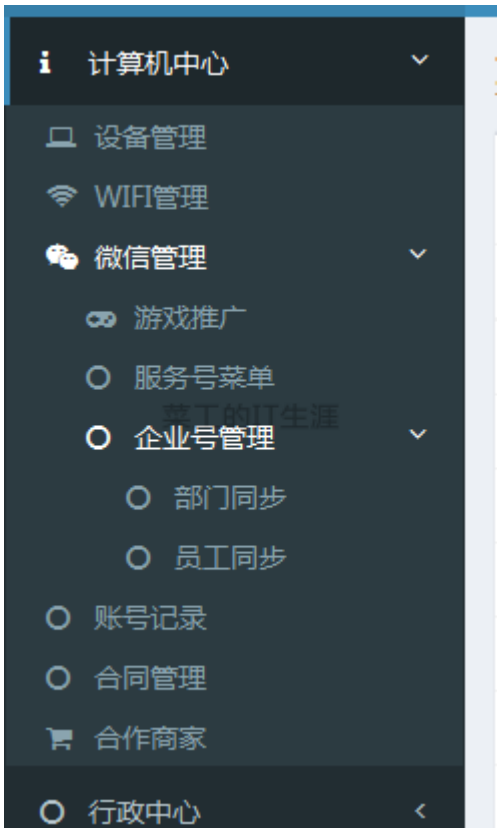
无限层级菜单—左右值树型数据结构

下面这个菜单是一个多层级菜单的，在 **计算机中心** 菜单下，有6个子菜单，在子菜单 **微信管理** 下面又有3个子菜单，子菜单 **企业号管理** 又有2个子菜单，层次可以无限地增加，而且层级也是不固定的。



