东北师范大学本科生课程作业

（2019 春季学期）

课程名称：　数据库系统概论

作业题目：　数据库设计实验

任课老师：　罗娜

小组成员：　孔一言、吴婷婷、徐文晴、梁莉莉、屈英杰

专　　业：　软件工程

年　　级：　2017 级

学　　院：　信息科学与技术学院

年 月 日：　2019 年 6 月 26 日

小组成员：

姓名	学号
梁莉莉	2017011581
孔一言	2017012842
徐文晴	2017013321
屈英杰	2017012454
吴婷婷	2017012259

小组角色分工：

序号	身份	对应成员
1	项目管理员	梁莉莉
2	DBA	梁莉莉
3	系统分析员	吴婷婷
4	系统设计员	孔一言、徐文晴、屈英杰
5	系统开发员	梁莉莉、孔一言、屈英杰、徐文晴、吴婷婷
6	系统测试员	孔一言

目录

1.1 系统分析

本系统是为小型软件企业设计的人员管理系统，其涉及部门有：技术部、产品部和新媒体运营部，其中技术部又包括前端技术开发部、后端技术开发部。

根据以往开发类似系统的经验来看，在本系统之中，用户想要从数据库中获得的信息主要有职工信息，部门信息，职位-工资信息以及打卡信息。其中，职位-工资信息中的职位又分为部门负责人和部门普通职员，对于所属部门不同、职位不同的员工的基本工资也有所不同。打卡信息是作为员工除基本工资以外额外奖金的唯一来源，每次打卡计 5 元，年底统一结算。

根据数据分析和数据收集获得的主要成果为数据字典。数据字典包含数据项、数据结构、数据流、数据存储和处理过程及部分。

1.1.1 数据字典

TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	COLUMN_TYPE	COLUMN_COMMENT
db_people_management	t_staff	staff_id	int(50)	职工编号 id
db_people_management	t_staff	department_id	int(50)	所属部门编号 id
db_people_management	t_staff	staff_name	varchar(255)	姓名
db_people_management	t_staff	staff_age	int(50)	年龄
db_people_management	t_staff	position_id	int(50)	职位编号 id
db_people_management	t_staff	total_salary	int(50)	职工总工资
db_people_management	t_department	department_id	int(50)	部门编号 id
db_people_management	t_department	department_name	varchar(255)	部门名称
db_people_management	t_department	department_manager	varchar(255)	部门负责人姓名
db_people_management	t_department	department_num	int(50)	部门人数
db_people_management	t_position_salary	position_id	int(50)	职位编号 id
db_people_management	t_position_salary	position_name	varchar(255)	职位名称
db_people_management	t_position_salary	department_id	int(50)	所属部门编号 id
db_people_management	t_position_salary	salary	int(50)	职位对应的资本工资
db_people_management	t_sign_in	sign_id	int(50)	打卡编号 id
db_people_management	t_sign_in	staff_id	int(50)	打卡职工编号 id
db_people_management	t_sign_in	sign_time	date	打卡时间

1.2 系统设计

1.2.1 概念结构设计

将我们在需求分析阶段得到的应用需求抽象为信息世界的结构，通过设计数据库的概念结构，形成独立于具体 DBMS 的概念模型。

现要建立关于职员、部门、职位、工资、考勤等信息的一个关系数据库。一个职员属于一个部门，一个部门有若干职员；一个职员对应一个职位，一个职位可属于若干职员；职员和工资是一一对应的关系；一个职员对应多条考勤记录，一条考勤记录对应一个职员。

