwage | number(12,2) | 12 | 否 | 否 | 推荐工资 |
| realwage | number(12,2) | 12 | 否 | 否 | 实际工资 |

# 物理设计

## 5.1表空间管理

为了不影响数据库的稳定性和执行效率，本系统会建立相应的表空间。表空间管理主要包括表空间的创建、修改、删除和查询。本系统利用SQL语句进行管理。

在实际应用中，我们通常需要为系统创建用于保存数据库表的数据表空间、存放数据索引的索引表空间和专用的临时表空间。本系统“职工信息工资管理系统”有约7张数据表，

通过测算，数据表空间大约需要4GB，估算索引表空间容量和数据表空间同样大小也为4GB，临时表空间容量设置为1GB，回退表空间容量设置为1GB。

综上所述，本系统需要创建一个数据表空间、1个索引表空间、1个临时表空间、一个回退表空间（由于在自己电脑上作业测试，我给每个表空间的容量缩小4倍，故就不建那么大容量的表空间，以下是系统需要的建立表空间语句）。

创建表空间的语句如下：

*--创建数据表空间，1个数据文件1GB，允许自动扩容。*

create tablespace staffbank\_data datafile 'e:\database\staffbank\_data01.dbf' size 1G autoextend on

next 200M maxsize 1G

*--创建索引表空间，1个数据文件1GB，允许自动扩容。*

create tablespace staffbank\_idx datafile 'e:\database\staffbank\_idx01.dbf' size 1G autoextend on

next 200M maxsize 1G

*--创建临时表空间，1个数据文件250MB，允许自动扩容。*

create temporary tablespace staffbank\_temp tempfile 'e:\database\staffbank\_temp01.dbf' size 250M autoextend on

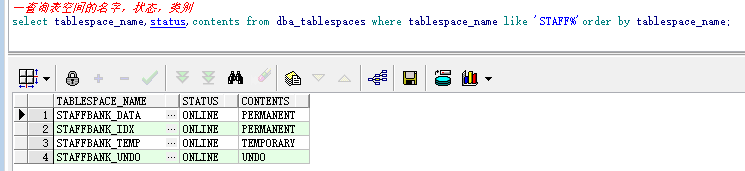
next 50M maxsize 250M

*--创建回退表空间，1个数据文件250MB，允许自动扩容。*

create undo tablespace staffbank\_undo datafile 'e:\database\staffbank\_undo01.dbf' size 250M autoextend on

next 50M maxsize 250M

查询刚才为系统所创建的表空间语句：



## 5.2 安全管理

Oracle作为大型数据库管理系统，其安全性一直是设计的重要目标，他拥有自己完备的安全体系，提供了一整套强大的安全管理手段，作为Oracle安全模型基础的安全原理是基于最小特权的原则，此原则认为用户只应该具有完成其任务所必须的特权，而不应该拥有更多的特权。

（1）创建用户admin,admin用户可以登录到oracle数据库中，创建属于自己的数据库对象，对于数据库开发用户只需授予connect和resource两个角色就可以在用户的方案中创建自己的所有数据库对象，故给admin授予这两个预定义角色；并且授予查询、删除、添加、修改、创建视图、创建触发器等功能。

*--创建用户admin,密码为admin123，可以无限额使用索引表空间staffbank\_idx,临时表空间为staffbank\_temp*

create user admin default tablespace staffbank\_data identified by admin123 quota unlimited on staffbank\_data

quota unlimited on staffbank\_idx temporary tablespace staffbank\_temp;

*--将预定义角色connect,resource授予用户admin*

grant connect,resource to admin

（2）创建用户genuser,给genuser用户授予连接数据库的权限，genuser用户只能登录进oracle系统，但是不能进行实质性操作，还需要等admin用户创建完自己的对象后，再将相应的数据库对象的查询权限授予genuser。同义词是数据库方案对象的一个别名，经常用于简化对象访问和提高对象访问的安全性，故将synonym权限授予genuser。

*--创建用户genuser,密码为genuser123，不需要为他指定表空间，因为该用户不能自己创建自己的数据对象*

create user genuser identified by genuser123;