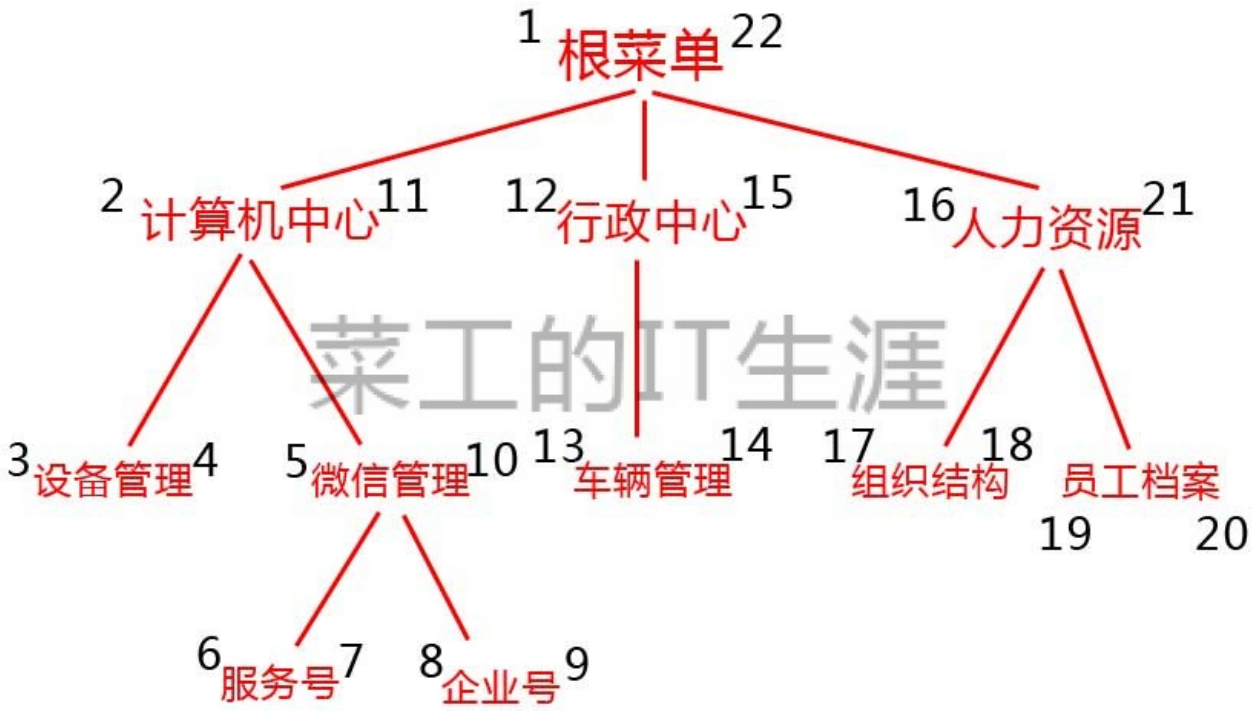
那像这样的菜单结构在数据库应该怎么存储呢。

树！没错，就是用树型数据结构，在之前的博客中，我也有写过对树型数据结构相关的想法（树型数组库结构设计），主要用 `parent_id` 字段记录父节点 id 值形成树成关系，为了方便查询不用使用递归，添加一个 `parent_ids` 辅助字段来记录节点的全部上级节点 id 。

这次我不想再用这种数据结构了，想尝试一下网上介绍的左右值的树型数据结构。

左右值意义

先看一下这张图



如果你一眼就看出每个数字的意义，那请受我一拜~~~

这些菜单上左右两边的黑色数字就分别是节点的左右值了

请沿着根菜单的左边开始数起（1），往计算机中心的左边（2），再设备管理左边（3），设备管理右边（4），微信管理左边（5），服务号左边（6）。。。大家是否发现了规律。

把每个节点左右都当成两个连接点，从根菜单左边开始，沿着整个树的节点外围连接，一直连回到根菜单的右边，所经过节点的顺序就是节点的左右值了。

（不知这样表达大家是否能理解，我能想到的最通俗的表达也是这样了，不管，我就当是理解了）

很好，大家都理解了。

那我上一下菜单的数据库表图

id	name	l	r	level	icon	controller
-1	菜单根目录	1	54	0		
2	计算机中心	2	25	1	info	
3	人力资源	32	39	1	users	(Null)
4	系统设置	44	53	1	gears	
5	设备管理	3	4	2	laptop	Equ
6	WIFI管理	5	6	2	wifi	Wifi
7	微信管理	7	18	2	weixin	(Null)
8	游戏推广	8	9	3	gamepad	WxGame
9	服务号菜单	10	11	3	circle-o	(Null)
10	组织结构	33	34	2	sitemap	Dep
11	员工管理	35	36	2	user	Employee
12	员工异动	37	38	2	user	(Null)
13	用户管理	45	46	2	user	User
14	角色管理	47	48	2	group	Role
15	菜单配置	49	50	2	sitemap	Menu
16	账号记录	19	20	2	circle-o	account
27	合同管理	21	22	2	circle-o	
28	企业号管理	12	17	3	circle-o	
29	部门同步	13	14	4	circle-o	WxOrg
30	员工同步	15	16	4	circle-o	
31	版本管理	51	52	2	circle-o	Version
32	行政中心	26	31	1	circle-o	
33	销售网络	27	28	2	circle-o	Salesoffice
34	车辆管理	29	30	2	car	
35	合作商家	23	24	2	shopping-cart	Seller
36	ERP系统	40	43	1	circle-o	
37	故障提交	41	42	2	bug	ErpBug

这里面还添加了一个 `level` 字段，这个大家都知道，这是节点的层级，根菜单的层级是0。

查询

让大家见识一下这种数据结构的魅力，你会发现查询变得如此的简单，曾经复杂的子孙节点查询，现在变得如果简单。

查询某一节点下面的全部子孙节点

```
//例如想查询第一张图上的计算机中心这个菜单底下全部的子菜单
//计算机中心的左值为2 右值为11 level 为 1

//查询节点下全部子孙菜单不包括自身菜单
SELECT * FROM menu WHERE l > 2  AND r < 11

//查询节点下全部子孙菜单包括自身菜单
SELECT * FROM menu WHERE l >= 2  AND r <= 11

//如果你只想查询子节点，孙节点不想查询到，那么在 where 条件 添加 level = 2
```

