数据库结构设计

## 表结构设计

#### 调度管理员表governer：

管理员：

1. 最高权限；
2. 普通调度员

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 取值 | 字段 | 类型 |
|  |  | index | integer |
| 标识 | 例如2010,0000,0000 | id | integer |
| 用户名 |  | name | varchar |
| 密码 |  | password | tinytext |
| 状态 | 1：启动  0：未启用 | status | integer |
|  | 管理员  调度员 | privilege | Integer：  1：管理员；  2：调度员 |
| 调度员权限类型 | 全权限调度员 | role | Integer  1: 全权限；  2：部分权限； |
| 下辖调度台 | 允许为空 | Desk\_index |  |
| 时间 | 用户创建的时间 | crtime | timestamp |
| 描述 |  | description | text |

创建脚本：

CREATE TABLE governer

(

index serial primary key,

id character varying(30) NOT NULL,

name character varying(20) NOT NULL,

password text,

status integer,

role integer NOT NULL DEFAULT 2,

crtime timestamp with time zone,

description text

)

WITH (

OIDS=FALSE

);

ALTER TABLE governer

OWNER TO xf;

更新时间信息：

update governer set crtime = (current\_timestamp) where id = '20180001';

修改字段类型

alter table "member" alter COLUMN imgfileid type int ;

alter table governer alter COLUMN crtime type **timestamp(0)** with time zone;

select current\_timestamp(0) ;精确到秒级别。

#### 2.软交换服务器表switch：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 取值 | 字段 | 类型 |
|  |  | index | integer |
| 标识 |  | id | Varchar |
| 名称 |  | devname | integer |
| 类型 |  | type | Integer  默认为1 |
| IP |  | ip |  |
| 端口 |  | port | integer |

CREATE TABLE switch

(

index serial NOT NULL,

id character varying(30) NOT NULL,

devname character varying(20) NOT NULL,

type integer NOT NULL DEFAULT 1,

ip bigint, 唯一

port integer,

CONSTRAINT switch\_pkey PRIMARY KEY (index)

)

WITH (

OIDS=FALSE

);

ALTER TABLE switch

OWNER TO xf;

#### 3.设备挂载电话列表member：

说明：软交换设备下挂载的电话列表。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 取值 | 字段 | 类型 |
|  |  | index | integer |
| 调度号码 |  | callno | varchar |
| 名称 |  | id | varchar |
| 调度等级 | 0-9，9表示最高优先级 | grade | integer |
| 软交换服务器 | switch.index | refindex | integer |
|  |  |  |  |

CREATE TABLE member

(

index serial NOT NULL,

callno character varying(11) NOT NULL,

id character varying(30),

class integer,

switch\_index integer NOT NULL,

CONSTRAINT member\_pkey PRIMARY KEY (index),

CONSTRAINT member\_switch\_fkey FOREIGN KEY (switch\_index)

REFERENCES switch (index) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT range\_check\_class CHECK (class >= 0 AND class <= 9)

)

#### 5.调度台表desk：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 取值 | 字段 | 类型 |
|  |  | index | serial |
| 标识 | 例如2010,0000,0000 | id | varchar |
| 名称 |  | name | varchar |
| 调度等级 | 0-9，9表示最高优先级 | class | Integer，默认为0 |
| IP |  | ip | bigint |
| MAC |  | mac | macaddr |
| 描述 |  | description | text |
|  |  |  |  |

CREATE TABLE desk

(

index serial NOT NULL,

id character varying(30),

name character varying(20) NOT NULL,

class integer NOT NULL DEFAULT 0,

ip bigint,

mac macaddr,

CONSTRAINT desk\_pkey PRIMARY KEY (index),

CONSTRAINT unique\_desk\_ip UNIQUE (ip),

CONSTRAINT range\_check\_desk\_ip CHECK (class >= 0 AND class <= 9)

)

调度台线路表schedule\_line：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 取值 | 字段 | 类型 |
|  |  | index | integer |
| 鉴权号码 |  | callno | integer |
| 所属调度台 |  | foreign key: index | integer |
| 线路分组 |  | groupid | integer |

#### 6.调度台资源desk\_totalmem

说明：调度台下面管理的所有调度成员的资源集合，是整个集合。

注意：不同的调度台可能管理到相同的调度成员，例如调度台A：[1,2,3,4,5]，调度台B:[4,5,6,7,8]，4和5可能有重合的情况。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 取值 | 字段 | 类型 |
|  |  | ~~index~~ | ~~integer~~ |
| 调度成员(用户) |  | 外键：callno | integer |
| 所属调度台 |  | foreign key: index  desk\_index外键 | integer |