*--修改admin用户密码,将密码改为admin111*

alter user admin identified by admin111;

## 5.3 各个基本对象的建立

### 5.3.1建立数据表

管理员表

*--创建user表*

create table admin.users(

username varchar2(20) not null primary key,

password varchar2(20) not null,

phone number(11) not null

);

*--给user表加注释*

comment on table admin.users IS '管理员信息';

comment on column admin.users.username IS '管理员用户名，作为主键';

comment on column admin.users.password IS '管理员密码';

comment on column admin.users.phone IS '管理员手机号';

部门信息表

*--创建department表*

create table admin.department(

deptnum varchar2(15) not null primary key,

deptname varchar2(20) not null,

manager varchar2(20) not null,

tel number(11) not null

);

*--给department表加注释*

comment on table admin.department IS '部门信息';

comment on column admin.department.deptnum IS '部门编号，作为主键';

comment on column admin.department.deptname IS '部门名称';

comment on column admin.department.manager IS '部门经理';

comment on column admin.department.tel IS '部门电话';

职工信息表

*--创建staff表*

create table admin.staff(

staffnum varchar2(15) not null primary key,

staffname varchar2(20) not null,

deptname varchar2(20) not null,

jobname varchar2(20) not null,

staffsex varchar2(5) not null,

staffage number(3) not null,

stafftel number(11) not null

);

*--给staff表添加注释*

comment on table admin.staff IS '职工信息';

comment on column admin.staff.staffnum IS '职工编号，作为主键';

comment on column admin.staff.staffname IS '职工姓名';

comment on column admin.staff.deptname IS '所在部门';

comment on column admin.staff.jobname IS '职务名称';

comment on column admin.staff.staffsex IS '职工性别';

comment on column admin.staff.staffage IS '职工年龄';

comment on column admin.staff.stafftel IS '职工电话';

职务信息表

*--创建job表*

create table admin.job(

jobnum varchar2(15) not null primary key,

jobname varchar2(20) not null,

deptname varchar2(20) not null

);

*--给job表加注释*

comment on table admin.job IS '职务信息';

comment on column admin.job.jobnum IS '职务编号，作为主键';

comment on column admin.job.jobname IS '职务名称';

comment on column admin.job.deptname IS '所属部门';

考勤信息表

*--创建attcheck表*

create table admin.attcheck(

staffnum varchar2(15) not null primary key,

attday integer not null,

overtime integer not null,

prize number(12,2) not null

);

*--给attcheck表添加注释*

comment on table admin.attcheck IS '考勤信息';

comment on column admin.attcheck.staffnum IS '职工编号，作为主键';

comment on column admin.attcheck.attday IS '出勤天数';

comment on column admin.attcheck.overtime IS '加班次数';

comment on column admin.attcheck.prize IS '出勤奖金';

工资标准明细表

*--创建wagelevel表*

create table admin.wagelevel(

jobnum varchar2(15) not null primary key,

basicwage number(12,2) not null,

wagegrade varchar2(6) not null,

allowance number(12,2) not null

);

*--给wagelevel表加注释*

comment on table admin.wagelevel IS '考勤信息';

comment on column admin.wagelevel.jobnum IS '职务编号，作为主键';

comment on column admin.wagelevel.basicwage IS '岗位工资';

comment on column admin.wagelevel.wagegrade IS '薪资等级';

comment on column admin.wagelevel.allowance IS '职务补贴';

工资信息表

*--创建wage表*

create table admin.wage(

staffnum varchar2(15) not null primary key,

staffname varchar2(20) not null,

referwage number(12,2) not null,

realwage number(12,2) not null

);

*--给wage表加注释*

comment on table admin.wage IS '工资信息';

comment on column admin.wage.staffnum IS '职工编号，作为主键';

comment on column admin.wage.staffname IS '职工姓名';

comment on column admin.wage.referwage IS '推荐工资';

comment on column admin.wage.realwage IS '实际工资';