计算某一节点下有多少个子孙节点

例如想查询第一张图上的计算机中心这个菜单底下有多少个子菜单 计算机中心的左值为2 右值为11。

子菜单数量（不包括自身节点）

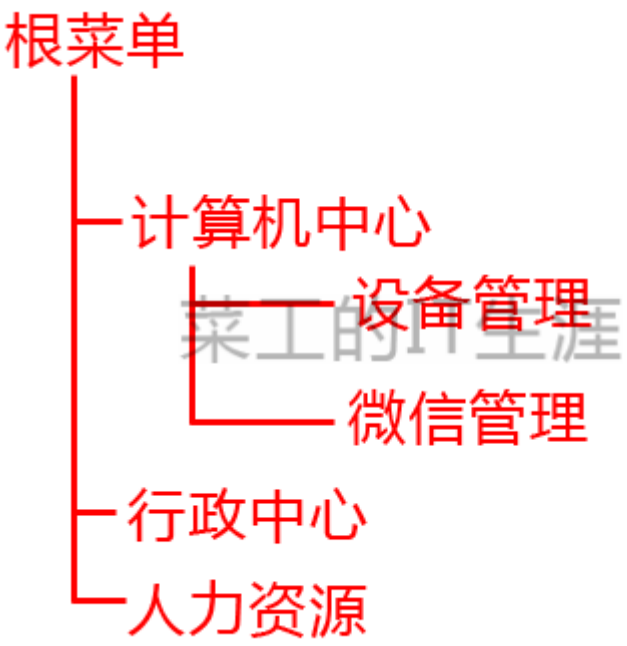
(右值 – 左值 – 1) / 2 => (11 – 2 – 1) / 2 = 4

子菜单数量（包括自身节点）

(右值 – 左值 + 1) / 2 => (11 – 2 + 1) / 2 = 5

节点排序

当你在 sql 查询时排序条件加上 order by l 的话，你会发现，返回的节点是按节点全部展开后由上往下的顺序，这个非常有用，在下面的 菜单跟树表格中非常关键。



查询某一节点的全部父祖节点

例如想查询第一张图上的 企业号 菜单的有哪些父级菜单

```
企业号 的左值为8 右值为9 level 为 1。

SELECT * FROM menu WHERE l < 8 AND r > 9 order by l

//返回菜单并按由上往下排序，分别为根菜单，计算机中心，微信管理

//如果想由下往上就把排序条件改成 order by r


//如果你只是想查询某一节点到某父节点经过的路径的话

//例如想查询 企业号 到 计算机中心 经过的菜单

SELECT * FROM menu WHERE l < 8 AND l > 2 AND r > 9 AND r < 11 order by l
```