CREATE TABLE desk\_totalmem

(

callno character varying(11) NOT NULL,

desk\_index integer NOT NULL,

CONSTRAINT totalmem\_desk\_fkey FOREIGN KEY (desk\_index)

REFERENCES desk (index) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE

)

#### 7.调度分组deskgrp：

调度台下面的分组。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 取值 | 字段 | 类型 |
|  |  | index | integer |
| 标识 |  | id | varchar |
| 分组名称 |  | name | varchar |
| 列数 |  | column | integer |
| 所属调度台 |  | foreign key: index  desk\_index | integer |
| 描述 |  | description | text |

CREATE TABLE deskgrp

(

index serial NOT NULL,

id character varying(30),

name character varying(20) NOT NULL,

colnum integer NOT NULL DEFAULT 6,

desk\_index integer NOT NULL,

description text,

CONSTRAINT deskgrp\_pkey PRIMARY KEY (index),

CONSTRAINT grp\_desk\_fkey FOREIGN KEY (desk\_index)

REFERENCES desk (index) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE

)

#### 8. 调度分组下的调度成员deskgrp\_mem

问题：当软交换设备中的用户列表发生改变后，该调度成员是否还有存在必要。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 取值 | 字段 | 类型 |
|  |  | index | integer |
| 调度成员号码 |  | 外键：callno | character varying(11) |
| 调度分组 |  | deskgrp.index | integer |

DROP TABLE deskgrp\_mem;

CREATE TABLE deskgrp\_mem

(

index serial NOT NULL,

callno character varying(11) NOT NULL,

desk\_grp\_index serial NOT NULL,

CONSTRAINT deskgrp\_mem\_pkey PRIMARY KEY (index),

CONSTRAINT deskgrpmem\_deskgrp\_fkey FOREIGN KEY (desk\_grp\_index)

REFERENCES deskgrp (index) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE

)

## 远程访问postgresql

如果要远程通过*TCP/IP*来访问*Postgresql*数据库，需要做这两步操作：

*1*、修改数据库配置文件*pg\_hba.conf  
# vi /var/database/pgsql/data/pg\_hba.conf  
# "local" is for Unix domain socketconnections only  
local  all        all                              trust  
# IPv4 local connections:  
host   all        all        127.0.0.1/32          trust  
host   all        all         192.168.2.0/24       trust  
host   all        all        192.168.23.53/32      trust  
  
2*、启动*postgresql*的时候，加上*“-i”*这个参数 *$ /usr/local/pgsql/bin/postmaster -D/var/database/pgsql/data/ -i &*这样*,*就能通过*TCP/IP*来访问*Postgresql*数据库了。*Postgresql*默认的通讯端口是*5432*。

﻿﻿**插入前判断是否已经存在重复项：**

对于会员注册，我们经常会做唯一性验证，通常情况下我们有这两种方式：   
1. 数据库表设计的时候loginName增加唯一约束   
2. 注册之前先查一下然后再去进行插入操作

针对以上两种情况，第一种情况，增加约束的话，是最有效的，但如果表结构已经确定了，再修改起来反而不太好，所以这里不多说。

我们这里就只说第二种情况，这种情况看似无懈可击，但是确实有致命的bug，如果两个人同时填好注册信息，数123一起点注册按钮，那么极有可能出现的情况就是两个人都能够注册成功，然后数据库就产生了两条数据。

虽然这种概率比较低，但用户数量上去了，这种情况也是有可能出现的。

下面用sql的方式处理这个问题：

insert into member(id,loginName,password)

select nextval('S\_T\_MEMBER'),${loginName},${password}

from member

where not exists(select id from member where loginName = ${loginName});

--上面的sql是配置在xml文件中的，相信玩过SSH或者SSI的同学们应该都能够领会其中的精神，不过这里还是解释下。

--nextval('S\_T\_MEMBER')：下一个自增id

--${loginName}：传入的参数

--${password}：传入的参数

--如果where条件不满足，那么insert就不会执行，返回的影响条数必然是0

--如果where条件满足，那么就是说相同的loginName不存在，那么insert会执行select出来的结果集，而这个结果集肯定只有一条而且就是我们传进来的参数，返回的影响条数肯定是1

至此问题解决了。