### 5.3.2 序列设计

序列是可被多个用户使用的用于产生一系列唯一整数的数据库对象，序列是一个连续的数字生成器，使用序列的好处是自动产生主键的键值，从而简化用户的输入工作量，根据分析，本系统的部门主键deptnum和职工主键staffnum可以考虑生成序列。生成序列的语句如下：

*--设置序列seq\_deptnum,初始值为10000,以1增长,不重复不预先分配*

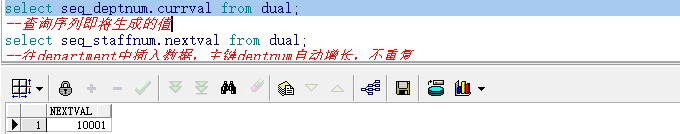
create sequence seq\_deptnum start with 10000 increment by 1 nocache nocycle;

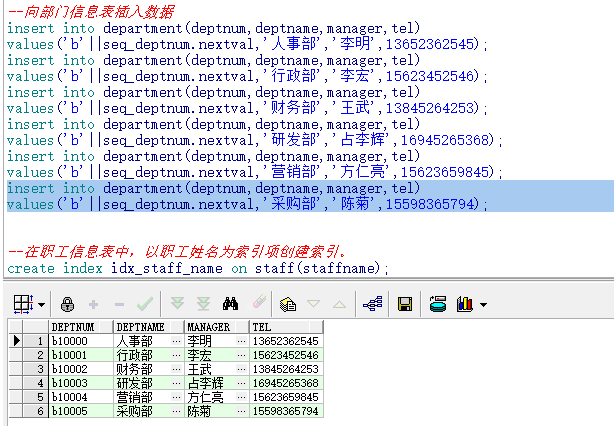
*--设置序列seq\_staffnum,初始值为10000,以1增长,不重复不预先分配*

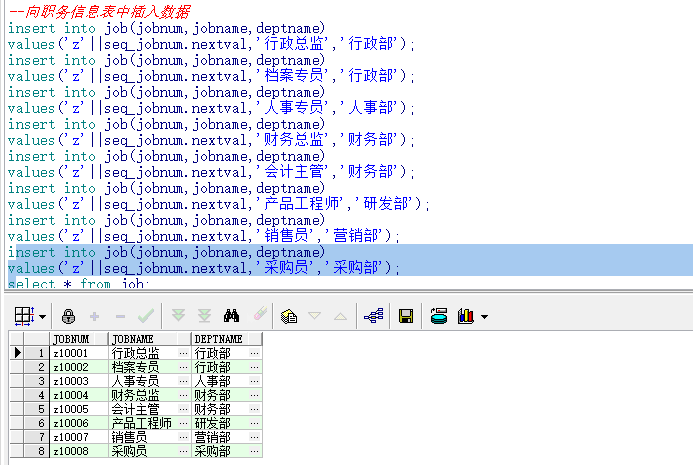
create sequence seq\_staffnum start with 10000 increment by 1 nocache nocycle;

*--设置序列seq\_jobnum,初始值为10000，以1增长，不重复不预先分配*

create sequence seq\_jobnum start with 10000 increment by 1 nocache nocycle;









### 5.3.3 索引设计

索引是一种可以提高查询性能的数据结构，使用索引有两个好处，一个是快速查询，一个是唯一值。虽然使用索引能提高查询速度，但是如果为了某个表创建了过多的索引，反而可能降低修改、插入、删除等操作的速度，因此，在进行应用设计时，一定要认真考虑，决定在什么表上创建索引、创建什么索引，创建几个索引等问题。索引的设计不但需要仔细考虑查询功能的特点以及数据变更的权衡因素，而且也需要结合整体数据量的大小和增长的可能等（注意：本部分没有对数据量大小作分析），但是关键还是实际的性能需求。

*--在职工信息表中，以职工姓名为索引项创建索引。*

create index idx\_staff\_name on staff(staffname);

*--在职工信息表中，以职务名称为索引项创建索引*

create index idx\_staff\_job on staff(jobname);