判断

判断某一节点是否是某一节点的子节点

假如我想查询企业号menu1（左：8，右：9）是否是设备管理menu2（左：3，右：4）的子菜单。

判断式： menu1.l > menu2.l AND menu1.r < menu2.r

8 > 3 AND 9 <4 = false

说明企业号不是设备管理的子菜单

判断某一节点是否是某一节点的父节点

假如我想查询计算机中心menu1（左：2，右：11）是否是服务号menu2（左：6，右：7）的父菜单。

判断公式： menu1.l < menu2.l AND menu1.r > menu2.r

2 < 6 AND 11 > 7 = true

说明计算机中心是服务号的父菜单

判断某一节点是否有子节点

将节点的右值减去左值，如果大于1则说明有。

相信以上的查询及判断已经可以满足对数据的日常使用了，那下面就来一个实际例子，就是我上个博客说到的AdminLTE 后台菜单。

根据 AdminLTE 模板的 html 结构，以我第一个图为例，那生成的 html 代码如下所示（菜单不包括根菜单）

```
<ul class="sidebar-menu">
    <li class='treeview'>
        <a href='#'>
            <i class='fa fa-info'></i> <span>计算机中心</span><i class='fa fa-angle-left pull-
right'></i>
        </a>
        <ul class='treeview-menu'>
            <li>
```

```
                <a href='/index.php/Admin2/Equ/index.html'><i class='fa fa-laptop'></i> <span>设备管理</span></a>
            </li>
            <li class='treeview'>
                <a href='#'><i class='fa fa-weixin'></i> <span>微信管理</span><i class='fa fa-angle-left pull-right'></i></a>
                <ul class='treeview-menu'>
                    <li>
                        <a href='/index.php/Admin2/WxGame/index.html'><i class='fa fa-gamepad'></i> <span>服务号</span></a>
                    </li>
                    <li>
                        <a href='/index.php/index.html'><i class='fa fa-circle-o'></i> <span>企业号</span></a>
                    </li>
                </ul>
            </li>
        </ul>
    </li>
    <li class='treeview'>
        <a href='#'><i class='fa fa-circle-o'></i> <span>行政中心</span><i class='fa fa-angle-left pull-right'></i></a>
        <ul class='treeview-menu'>
            <li>
                <a href='/index.php/index.html'><i class='fa fa-car'></i> <span>车辆管理</span></a>
            </li>
        </ul>
    </li>
    <li class='treeview'>
        <a href='#'><i class='fa fa-users'></i> <span>人力资源</span><i class='fa fa-angle-left pull-right'></i></a>
        <ul class='treeview-menu'>
            <li>
                <a href='/index.php/Admin2/Dep/index.html'><i class='fa fa-sitemap'></i> <span>组织结构</span></a>
            </li>
            <li>
                <a href='/index.php/Admin2/Employee/index.html'><i class='fa fa-user'></i> <span>员工档案</span></a>
            </li>
        </ul>
    </li>
</ul><!-- /.sidebar-menu -->
```

一个菜单就是一个 `li` ，如果菜单有子菜单的话，就得加上 `treeview` 的 `css` 样式，然后在里面嵌套一个 `treeview-menu` 样式的 `ul` ，再写上菜单 `li` ，以此类推，然后哪个菜单是当前打开的就在其 `li` 上及全部上级菜单 `li` 加上 `active` 样式。

