用户及权限系统数据库设计

1 设计思路

为了设计一套具有较强可扩展性的用户认证管理，需要建立用户、角色和权限等数据库表，并且建立之间的关系，具体实现如下。

1.1 用户

用户仅仅是纯粹的用户，用来记录用户相关信息，如用户名、密码等，权限是被分离出去了的。用户（User）要拥有对某种资源的权限，必须通过角色（Role）去关联。

用户通常具有以下属性：

编号，在系统中唯一。

名称。

用户密码。

注释，描述用户或角色的信息，例如游客或者用户。

1.2 角色

角色是使用权限的基本单位，拥有一定数量的权限，通过角色赋予用户权限，通常具有以下属性：

编号，在系统中唯一。

名称，在系统中唯一。

注释，描述角色信息

1.3 权限

权限指用户根据角色获得对程序某些功能的操作，例如对文件的读、写、修改和删除功能，通常具有以下属性：

编号，在系统中唯一。

名称，在系统中唯一。

注释，描述权限信息

1.4 用户与角色的关系

一个用户（User）可以隶属于多个角色（Role），例如是租人的用户，也可以是出租的用户，一个角色组也可拥有多个用户，用户角色就是用来描述他们之间隶属关系的对象。用户（User）通过角色（Role）关联所拥有对某种资源的权限，例如

l 用户（User）：

UserID	UserName	UserPwd
1	张三	xxxxxx
2	李四	xxxxxx
……		

l 角色（Role）：

RoleID	RoleName	RoleNote
01	游客	未注册的用户
02	出租用户	可以发布出租项和被租的用户
03	租人用户	可以发布租人项和租人的用户
04	一般工作人员	工作人员
……		

l 用户角色（User_Role）：

UserRoleID	UserID	RoleID	UserRoleNote
1	1	01	用户“张三”被分配到角色“游客”
2	2	02	用户“李四”被分配到角色“租人用户”
3	2	03	用户“李四”被分配到角色“出租用户”
……			

从该关系表可以看出，用户所拥有的特定资源可以通过用户角色来关联。

1.5 权限与角色的关系

一个角色（Role）可以拥有多个权限（Permission），同样一个权限可分配给多个角色。

例如：

l 角色（Role）：

RoleID	RoleName	RoleNote
01	游客	未注册的用户
02	出租用户	可以发布出租项和被租的用户
03	租人用户	可以发布租人项和租人的用户
04	一般工作人员	工作人员
……		

l 权限（Permission）：

PermissionID	PermissionName	PermissionNote
0001	发布出租项	允许发布出租项的对象
0002	发布租人项	允许发布租人项的对象

0003 付款/下单 允许付款和下单的对象

0004 浏览列表 允许浏览出租和租人的列表

……

l 角色权限（Role_Permission）：

RolePermissionID RoleID PermissionID RolePermissionNote

1 02 0001 角色“出租用户”具有权限“发布出租项”

2 03 0003 角色“租人”具有权限“下单/付款”

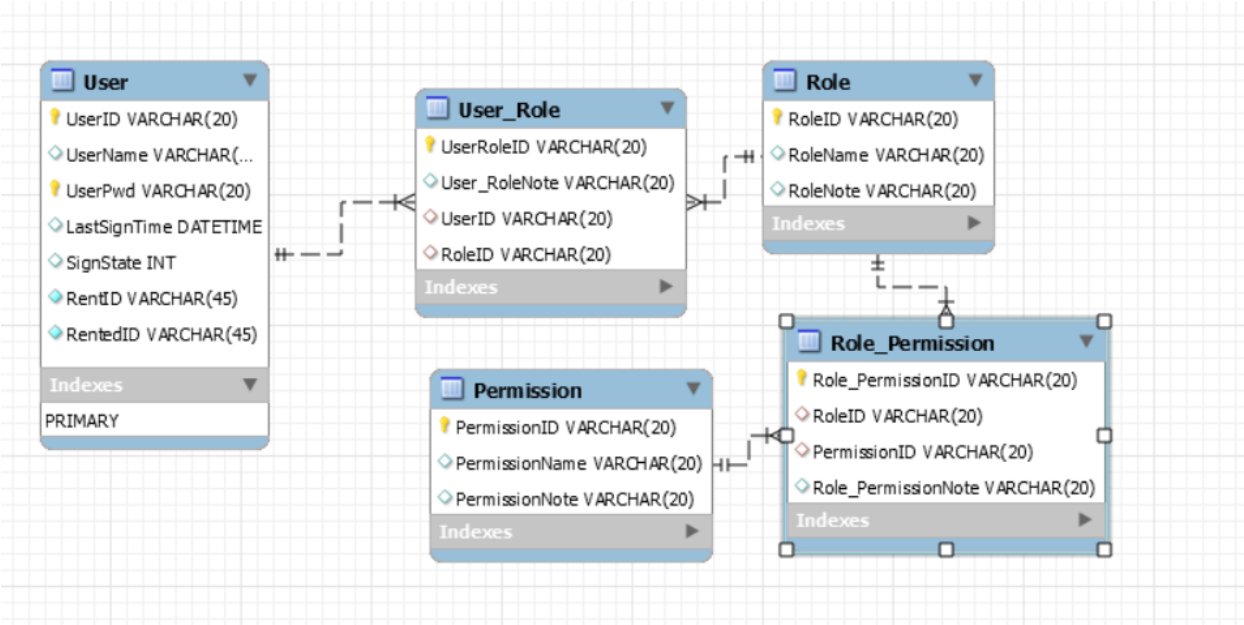
3 01 0004 角色“游客”具有权限“浏览列表”