*--在部门信息表中，以部门名称为索引项创建索引。*

create index idx\_department\_name on department(deptname);

*--在工资信息表中，以职工姓名为索引项创建索引*

create index idx\_wage\_staffname on wage(staffname);

### 5.3.4视图设计

视图是从一个或多个表中通过查询而导出数据的虚表，Oracle仅仅存储了视图的定义，并不存储视图对应的存储查询所涉及到的数据，所以建立视图不占用其他空间。使用视图有三个好处，分别为安全性、方便性、一致性。视图和表一样由列组成，其查询方式与表完全相同。结合以上分析，以下是基于本系统的视图设计。

*--将system用户中将创建视图的权限给admin用户*

grant create view to admin;

*--创建一个职工信息视图，视图包含职工编号，职工姓名，所在部门，职务名称，职工年龄，职工电话，*

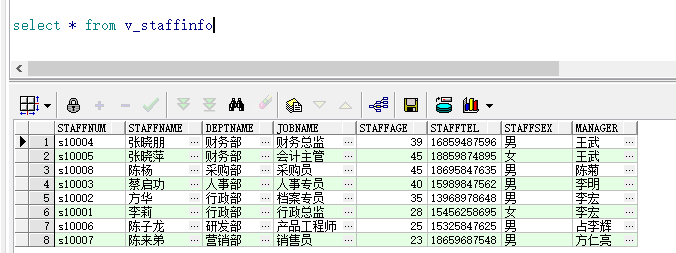
*--性别，所在部门经理*

create or replace view v\_staffinfo(staffnum,staffname,deptname,jobname,staffage,stafftel,staffsex,manager)

as select

s.staffnum,s.staffname,s.deptname,s.jobname,s.staffage,s.stafftel,s.staffsex,d.manager

from staff s,department d where s.deptname=d.deptname;



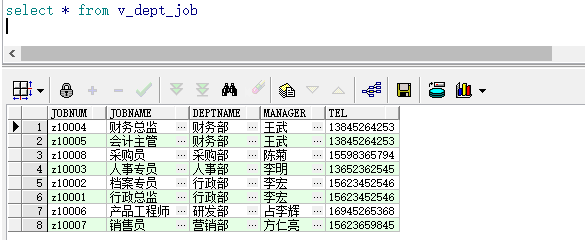
*--创建一个部门职务信息视图，视图包含职务编号，职务名称，所在部门，部门经理，部门电话*

create or replace view v\_dept\_job(jobnum,jobname,deptname,manager,tel)

as select

j.jobnum,j.jobname,j.deptname,d.manager,d.tel

from job j,department d where j.deptname=d.deptname;



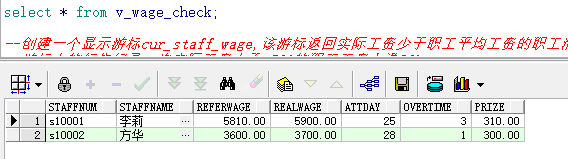
*--创建一个工资考勤视图，视图包含职工编号，职工姓名，推荐工资，实际工资，出勤天数，加班次数，出勤奖金*

create or replace view v\_wage\_check(staffnum,staffname,referwage,realwage,attday,overtime,prize)

as select

w.staffnum,w.staffname,w.referwage,w.realwage,c.attday,c.overtime,c.prize

from wage w,attcheck c where w.staffnum=c.staffnum;