我需要拼结的就是 `<ul class="sidebar-menu"></ul>` 中间的那些 html ，那怎么把这些数据转变成菜单的 html 呢，理解上比递归还要难，现在回头看看我当时写的代码，我真感谢我当时有写注释。

```
public function getMenu(){
```

```
$admin_user=session("admin_user");
//如果当前用户不是超级管理员
if(-1!=$admin_user["id"]){
    $role_id=$admin_user["role_id"];
    $arr_menu_id=M("roleMenu")->where(array("role_id"=>$role_id))->getField("menu_id",true);
    if(!$arr_menu_id){
        return;
    }
    //到数据库查询权限对应菜单
    $arr_menu=M()->table("menu m1,menu m2")->distinct(true)->field("m1.*")->where(array(
    "m2.id"=>array("IN",$arr_menu_id),
    "m1.l"=>array("EXP","<=m2.l"),
    "m1.r"=>array("EXP",">=m2.r"),
    "m2.r-m2.l"=>array("EXP","=1"),
    "m1.level"=>array("gt","0")
    ))->order("m1.l")->select();
}else{
    //超级管理员直接获取全部菜单
    $arr_menu=D("Menu")->where(array("level"=>array("gt","0")))->order("l")->select();
}
    //上面是根据当前登录用户获取对应的菜单，大家可以不用理会，直接当成是超级管理员获得全部菜单

//在数组中查找当前的active的菜单
$current_menu;
$controller=CONTROLLER_NAME;//获取当前的控制器
if($controller!=="Index"){
    foreach ($arr_menu as $m) {
        if(CONTROLLER_NAME===$m["controller"]){
            $current_menu=$m;
            break;
        }
    }
}
$last_menu=null;//上一个菜单
$str_menu_html="";//菜单html
$end_index=count($arr_menu)-1;//最后一个菜单的索引
foreach ($arr_menu as $key => $m) {
    //判断
    if($last_menu){
        //判断当前菜单是否小于上一个菜单的层级
        //小于说明当前菜单已经离开上个菜单的li，需要加关闭标签
        //层次差的值等于需加上的关闭标签的数量
        $level_d=intval($last_menu["level"])-intval($m["level"]);
        if($level_d>0){
            $str_menu_html.=str_repeat("</ul></li>", $level_d);
        }
    }
    //判断当前菜单是否为是当前打开的菜单或者它的上级菜单
    if($current_menu){
        $is_active=false;
        if($m["l"]<=$current_menu["l"]&&$m["r"]>=$current_menu["r"]){
            $is_active=true;

        }
```

```
        }
        //判断当前菜单是否有子菜单
        if(intval($m["r"])-intval($m["l"])>1){
            //当前菜单有子菜单的
            $str_menu_html.="<li class='treeview".($is_active?" active":"")."'><a href='#'><i
class='fa fa-".$m["icon"]."'></i> <span>".$m["name"]."</span><i class='fa fa-angle-left pull-
right'></i></a><ul class='treeview-menu'>";
        }else{
            //当前菜单是最底层菜单了
            $str_menu_html.="<li".($is_active?" class='active'":"")."><a
href='".U($m["controller"]."/index")."'><i class='fa fa-".$m["icon"]."'></i>
<span>".$m["name"]."</span></a></li>";
        }
        //判断是否为最后一个菜单
        if($key===$end_index){
            $level_d=intval($m["level"])-intval($arr_menu[0]["level"]);
            if($level_d>0){
                $str_menu_html.=str_repeat("</ul></li>", $level_d);
            }
        }else{
            $last_menu=$m;
        }
    }
    return $str_menu_html;
}
```

上面的理解完，再来看下面这张图，想想这个 table 怎么写的，特别是菜单名称前面那些线，这可不是什么树插件哦，这些都是普通的文本。

我直接上代码吧，这代码我也快看不懂了，下面这个方法，可以获取一个带有 `line` 键名的菜单数组。

在模板中遍历数组时，就可以用 $menu[“line”] 来输出菜单名称前面的连接线。

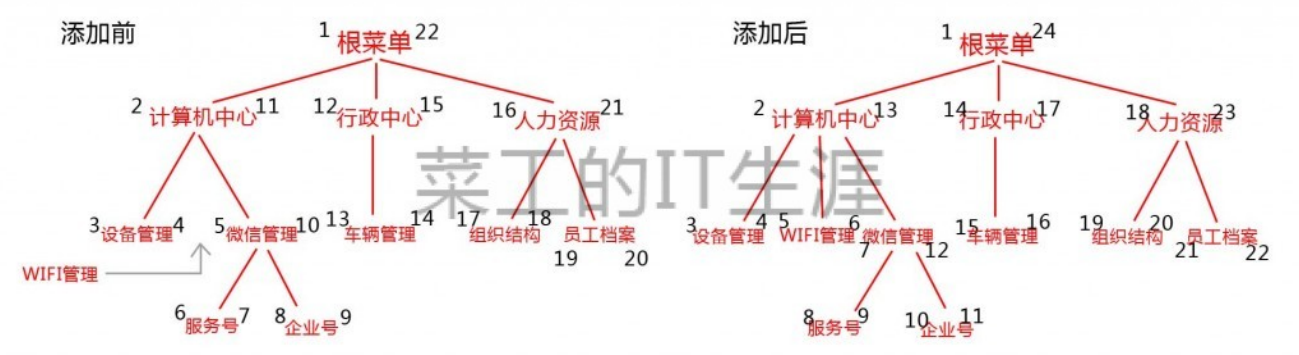
```
public function getMenuWithLine(){
    $arr_menu=D("Menu")->where(array("level"=>array("gt","0")))->order("l")->select();
    $arr_prefix=array();
    $arr_parent=array();
    foreach ($arr_menu as &$m) {
        //以下代码很吐血
        $line="";
        $parent=end($arr_parent);//取得当前分类菜单
        //当有分类菜单时
        if($parent){
            $level_d=$parent["level"]-$m["level"];//当前菜单跟分类菜单的层次差
            //当层次差大于等于0时，说明当前菜单跟当前分类菜单没有父子关系
            if($level_d>=0){
                //对应层次差从数组弹出对应数量的元素
                for($i=0;$i<$level_d+1;$i++){
                    array_pop($arr_prefix);
                    array_pop($arr_parent);
                }
                $parent=end($arr_parent);//重新取当前分类菜单
            }
            $is_level_end=$parent["r"]-$m["r"]===1?true:false;//当前菜单是否是本层次最后一个
        }
        //当有分类菜单时，说明当前菜单是子菜单，需加线
        if($parent){
            //当前菜单是分类中最后一个菜单时
            if($is_level_end){
                $line="└";
            }else{
                $line="├";
            }
            $m["line"]=implode("",$arr_prefix)." ".$line;
        }else{
            $m["line"]="";
        }

        $has_sub=$m["r"]-$m["l"]>1?true:false;
        //当前菜单有子菜单时
        if($has_sub){
            //将当前分类压入栈
            array_push($arr_parent,$m);
            //当前菜单是子菜单时，需加前缀
            if($parent){
                //当前菜单是分类中最后一个菜单时
                if($is_level_end){
                    array_push($arr_prefix,"       ");
                }else{
                    array_push($arr_prefix," |    ");
                }
            }
        }
    }
```

```
    return $arr_menu;
}
```