职员：职工编号，所属部门编号，姓名，年龄，职位编号，总工资；

部门：部门编号，部门名称，部门负责人姓名，部门人数；

职位：职位编号，职位名称，职位对应的基本工资；

工资：基本工资，总工资；

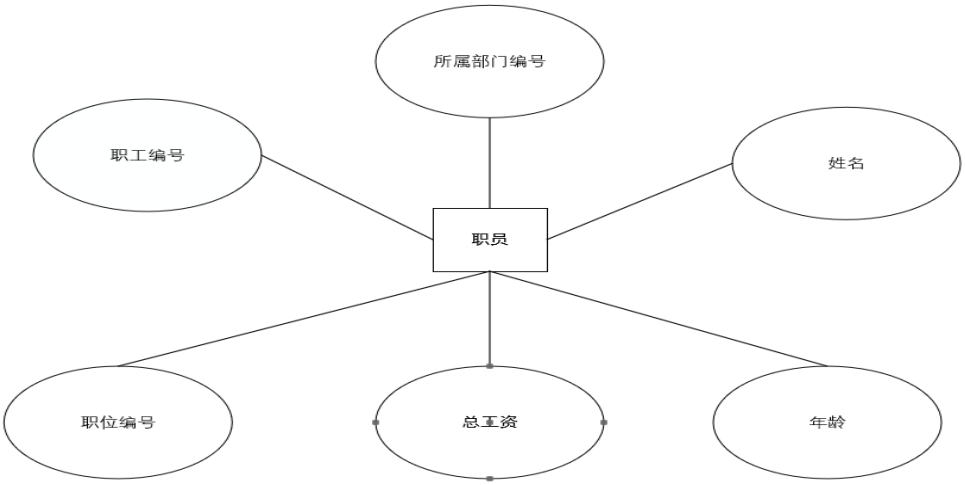
考勤：打卡编号，打卡职工编号，打卡时间

1. 实体及属性 E-R 图描述

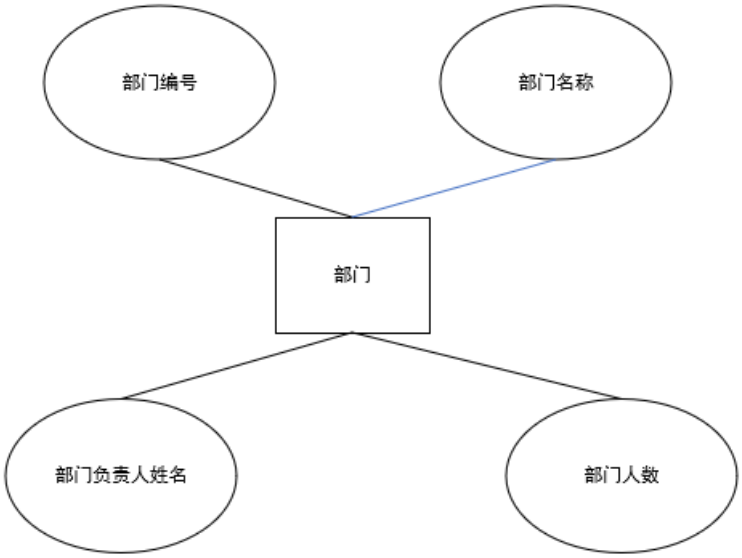
E-R 图提供了表示实体型、属性和联系的方法。

设计 E-R 图时，能作为属性的就不作为实体，这样有利于 E-R 图的简化。

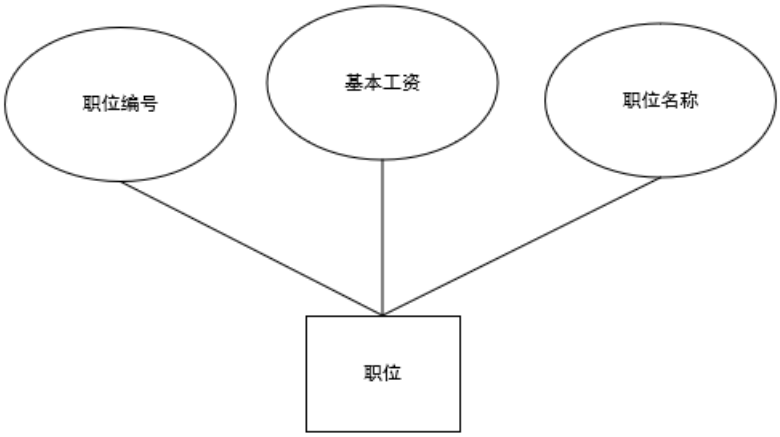
职员实体 E-R 图：



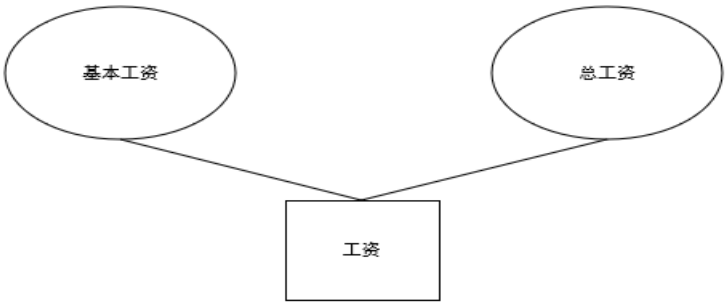
部门实体 E-R 图：



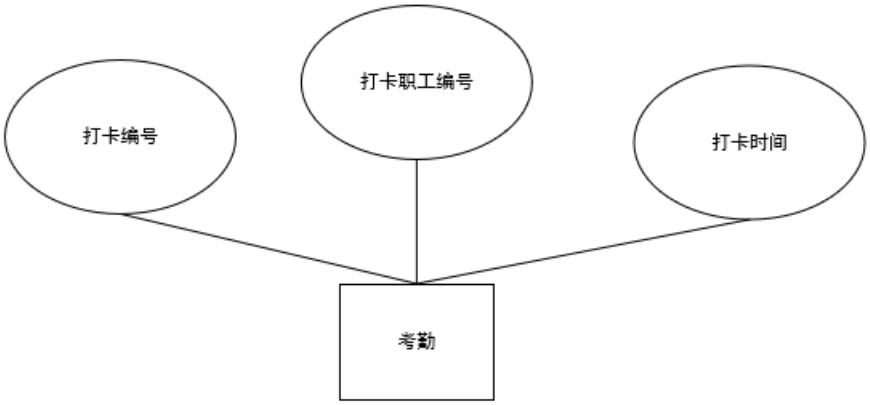
职位实体 E-R 图：



工资实体 E-R 图：



考勤实体 E-R 图：

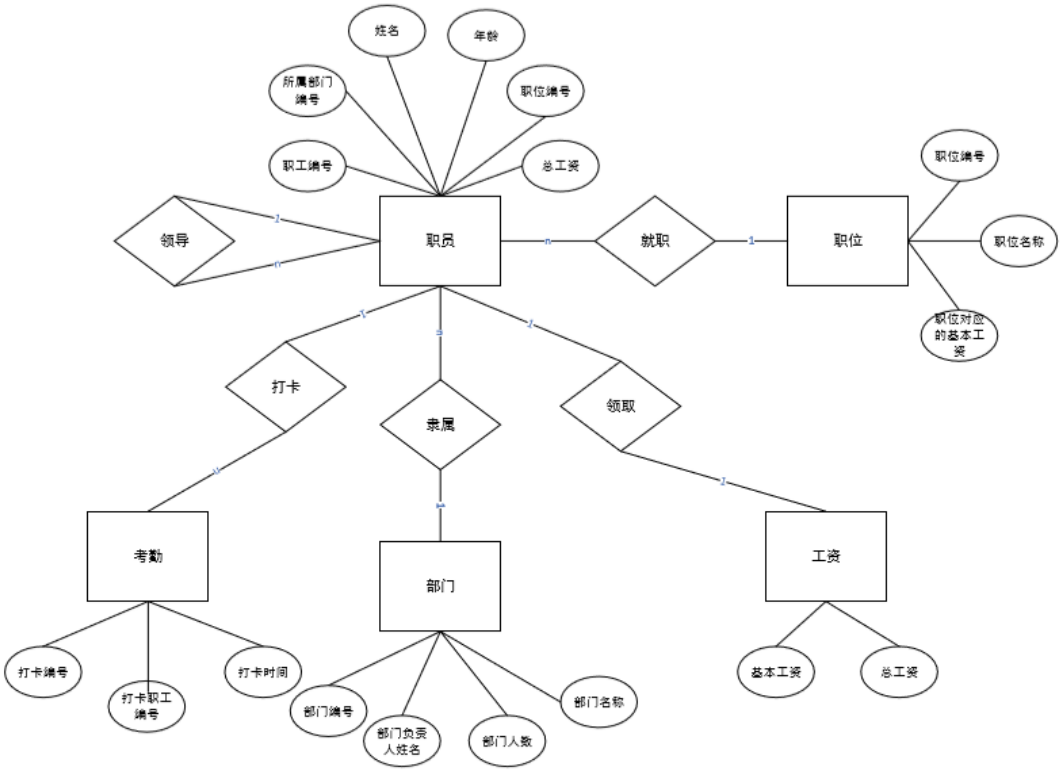


2．设计子 E-R 图

确定实体与属性的两条准则：

（1）属性是不可再分的数据项，属性不可以再有属性；

（2）属性不能与其他实体发生关系，联系只能存在于实体与实体之间。



3．视图集成

1) 视图集成的作用

一方面可以完整地描述企业的信息及其联系，另一方面在集成的过程中可以解决冲突和消除冗余。

2) 视图集成中 E-R 图之间的三类冲突：

a) 属性冲突：同一属性可能会存在于不同的 E-R 图，由于设计人员不同或是出发点不同，对属性的类型、取值范围、数据单位等可能会不一致，这些属性对应的数据将来只能以一种形式在计算机中存储，这就需要在设计阶段进行统一。

b) 命名冲突：相同意义的属性，在不同的 E-R 图上有着不同的命名（异名同义），或是名称相同的属性在不同的 E-R 图中代表着不同的意义（同名异义），这些都需要进行统一。

c) 结构冲突：同一实体在不同的 E-R 图中有不同的属性，同一对象在某一 E-R 图中被抽象为实体而在另一个 E-R 图中又被抽象为属性，需要统一。

3）视图集成消除冗余注意点：