## 5.4用户管理

用户：

staff\_manage

拥有角色：s\_manage

分配表空间：staff\_manage、staff\_user、staff\_public

staff\_user

拥有角色：s\_user

分配表空间：staff\_user、staff\_public

角色：

s\_manage

拥有角色：connect、resource、dba

s\_user

拥有角色：connect、resource

-- 创建角色   
CREATE ROLE s\_manage;   
CREATE ROLE s\_user;   
-- 授权角色   
GRANT connect,resource,dba, create table,create view,create trigger, create procedure,create sequence TO s\_manage;   
GRANT connect,resource, create table,create view,create trigger, create procedure,create sequence TO s\_user;   
-- 创建用户   
CREATE USER staff\_manage IDENTIFIED BY 123 DEFAULT TABLESPACE staff\_manage;   
-- 指定用户额外表空间   
ALTER USER staff\_manage QUOTA UNLIMITED ON staff\_user;   
ALTER USER staff\_manage QUOTA UNLIMITED ON staff\_public;   
-- 创建用户   
CREATE USER staff\_user IDENTIFIED BY 123 DEFAULT TABLESPACE staff\_user;   
-- 指定用户额外表空间   
ALTER USER staff\_user QUOTA UNLIMITED ON staff\_public ;   
-- 分配角色给用户   
GRANT s\_manage TO staff\_manage;   
GRANT s\_user TO staff\_user;

## 5.5 PL/SQL编程设计

### 5.5.1 游标管理

游标是一种Oracle的一种内存结构，用来存放SQL语句或程序执行的结果。游标使用select语句从基表或视图中取出数据生成行集放入内存，最初游标指针指向查询结果的行集的首部，随着游标的推进，就可以访问相应的记录。

*--创建一个显示游标cur\_staff\_wage,该游标返回实际工资少于职工平均工资的职工清单信息，然后逐个读取*

*--游标中的行集纪录，将实际工资少于1500的职工工资上调20%*

declare

cursor cur\_staff\_wage (under\_wage Number) return wage%rowtype is

select \* from wage where realwage<under\_wage;

rowwage wage%rowtype;

begin

open cur\_staff\_wage(1500);

if cur\_staff\_wage%isopen then

dbms\_output.put\_line('此游标共有'||cur\_staff\_wage%rowcount||'行记录。');

end if;

fetch cur\_staff\_wage into rowwage;

if cur\_staff\_wage%isopen then

dbms\_output.put\_line('此游标共有'||cur\_staff\_wage%rowcount||'行记录。');

end if;

while cur\_staff\_wage%found loop

update wage set realwage=realwage+realwage\*0.2 where staffnum=rowwage.staffnum;

fetch cur\_staff\_wage into rowwage;

end loop;

if cur\_staff\_wage%isopen then

dbms\_output.put\_line('此游标共有'||cur\_staff\_wage%rowcount||'行记录。');

end if;

close cur\_staff\_wage;

end;

## 5.6 命名块对象管理

### 5.6.1 存储过程设计

过程是为了执行一定任务而组合在一起的PL/SQL语句，使用过程的好处有以下四点。

* 模块化：每个过程完成一个相对独立的功能，提高了应用程序的模块独立性。
* 信息隐藏：调用过程的应用程序只需知道该过程做什么，而无需知道怎么做。
* 可重用性：过程可被多次重用。
* 较高的性能：过程是在服务器上执行，大大降低了网络流量，提高了运行性能。

*\*存储过程aver\_age,统计指定性别员工的平均年龄*

*输入参数：性别*

*输出参数：平均年龄*

*\*/*

create or replace procedure aver\_age

(

sex varchar2,*--性别*

average out number*--平均年龄*

)

is

begin

*--平均年龄*

select AVG(staffage) into average

from staff where staffsex=sex;

end;

*--执行该存储过程*

declare

sex varchar2(4):='男';

average1 number;

begin

aver\_age('男',average1);

dbms\_output.put\_line('男性的平均年龄'||average1);

insert into test (sex) values(average1);

end;

### 5.6.2函数设计

*--定义函数fun\_dept\_sumwage,功能是计算指定部门的所有员工的应发放的工资总和*

create or replace function fun\_dept\_sumwage

(

deptname varchar2

)return number

is

sumwage number(12,2);*--工资总和*

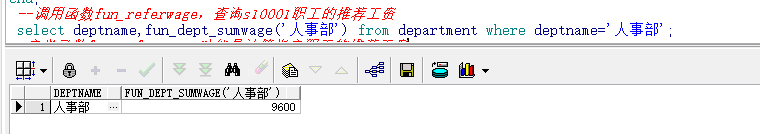
begin

select sum(realwage) into sumwage from wage w,staff s where w.staffnum=s.staffnum

and s.deptname=deptname;

return sumwage;

end;