添加节点

查询是如此的优雅，但它的增删修就略显麻烦，这就是美丽的代价。

想想，现在我想添加个 **WIFI管理** 菜单到 **设备管理** 后面，如下图所示，整个树的数据会产生什么变化。



你会发现在 **WIFI管理** 后面的到 **根菜单** 的右边，所有的连接点都增加了 2。

因为一个节点有两个连接点，将这个新增节点放到树结构的任意位置后，新增节点后面的节点将受到影响，都增加了2。

下面是我基于 thinkphp 写的一个新增菜单的代码，传入一个父菜单的 Id 和新增菜单的信息，新增菜单将添加到父菜单的子级菜单的最后一个。

```
/*
根据传入的父节点 id ，在父节点的子节点最后的位置新增一个新节点
*/
public function add($parent_id,$menu_name){
    $d_menu=D("Menu");
    $d_menu->startTrans();

    //获取父节点信息
    $parent_menu=D("Menu")->field("r,level")->where(array("id"=>$parent_id))->find();
    $data=["name"=>$menu_name];

    //根据父节点计算出新节点的左右值及level
    $data["l"]=$parent_menu["r"];
    $data["r"]=$parent_menu["r"]+1;
    $data["level"]=$parent_menu["level"]+1;

    if(false===$d_menu->create($data)) $this->error($d_menu->getError());

    //将受到影响的节点左值加2
    if(false===M("menu")->where(array("l"=>array("gt",$parent_menu["r"])))->setInc("l",2)){
        $d_menu->rollback();
        $this->error($d_menu->getError());
    }
    //将受到影响的节点右值加2
    if(false===M("menu")->where(array("r"=>array("egt",$parent_menu["r"])))->setInc("r",2)){
```

```
            $d_menu->rollback();
            $this->error($d_menu->getError());
        }
        if(false===$d_menu->add()) {
            $d_menu->rollback();
            $this->error($d_menu->getError());
        }
        $d_menu->commit();
        $this->success("添加成功",U("Menu/index"));

    }
```

删除节点

删除节点跟新增节点方向相反，删除节点后，将受到影响的节点减去被删除节点的影响值（一个节点两个连接点，就是2，如果移动的节点是父节点，那么影响值就是右值 - 左值 + 1）。