a) 在 E-R 图的综合过程中，同名实体只能出现一次，还要去掉不必要的联系，而且不能出现环路，这样才能消除冗余数据的冗余联系。

b) 一般来说，从总体 E-R 图必须能导出所有局部 E-R 视图，包括所有的实体、属性和联系。

在本次实验中实体较少，最后完整的 E-R 图并不存在不必要的冗余和冲突。

1.2.2 逻辑结构设计

1. E-R 图向关系模式的转换

一个实体型转换为一个关系模式

关系的属性就是实体的属性；

关系的码就是实体的码；

实体间的联系转换：

A．1:1 联系可以转换成一个独立的关系模式，也可以与任意一端对应的关系模式合并。

转换为一个独立的关系模式

关系的属性：与该联系相连的各实体的标识符以及联系本身的属性

关系的候选码：每个实体标识符均是该关系的候选码

与某一端对应的关系模式合并

合并后关系的属性：加入另一关系的码和联系本身的属性

合并后关系的候选码：不变

B．1:n 联系可以转换成一个独立的关系模式，也可以与 n 端对应的关系模式合并。

转换为一个独立的关系模式

关系的属性：与该联系相连的各实体的标识符以及联系本身的属性

关系的候选码：n 端实体的标识符

与 n 端对应的关系模式合并

合并后关系的属性：在 n 端关系中加入 1 端关系的码和联系本身的属性

合并后关系的候选码：不变

C．m:n 联系转换成一个关系模式

关系的属性：与该联系相连的各实体的标识符以及联系本身的属性

关系的候选码：各实体标识符的组合

D．三个或三个以上实体间的一个多元联系转换为一个关系模式

关系的属性：与该联系相连的各实体的标识符以及联系本身的属性

关系的候选码：各实体标识符的组合

转换结果如下：

1) t_staff（staff_id，department_id，staff_name，staff_age，position_id, total_salary)
2) t_department(department_id,department_name,department_manager,department_num)
3) t_position_salary(position_id，position_name，department_id，salary)
4) t_sign_in（sign_id，staff_id，sign_time）