*--定义函数fun\_referwage,功能是计算指定职工的推荐工资*

create or replace function fun\_referwage

(

staffnum1 varchar2

)return number

is

referwage1 number(12,2);*--推荐工资*

begin

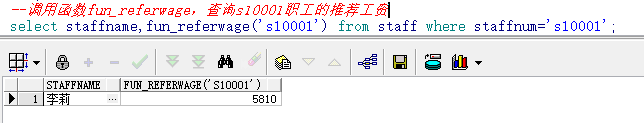
select sum(wagelevel.basicwage+wagelevel.allowance+attcheck.prize) into referwage1

from job join staff on job.jobname=staff.jobname join wagelevel on wagelevel.jobnum=job.jobnum

join attcheck on staff.staffnum=attcheck.staffnum where staff.staffnum=staffnum1;

return referwage1;

end;



### 5.6.3 触发器设计

*--创建基于考勤表的触发器tri\_prize,自动算出考勤表中的出勤奖金*

create or replace trigger tri\_prize before insert on attcheck

for each row

begin

:new.prize := :new.attday \*10 + :new.overtime \*20;

end;

drop trigger tri\_prize;

*--根据已经创建好的序列seq\_staffnum,在职工信息表staff上定义触发器；*

*--当在此表中插入记录时读取序列seq\_staffnum的nextval值，并将此值赋予给staff的staffnum字段*

*--完成主键序列的自动增长*

create or replace trigger tri\_staffnum\_auto before insert on staff

for each row

begin

select seq\_staffnum.nextval into :new.staffnum from dual;

:new.staffnum:=('s'||:new.staffnum);

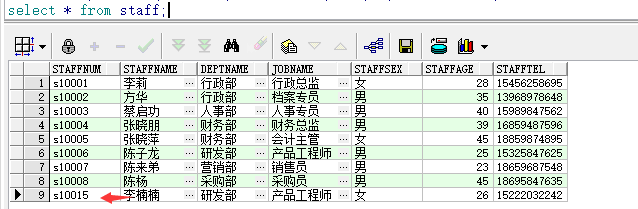
end;

*--向职工表中插入数据验证触发器*

insert into staff(staffnum,staffname,deptname,jobname,staffsex,staffage,stafftel)

values('s','李楠楠','研发部','产品工程师','女',26,15222032242);

select \* from staff;



*--根据已经创建好的序列seq\_deptnum,在部门信息表department上定义触发器；*

*--当在此表中插入记录时读取序列seq\_deptnum的nextval值，并将此值赋予给department的deptnum字段*

*--完成主键序列的自动增长*

create or replace trigger tri\_deptnum\_auto before insert on department

for each row

begin

select seq\_deptnum.nextval into :new.deptnum from dual;

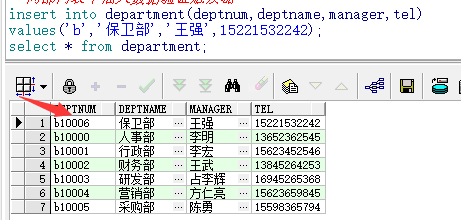
:new.deptnum:=('b'||:new.deptnum);

end;

*--向部门表中插入数据验证触发器*

insert into department(deptnum,deptname,manager,tel)

values('b','保卫部','王强',15221532242);



*--根据已经创建好的序列seq\_jobnum,在部门信息表job上定义触发器；*

*--当在此表中插入记录时读取序列seq\_jobnum的nextval值，并将此值赋予给job的jobnum字段*

*--完成主键序列的自动增长*

create or replace trigger tri\_jobnum\_auto before insert on job

for each row

begin

select seq\_jobnum.nextval into :new.jobnum from dual;