下面是我写的删除菜单的代码，但我这个限制了只能删除末级菜单，就是没有子菜单的菜单。

```
public function del($menu_id){
    if(IS_POST){
        $r=D("Result");
        try{
            $d_menu=D("Menu");
            $d_menu->startTrans();
            $menu=$d_menu->find($menu_id);
            if(!$menu) throw new \Exception("查无此菜单！");
            if($menu["r"]-$menu["l"]>1) throw new \Exception("当前菜单含有子菜单！");
            $field_r=$menu["r"];
            if(false===$d_menu->delete($menu_id)) throw new \Exception($d_menu->getError());
            if(false===$d_menu->where(array("l"=>array("gt",$field_r)))->setDec("l",2))throw new
\Exception($d_menu->getError());
            if(false===$d_menu->where(array("r"=>array("gt",$field_r)))->setDec("r",2))throw new
\Exception($d_menu->getError());
            if(false===M("roleMenu2")->where(array("menu_id"=>$menu_id))->delete()) throw new
\Exception($d_menu->getError());
            $d_menu->commit();
        }catch(\Exception $e){
            $d_menu->rollback();
            $r->success=false;
            $r->message=$e->getMessage();
        }
        $this->ajaxReturn($r);
    }
}
```

移动节点位置

接下来才是最难以理解的地方。

如果我现在想把 **组织结构** 菜单移动到 **计算机中心** 菜单下，**微信管理** 菜单前，如下图所示，那会对整个树产生什么影响呢。

比对一下两边的节点的左右值，会发现，受到影响的节点都是移动节点的原位置到目标位置中间经过路径的节点。

其实移动节点位置最重要的就是计算出受到影响的节点，然后计算机移动节点的影响值（一个节点两个连接点，就是2，如果移动的节点是父节点，那么影响值就是右值 - 左值 + 1），根据移动节点的相对位置（前、后、子级）加或减去这个影响值。

我还是直接上代码吧，这个方法需要传入一个移动节点的 Id，一个目标节点的 Id，一个相对于目标节点的位置。

比如我想把 **组织结构** 移动到 **微信管理** 前面，那么 **组织结构** 就是 **移动节点**，**微信管理**就是 **目标菜单**，**前面** 就是 **相对位置。**