2. 关系模式优化

本阶段将上阶段由 E-R 图转化的模式进行优化。经过上一过程的转化，关系模式不一定是最“好”，所以要经过这一步的优化。

（1）关系模式规范化

目的：解决更新异常和数据冗余，对不符合规范的关系模式分解成 3NF 或者 BCNF。

1NF 定义：

如果一个关系模式 R 的所有属性都是不可分的基本数据项，则称关系 R 为第一范式的关系模式，简称关系 R 属于第一范式。

经检验，以上 4 个关系模式，均满足第一范式要求。每一个属性都是不可再分的。

2NF 定义：

若关系模式 R 属于第一范式，并且每一个非主属性都完全函数依赖于 R 码，此时关系模式为第二范式。

经检验，以上 4 个关系模式，均满足第二范式要求。没有非主属性对码的传递部分函数依赖。

3NF 定义：

关系模式 R<U,F>中若不存在这样的码 X、属性组 Y 及非主属性 Z(Z⊆Y)，使得 X→Y，Y→X,Y→Z 成立，则称 R⊆3NF。

经检验，以上 4 个关系模式，均满足第三范式要求。没有主属性对码的部分依赖及传递依赖。

具体如下：

5) t_staff(staff_id，department_id，staff_name，staff_age，position_id, total_salary)
6) t_department(department_id,department_name,department_manager,department_num)
7) t_ positionsalary(position_id，position_name，department_id，salary)
8) t_ sign_in（sign_id,staff_id,sign_time）