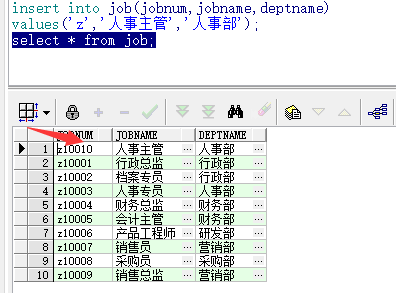
:new.jobnum:=('z'||:new.jobnum);

end;

*--向职务表中插入数据验证触发器*

insert into job(jobnum,jobname,deptname)

values('z','人事主管','人事部');



## 5.7 数据库服务器性能优化

服务器的性能优化是通过调节的目的是通过将网络流通、磁盘 I/O 和 CPU 时间减到最小，使每个查询的响应时间最短并最大限度地提高整个数据库服务器的吞吐量。为达到此目的，需要了解应用程序的需求和数据的逻辑和物理结构，并在相互冲突的数据库使用之间（如联机事务处理 (OLTP) 与决策支持）权衡。

对性能问题的考虑应贯穿于开发阶段的全过程，不应只在最后实现系统时才考虑性能问题。许多使性能得到显著提高的性能事宜可通过开始时仔细设计得以实现。为最有效地优化数据库的性能，必须在极为多样化的情形中识别出会使性能提升最多的区域，并对这些区域集中分析。

虽然其它系统级性能问题（如内存、硬件等）也是研究对象，但经验表明从这些方面获得的性能收益通常会增长，因为机器的配置根据需要很容易实现更新甚至更换。

总的来说，数据库服务器在安装过程中做到基本优化的配置，通常不需要作太大的调整，根据实际运行(维护)阶段的实际运行效果再作必要的调整。

## 5.8 备份

* 检查主库是否开归档

SELECT log\_mode FROM v$database;

* 检查主库的DB\_NAME,DB\_UNIQUE\_NAME.  
  主库和备库前者一致，后者不能一致；

SQL> show parameter db\_name

SQL> show parameter db\_unique\_name

* 查看数据文件位置  
  DB\_FILE\_NAME\_CONVERT该参数需要使用

SQL> select name from v$datafile;

* 查看日志文件位置  
  LOG\_FILE\_NAME\_CONVERT该参数需要使用

SQL> select \* from v$logfile;

* 设置dg远程归档路径

ALTER SYSTEM SET LOG\_ARCHIVE\_DEST\_2='SERVICE=wqdbdg NOAFFIRM ASYNC VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=wqdbdg';

ALTER SYSTEM SET LOG\_ARCHIVE\_DEST\_STATE\_2=ENABLE;

ALTER SYSTEM SET LOG\_ARCHIVE\_FORMAT='%t\_%s\_%r.arc' SCOPE=SPFILE;

ALTER SYSTEM SET LOG\_ARCHIVE\_MAX\_PROCESSES=10;

ALTER SYSTEM SET REMOTE\_LOGIN\_PASSWORDFILE=EXCLUSIVE SCOPE=SPFILE;

* 配置以下参数

ALTER SYSTEM SET FAL\_SERVER=wqdbdg;

alter system set fal\_client=wqdb;

ALTER SYSTEM SET DB\_FILE\_NAME\_CONVERT='/opt/oracle/oradata/wqdb/','/opt/oracle/oradata/wqdbdg/' SCOPE=SPFILE;

ALTER SYSTEM SET LOG\_FILE\_NAME\_CONVERT='/opt/oracle/oradata/wqdb/','/opt/oracle/oradata/wqdbdg/' SCOPE=SPFILE;

ALTER SYSTEM SET STANDBY\_FILE\_MANAGEMENT=AUTO;

* 重启数据库，生效修改的参数

commit;

shutdown immediate

startup

* 创建standby redo logs

ALTER DATABASE ADD STANDBY LOGFILE ('/opt/oracle/oradata/wqdb/standby\_redo04.log') SIZE 50M;

ALTER DATABASE ADD STANDBY LOGFILE ('/opt/oracle/oradata/wqdb/standby\_redo05.log') SIZE 50M;

ALTER DATABASE ADD STANDBY LOGFILE ('/opt/oracle/oradata/wqdb/standby\_redo06.log') SIZE 50M;