4 03 0002 角色“租人用户”具有权限“发布租人项”

……

2 用户及权限系统数据库设计

2.1 数据库表



2.2 数据库表说明

2.2.1 用户表（Static_User）

Static_User

Static_User **字段名**	**详细解释**	**类型**	**备注**
UserID	用户编号	varchar(20)	PK
UserName	用户名称	varchar(20)	
UserPwd	用户密码	varchar(20)	
LastSignTime	最后登陆时间	datatime	
SignState	用户登陆状态标记	int	
RentID	租人记录编号	varchar(128)	

RentedID	出租记录编号	varchar(128)	

2.2.2 角色表（Static_Role）

Static_Role

Static_User **字段名**	**详细解释**	**类型**	**备注**
RoleID	角色编号	varchar(20)	PK
RoleName	角色名称	varchar(20)	
RoleNote	角色信息描述	varchar(20)	

2.2.3 用户 - 角色表（Static_User_Role）

Static_User_Role

Static_User **字段名**	**详细解释**	**类型**	**备注**
UserRoleID	用户角色编号	varchar(20)	PK
UserID	用户编号	varchar(20)	FK
RoleID	角色编号	varchar(20)	FK
UserRoleNote	用户角色信息描述	varchar(20)	

2.2.4 权限表（Static_Permission）

Static_Permission

Static_User **字段名**	**详细解释**	**类型**	**备注**
PermissionID	编号	varchar(20)	PK
PermissionName	权限名称	varchar(20)	
PermissionNote	全息信息描述	varchar(20)	

2.2.5 角色 - 权限表（Static_Role_Permission）

Static_Role_Permission

Static_User **字段名**	**详细解释**	**类型**	**备注**
RolePermissionID	角色权限编号	varchar(20)	PK
RoleID	角色编号	varchar(20)	FK
PermissionID	权限编号	varchar(20)	FK
RolePermissionNote	角色权限信息描述	varchar(20)	

2.3 数据库表代码

User:

```
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `UserID` VARCHAR(20) NOT NULL,
  `UserName` VARCHAR(20) NOT NULL,
  `UserPwd` VARCHAR(20) NOT NULL,
  `LastSignTime` DATETIME NOT NULL,
  `SignState` INT NOT NULL,
  `RentID` VARCHAR(45) NOT NULL,
  `RentedID` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`UserID`, `UserPwd`))
ENGINE = InnoDB
```

Role:

```
CREATE TABLE IF NOT EXISTS `mydb`.`Role` (
  `RoleID` VARCHAR(20) NOT NULL,
  `RoleName` VARCHAR(20) NOT NULL,
  `RoleNote` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`RoleID`))
ENGINE = InnoDB
```

User_Role:

```
CREATE TABLE IF NOT EXISTS `mydb`.`User_Role` (
  `UserRoleID` VARCHAR(20) NOT NULL,
  `User_RoleNote` VARCHAR(20) NOT NULL,
  `UserID` VARCHAR(20) NOT NULL,
  `RoleID` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`UserRoleID`),
  INDEX `UserID_idx` (`UserID` ASC),
  INDEX `RoleID_idx` (`RoleID` ASC),
  CONSTRAINT `UserID`
    FOREIGN KEY (`UserID`)
    REFERENCES `mydb`.`User` (`UserID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `RoleID`
    FOREIGN KEY (`RoleID`)
    REFERENCES `mydb`.`Role` (`RoleID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
```

Permission:

```
CREATE TABLE IF NOT EXISTS `mydb`.`Permission` (
  `PermissionID` VARCHAR(20) NOT NULL,
  `PermissionName` VARCHAR(20) NOT NULL,
  `PermissionNote` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`PermissionID`))
ENGINE = InnoDB
```

Role_Permission:

```
CREATE TABLE IF NOT EXISTS `mydb`.`Role_Permission` (
  `Role_PermissionID` VARCHAR(20) NOT NULL,
  `RoleID` VARCHAR(20) NOT NULL,
  `PermissionID` VARCHAR(20) NOT NULL,
  `Role_PermissionNote` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`Role_PermissionID`),
  INDEX `RoleID_idx` (`RoleID` ASC),
  INDEX `PermissionID_idx` (`PermissionID` ASC),
  CONSTRAINT `RoleID`
    FOREIGN KEY (`RoleID`)
    REFERENCES `mydb`.`Role` (`RoleID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `PermissionID`
```

```
    FOREIGN KEY (`PermissionID`)
    REFERENCES `mydb`.`Permission` (`PermissionID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```