3. 对关系模式进行合并

对具有关系的模式进行合并，通常这类关系模式会经常被查询而频繁地进行连接运算而降低查询的效率，合并后的关系模式可能会带来冗余，但这样做事值得的。

经检验，不需要合并。

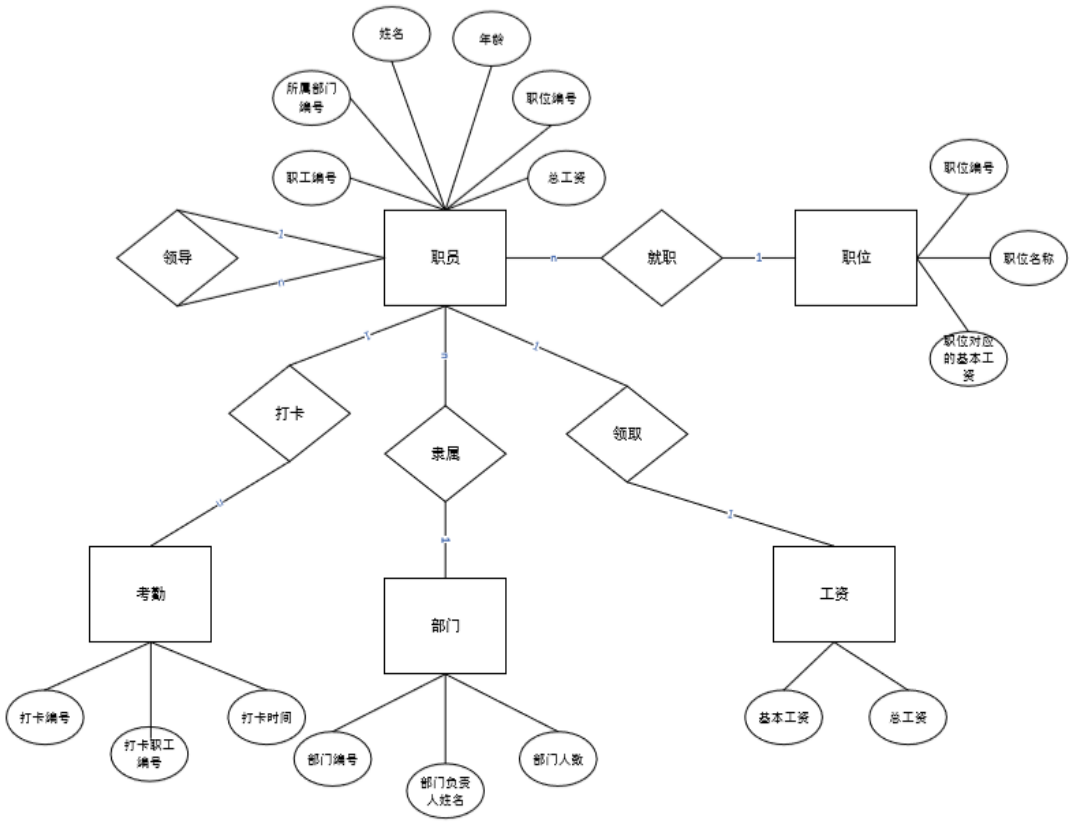
4. 对关系模式进行分解

水平分解：把基本关系的元祖分解为若干子集，定义每个子集合为一个子关系，以提高系统效率。

垂直分解：把关系模式 R 的属性分解为若干子集合，形成若干关系模式。

经检验，不需要分解。

优化后的 E-R 图如下：



1.2.3 物理结构设计

数据库管理系统：MySQL

存取方法：B+树索引，由于每次查询的结果所占比例较高，所以选择建立索引，以减少查询的代价。

存储路径：存储对象和日志文件都放在数据库管理系统的相同盘下。

系统配置：同时操作数据库的最大用户数：100

同时打开的数据对象最大数：10

数据库最大容量：10K

维护代价：日常维护：需要定期转储数据库，以维护数据库安全性。

特殊维护：当应用需求发生变化时，需要对数据库进行必要的修改。

经检验数据库符合用户要求。

1.3 数据库创建和数据装载

1.3.1 创建数据库

确定了数据库的逻辑结构与物理结构后，就可以用所用的 DBMS 提供的数据定义语言来严格描述数据库结构

1) 创建 t_staff 表（staff_id, department_id, staff_name, staff_age, position_id, total_salary）
SQL 代码:

```
CREATE TABLE `t_staff` (
    `staff_id` int(50) NOT NULL AUTO_INCREMENT COMMENT '职工编号 id',
    `department_id` int(50) DEFAULT NULL COMMENT '所属部门编号 id',
    `staff_name` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT '姓名',
    `staff_age` int(50) DEFAULT NULL COMMENT '年龄',
    `position_id` int(50) DEFAULT NULL COMMENT '职位编号 id',
    `total_salary` int(50) DEFAULT NULL COMMENT '职工总工资',
    PRIMARY KEY (`staff_id`) USING BTREE,
    INDEX `department_id`(`department_id`) USING BTREE,
    INDEX `position_id`(`position_id`) USING BTREE,
    CONSTRAINT `department_id` FOREIGN KEY (`department_id`) REFERENCES `t_department` (`department_id`) ON DELETE CASCADE ON UPDATE CASCADE,
```

CONSTRAINT `position_id` FOREIGN KEY (`position_id`) REFERENCES `t_position_salary` (`position_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB AUTO_INCREMENT = 1 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Compact;

2) 创建 t_department 表（department_id，department_name，department_manager，department_num）

SQL 代码：

```
CREATE TABLE `t_department` (
    `department_id` int(50) NOT NULL AUTO_INCREMENT COMMENT '部门编号 id',
    `department_name` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT '部门名称',
    `department_manager` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT '部门负责人姓名',
    `department_num` int(50) DEFAULT NULL COMMENT '部门人数',
    PRIMARY KEY (`department_id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 2 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Compact;
```

3) 创建 t_position_salary 表（position_id，position_name，department_id，salary）

SQL 代码：

```
CREATE TABLE `t_position_salary` (
    `position_id` int(50) NOT NULL AUTO_INCREMENT COMMENT '职位编号 id',
    `position_name` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT '职位名称',
    `department_id` int(50) DEFAULT NULL COMMENT '所属部门编号 id',
    `salary` int(50) DEFAULT NULL COMMENT '职位对应的资本工资',
    PRIMARY KEY (`position_id`) USING BTREE,
    INDEX `department_id2`(`department_id`) USING BTREE,
    CONSTRAINT `department_id2` FOREIGN KEY (`department_id`) REFERENCES `t_department` (`department_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB AUTO_INCREMENT = 1 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Compact;
```

4) 创建 t_sign_in 表（sign_id，staff_id，sign_time）

SQL 代码：

```
CREATE TABLE `t_sign_in` (
    `sign_id` int(50) NOT NULL AUTO_INCREMENT COMMENT '打卡编号 id',
    `staff_id` int(50) DEFAULT NULL COMMENT '打卡职工编号 id',
    `sign_time` date DEFAULT NULL COMMENT '打卡时间',
    PRIMARY KEY (`sign_id`) USING BTREE,
    INDEX `staff_id`(`staff_id`) USING BTREE,
    CONSTRAINT `staff_id` FOREIGN KEY (`staff_id`) REFERENCES `t_staff` (`staff_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB AUTO_INCREMENT = 1 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Compact;
```

1.3.2数据装载

1. t_staff：

```
INSERT INTO `t_staff` VALUES (101, 1, '杨兴邦', 38, 15, 16900);
INSERT INTO `t_staff` VALUES (102, 1, '王兆国', 29, 15, 15600);
INSERT INTO `t_staff` VALUES (103, 1, '刘少华', 30, 15, 17000);
INSERT INTO `t_staff` VALUES (104, 1, '徐敏洪', 45, 11, 23000);
INSERT INTO `t_staff` VALUES (105, 2, '孙霞', 25, 16, 13890);
INSERT INTO `t_staff` VALUES (106, 2, '金溪', 28, 12, 20800);
INSERT INTO `t_staff` VALUES (107, 2, '李万', 36, 16, 15000);
INSERT INTO `t_staff` VALUES (108, 2, '余能', 30, 16, 14800);
INSERT INTO `t_staff` VALUES (109, 3, '王爱莲', 23, 17, 8560);
INSERT INTO `t_staff` VALUES (110, 3, '杨孟贤', 26, 13, 18900);
INSERT INTO `t_staff` VALUES (111, 3, '沈洁', 24, 17, 14000);
INSERT INTO `t_staff` VALUES (112, 3, '梁红', 30, 17, 12100);
INSERT INTO `t_staff` VALUES (113, 4, '孔凡河', 22, 18, 11810);
INSERT INTO `t_staff` VALUES (114, 4, '甘燕', 23, 18, 10470);
INSERT INTO `t_staff` VALUES (115, 4, '屈海婉', 22, 18, 9600);
INSERT INTO `t_staff` VALUES (116, 4, '肖晓潇', 25, 14, 16040);
```

2. t_department：

```
INSERT INTO `t_department` VALUES (1, '技术部后端', '徐敏洪', 4);
INSERT INTO `t_department` VALUES (2, '技术部前端', '金溪', 4);
INSERT INTO `t_department` VALUES (3, '产品部', '杨孟贤', 4);
INSERT INTO `t_department` VALUES (4, '新媒体运营部', '肖晓潇', 4);
```