ALTER DATABASE ADD STANDBY LOGFILE ('/opt/oracle/oradata/wqdb/standby\_redo07.log') SIZE 50M;

commit;

* 重启数据库，生效修改的参数

alter system set local\_listener='';

commit;

shutdown immediate

startup

* 服务配置 主备系统均需配置

cat /opt/oracle/products/network/admin/tnsnames.ora

WQDB =

(DESCRIPTION =

(ADDRESS\_LIST =

(ADDRESS = (PROTOCOL = TCP)(HOST = 10.10.160.117)(PORT = 1521))

)

(CONNECT\_DATA =

(SERVICE\_NAME = wqdb)

)

)

WQDBDG =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 10.10.160.118)(PORT = 1521))

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = wqdbdg)

)

)

* 重启数据库，生效修改的参数

commit;

shutdown immediate

startup

复制文件,修改的目录是为了与传过来的pfile目录一致，后面拉起备库使用

$ mkdir -p /opt/oracle/oradata/wqdb //control file

$ mkdir -p /opt/oracle/admin/wqdb/adump //adump目录

$ mkdir -p /opt/oracle/oradata/wqdb //datafile

$ mkdir -p /opt/oracle/fast\_recovery\_area/wqdb

$ # Standby controlfile to all locations.

$ scp oracle@10.10.160.117:/archive/archive1/wqdbdg.ctl /opt/oracle/oradata/wqdb/control01.ctl

$ cp /opt/oracle/oradata/wqdb/control01.ctl /opt/oracle/fast\_recovery\_area/wqdb/control02.ctl

$ # Parameter file.

$ scp oracle@10.10.160.117:/archive/archive1/initwqdb.ora /data/data1/initwqdb.ora //pfile

$ # Remote login password file.

$ scp oracle@10.10.160.117:$ORACLE\_HOME/dbs/orapwwqdb $ORACLE\_HOME/dbs/

起监听

SID\_LIST\_LISTENER =

(SID\_LIST =

(SID\_DESC =

(GLOBAL\_DBNAME = wqdbdg)

(ORACLE\_HOME = /opt/oracle/products)

(SID\_NAME = wqdbdg)

)

)

LISTENER =

(DESCRIPTION\_LIST =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 10.10.160.118)(PORT = 1521))

)

(DESCRIPTION =

(ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))

)

)

ADR\_BASE\_LISTENER = /opt/oracle

$ lsnrctl start

$ lsnrctl reload

$ lsnrctl status

使用RAMN复制来创建备库

export ORACLE\_SID=wqdbdg

$ sqlplus / as sysdba

SQL> STARTUP NOMOUNT PFILE='/data/data1/initwqdbdg.ora';

SQL> exit

rman TARGET sys/oracle@wqdb AUXILIARY sys/oracle@wqdbdg

DUPLICATE TARGET DATABASE

FOR STANDBY

FROM ACTIVE DATABASE

DORECOVER

SPFILE

SET db\_unique\_name='WQDBDG' COMMENT 'Is standby'

SET LOG\_ARCHIVE\_DEST\_2='SERVICE=wqdb ASYNC VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=wqdb'

SET FAL\_SERVER='WQDB' COMMENT 'Is primary'

NOFILENAMECHECK;

开始应用standby redo log

在备库上操作，主要使用第二个命令

# Foreground redo apply. Session never returns until cancel.

SQL>ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;

# Background redo apply. Control is returned to the session once the apply process is started.

SQL>ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;

测试日志传输

在主库，检查最新的归档日志和强制归档

ALTER SESSION SET nls\_date\_format='DD-MON-YYYY HH24:MI:SS';

SELECT sequence#, first\_time, next\_time FROM v$archived\_log ORDER BY sequence#;

ALTER SYSTEM SWITCH LOGFILE;

在备库检查已经被应用的最新归档日志

ALTER SESSION SET nls\_date\_format='DD-MON-YYYY HH24:MI:SS';

SELECT sequence#, first\_time, next\_time, applied FROM v$archived\_log ORDER BY sequence#;