大家自己理解，我很难用言语来讲述这坨代码。

```
/*
该方法是将菜单移动到目标菜单的前面/后面/子级
$position 源菜单移动到目标菜单的相对位置 0 前面; 1 后面; 2 子级
$target_menu_id 目标菜单id
$source_menu_id 移动菜单的id
*/
public function move($position,$target_menu_id,$source_menu_id){
    if($position===null||!$target_menu_id||!$source_menu_id)$this->error("缺少参数! ");

    $effect_left;//所有受到影响的节点最左值
    $effect_right;//所有受到影响的节点最右值
    $source_d;//源菜单需要加或减去的值

    $d_menu=D("Menu");
    $target_menu=$d_menu->find($target_menu_id);//获取移动到的目标菜单
    $source_menu=$d_menu->find($source_menu_id);//获取移动的源菜单
    $level_d=$target_menu["level"]-$source_menu["level"];//目标菜单与源菜单的层次差
    $effect_value=$source_menu["r"]-$source_menu["l"]+1;//所有受到影响节点要加或减的影响值
    //判断如果目标菜单等于或在源菜单子级的话，则报错
    if($target_menu["r"]<=$source_menu["r"]&&$target_menu["l"]>=$source_menu["l"]){
        $this->error("不能选择其本身或子级菜单! ");
    }
    $target_position=false;
    //判断目标菜单相对源菜单的位置，0为前面 1为后面 2为父祖先级
    if($source_menu["r"]>$target_menu["r"]&&$source_menu["l"]>$target_menu["l"]){
        $target_position=0;
    }elseif($source_menu["r"]<$target_menu["r"]&&$source_menu["l"]<$target_menu["l"]){
        $target_position=1;

    }elseif($source_menu["r"]<$target_menu["r"]&&$source_menu["l"]>$target_menu["l"]){
```

```
        $target_position=2;
    }
    if($target_position===false)$this->error("菜单位置判断失败！");
    switch($target_position){
        //目标菜单在源菜单前面
        case 0;
            //目标菜单位于源菜单的前面
            $effect_right=$source_menu["l"]-1;
            switch($position){
                //移至目标菜单的前面
                case 0:
                    $effect_left=$target_menu["l"];
                    break;
                //移至目标菜单的后面
                case 1:
                    $effect_left=$target_menu["r"]+1;
                    break;
                //移至目标菜单的子级最后一个菜单
                case 2:
                    $effect_left=$target_menu["r"];
                    $level_d++;
                    break;
            }
            $source_d=$effect_right-$effect_left+1;
        break;
        //目标菜单位于源菜单后面
        case 1:
            $effect_left=$source_menu["r"]+1;
            switch($position){
            //移至目标菜单的前面
            case 0:
                $effect_right=$target_menu["l"]-1;
                break;
            //移至目标菜单的后面
            case 1:
                $effect_right=$target_menu["r"];
                break;
            //移至目标菜单的子级最后一个菜单
            case 2:
                $effect_right=$target_menu["r"]-1;
                $level_d++;
                break;
            }
            $source_d=-($effect_right-$effect_left+1);
            $effect_value=-$effect_value;
            break;
        //目标菜单位于源菜单父级或祖先级
        case 2:
            switch($position){
            //移至目标菜单的前面
            case 0:
                $effect_right=$source_menu["l"]-1;

                $effect_left=$target_menu["l"];
```

```
                $source_d=$effect_right-$effect_left+1;
                break;
            //移至目标菜单的后面或子级
            case 1:
            case 2:
                $effect_left=$source_menu["r"]+1;
                if($position==1){
                    $effect_right=$target_menu["r"];
                }else{
                    $effect_right=$target_menu["r"]-1;
                    $level_d++;
                }
                $source_d=-($effect_right-$effect_left+1);
                $effect_value=-$effect_value;
                break;
            }
        break;
    }
    //如果影响节点最右值小于最左值，说明位置不变
    if($effect_right<$effect_left){
        $this->error("菜单位置无变化！",U("index"));
    }
    $d_menu->startTrans();//启用事务

    //获取源菜单包括子级菜单的id，方便后面更新
    $arr_source_id=$d_menu->where(array(
        "l"=>array("egt",$source_menu["l"]),
        "r"=>array("elt",$source_menu["r"])
    ))->getField("id",true);
    //更新所有影响到的节点左值
    if(false===$d_menu->where(array("l"=>array("between",array($effect_left,$effect_right))))-
>setInc("l",$effect_value)){
        $d_menu->rollback();
        $this->error($d_menu->getError());
    }
    //更新所有影响到的节点右值
    if(false===$d_menu->where(array("r"=>array("between",array($effect_left,$effect_right))))-
>setInc("r",$effect_value)){
        $d_menu->rollback();
        $this->error($d_menu->getError());
    }

    //源菜单更新
    $update_data=array(
        "l"=>array("exp","l-".$source_d),
        "r"=>array("exp","r-".$source_d)
    );
    if($level_d){
        $update_data["level"]=array("exp","level+".$level_d);
    }
    if(false===$d_menu->where(array("id"=>array("in",$arr_source_id)))->save($update_data)){
        $d_menu->rollback();

        $this->error($d_menu->getError());
```

```
    }
    //对修改后的树进行检测，防止写错，导致整个整数据乱了
    if($d_menu->group("l")->having("count(*)>1")->count())$this->error("修改失败！请联系计算机中心处理！");
    if($d_menu->group("r")->having("count(*)>1")->count())$this->error("修改失败！请联系计算机中心处理！");

    $d_menu->commit();
        $this->success("菜单位置修改成功！",U("index"));
}
```

总算写完了，这就是我对于左右值树型数据结构的一些想法跟实践，希望对大家有所帮助。