3. t_position_salary：

```
INSERT INTO `t_position_salary` VALUES (11, '负责人', 1, 21000);
INSERT INTO `t_position_salary` VALUES (12, '负责人', 2, 18000);
INSERT INTO `t_position_salary` VALUES (13, '负责人', 3, 15000);
INSERT INTO `t_position_salary` VALUES (14, '负责人', 4, 14000);
INSERT INTO `t_position_salary` VALUES (15, '普通职工', 1, 15000);
INSERT INTO `t_position_salary` VALUES (16, '普通职工', 2, 13000);
INSERT INTO `t_position_salary` VALUES (17, '普通职工', 3, 10000);
INSERT INTO `t_position_salary` VALUES (18, '普通职工', 4, 8000);
INSERT INTO `t_position_salary` VALUES (8, '普通职工', 4, 8000);
```

4. t_sign_in：

```
INSERT INTO `t_sign_in` VALUES (21, 104, '2018-05-30');
INSERT INTO `t_sign_in` VALUES (22, 101, '2018-05-30');
INSERT INTO `t_sign_in` VALUES (23, 103, '2018-05-30');
INSERT INTO `t_sign_in` VALUES (24, 110, '2018-05-30');
INSERT INTO `t_sign_in` VALUES (25, 116, '2018-05-30');
INSERT INTO `t_sign_in` VALUES (26, 104, '2019-06-01');
INSERT INTO `t_sign_in` VALUES (27, 111, '2018-05-30');
INSERT INTO `t_sign_in` VALUES (28, 102, '2018-05-30');
```

```
INSERT INTO `t_sign_in` VALUES (29, 106, '2018-05-30');
```

1.4 数据库应用软件的功能设计和开发

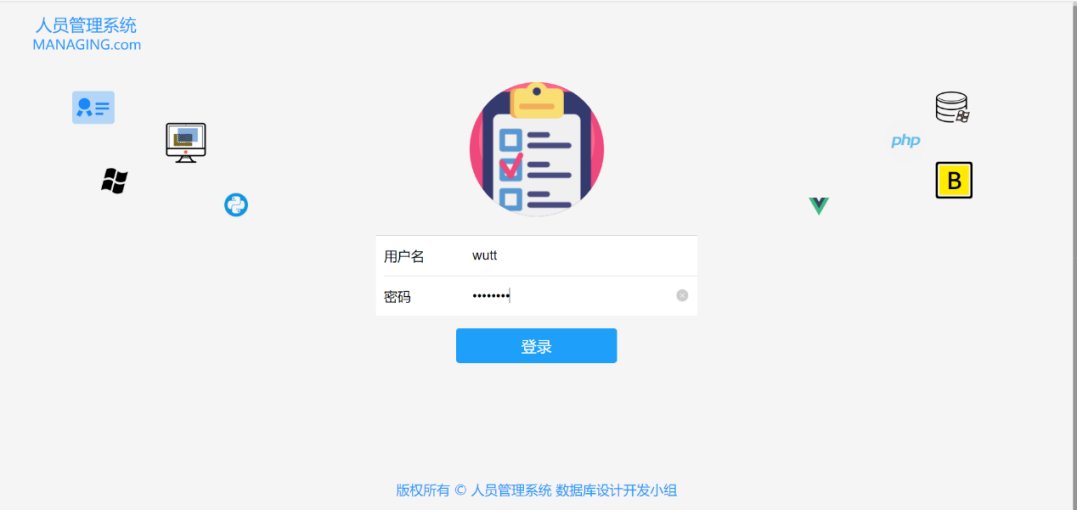
1.4.1 数据库应用软件的功能

本次的数据库设计与应用开发大作业，我们小组设计的是小型软件企业的人员管理系统中的数据库。

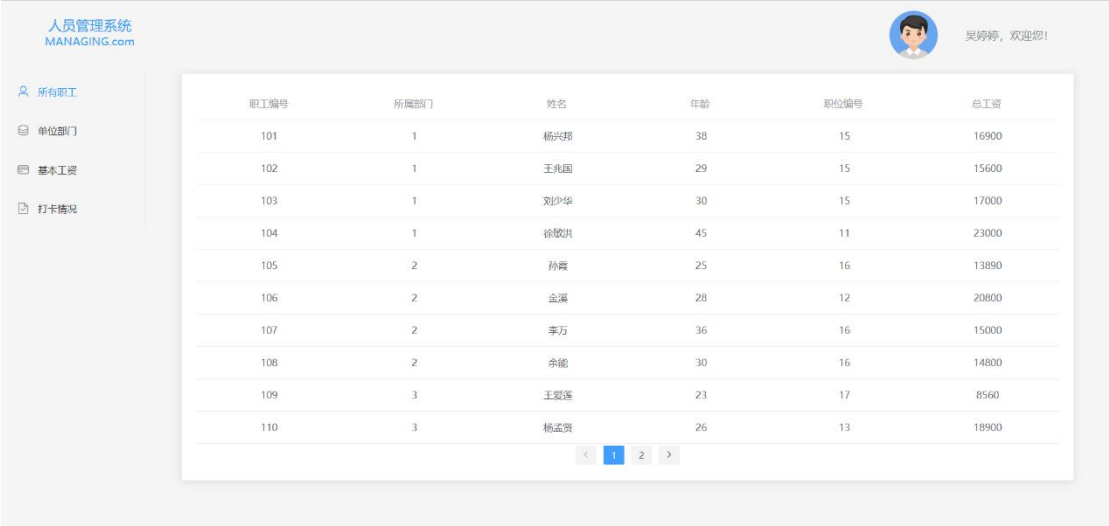
1.数据库应用软件的功能

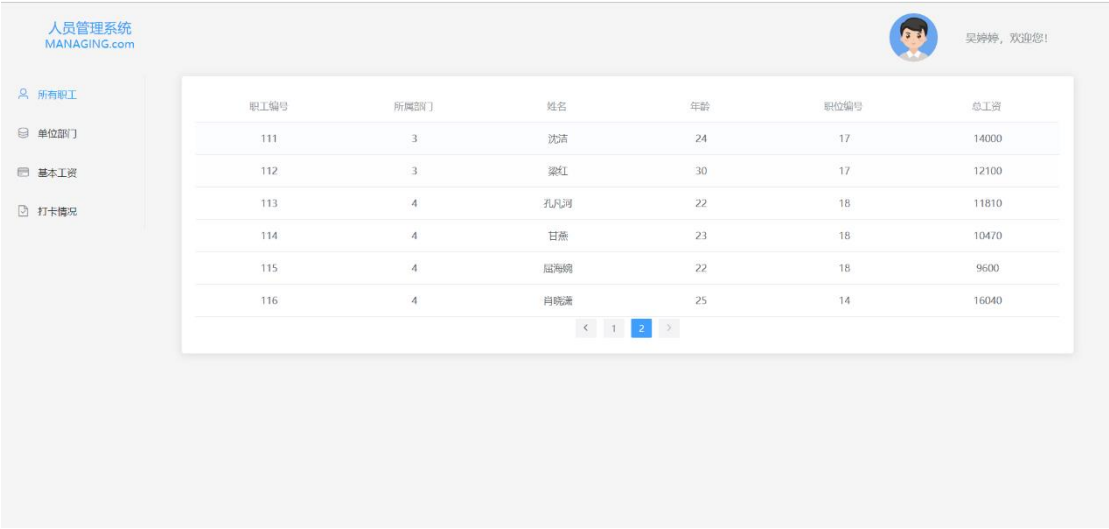
（1）登录

登录用户名为系统开发人员分配的账号（选用账号 wutt 进行系统的登录演示），初始密码为 123456。

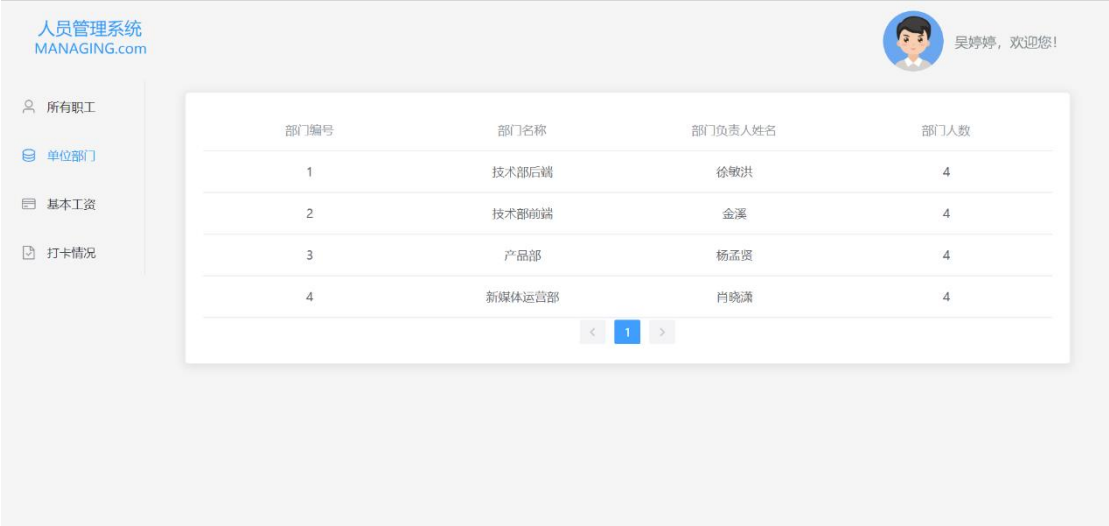


（2）查看企业中所有的职员

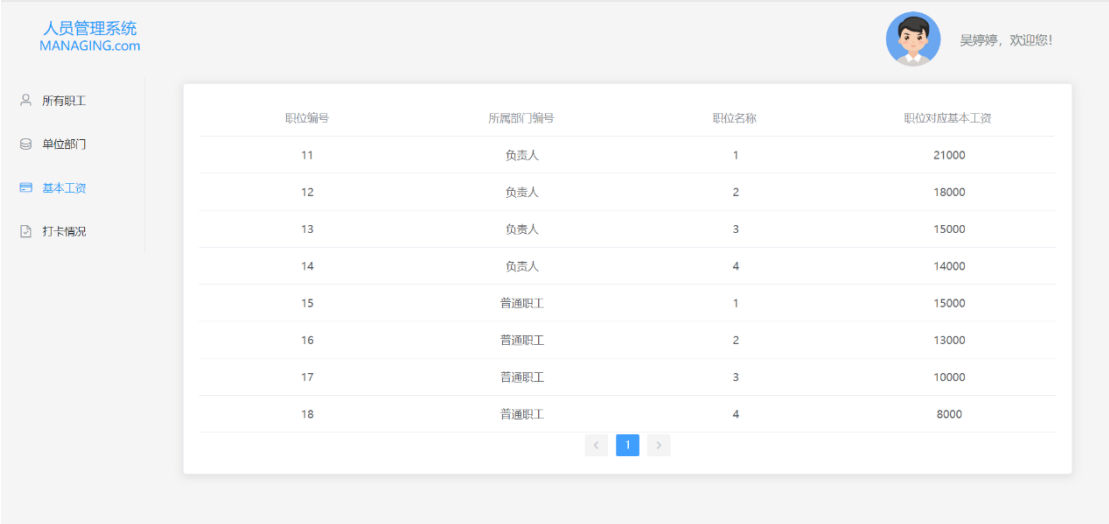




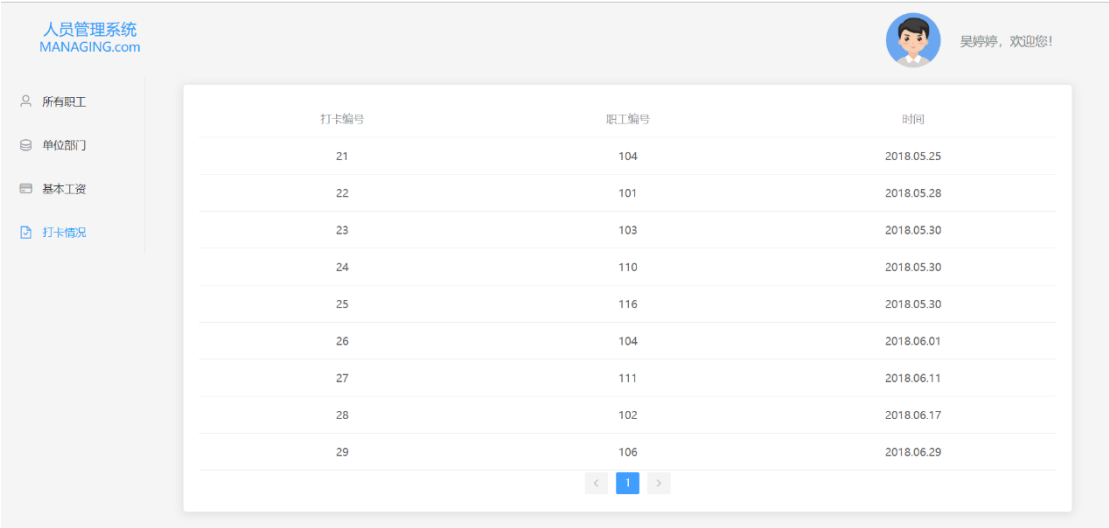
（3）查看企业所设有的所有部门



（4）查看职位对应的基本工资



（5）查看打卡情况



2. 应用软件的开发

整个项目所使用的技术平台、框架支持如下：

数据库管理系统：MySQL

后端：Springboot

前端：Vue.js、Element-ui

项目开发任务分工:

分工	主负责人	参与成员
前端设计及开发	吴婷婷	徐文晴
后端开发	梁莉莉	孔一言、屈英杰
文档的编写汇总	屈英杰	吴婷婷、梁莉莉、孔一言、徐文晴
测试	孔一言	吴婷婷、徐文晴、孔一言、屈英杰

本项目开发时间共计 2 周。

1.4.2 功能对应的 SQL 语句

1. 实现对应功能的 SQL 语句

（1）列出所有职工以及其相应信息

```
<select id="listAllStaff " resultMap="Staff">
        SELECT *
        FROM `t_staff`
</select>
```

（2）列出所有部门

```
<select id="listAllDepartment " resultMap="Department">
        SELECT *
        FROM `t_separtment`
</select>
```

（3）列出所有职位以及其对应的基本工资

```
<select id="listAllPositionSalary " resultMap="PositionSalary">
        SELECT *
        FROM `t_position_salary`
</select>
```

（4）列出所有打卡签到情况

```
<select id="listAllSignIn " resultMap="SignIn">
        SELECT *
        FROM `t_sign_in`
</select>
```

1.5 数据库应用系统测试

```
//测试往表内添加记录
@Test
    public void testAddStaff() throws Exception{
        Staff staff = new Staff();
        staff.setDepartment_id(2);
        staff.setStaff_name("李明");
        staff.setPosition_id(2);
        staff.setTotal_salary(1000);
        staffService.addStaff(staff);
}

//测试删除表中一条记录
@Test
    public void testdelStaff() throws Exception{
        staffService.delStaff(3);
}

//测试修改表中记录
@Test
    public void testUpdateStaff() throws Exception{
        Staff staff = new Staff();
        staff.setDepartment_id(2);
        staff.setStaff_name("李明");
        staff.setPosition_id(2);
        staff.setTotal_salary(1000);
        staffService.updateStaff(staff);
}

//测试模糊查询用户的名字和所属部门
@Test
    public void testQueryStaff() throws Exception{
        String keyword = “明”;
        staffService.queryStaff(keyword);
}

//测试列出表内所有记录
@Test
    public void testlistAllStaff() throws Exception{
       System.out.println(staffService.listAllStaff());
}
```