一、权限系统

这一天将讲述一个基本的基于数据库的权限管理系统的设计，在这一天的课程的最后将讲述“左右值无限分类实现算法”如何来优化“系统菜单”的结构而告终。今天的内容和前几天的基础框架是一样的它们都属于基础知识，在这些基础知识上还可以扩展出无数的变种与进化设计。

二、先来看客户的一个需求

2.1 用户实际需求

1.所有的用户、角色可动态配置

2.所有的系统菜单的权限要求具体到“增，删，改、查、打印、导出”这样的小权限的设计

3.所有的权限基于角色来进行划分和判断

4. 一个用户可能属于多个角色

5.系统菜单也能够动态的“增、删、改、查”

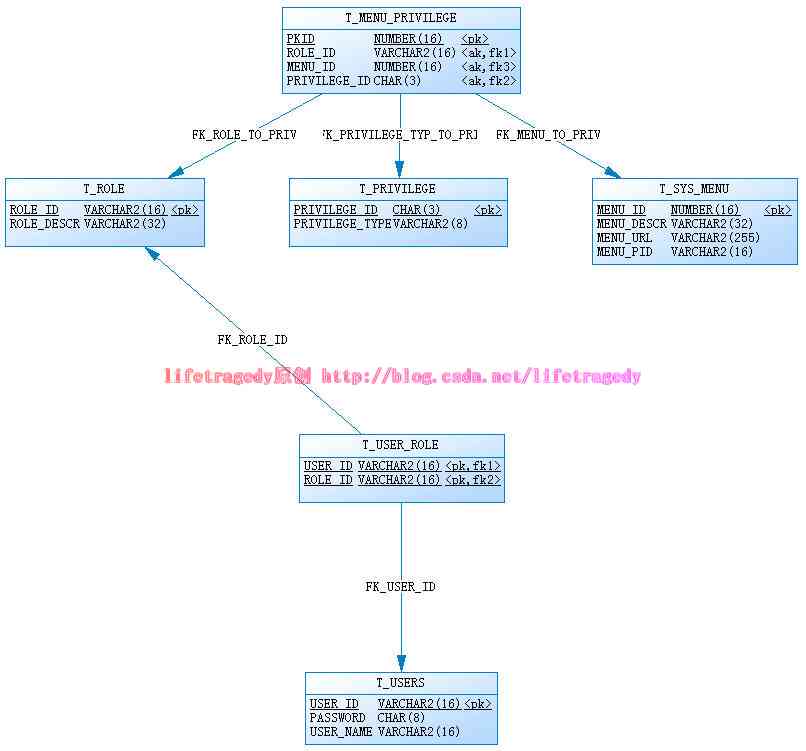
2.2 系统权限菜单样例



三、基于数据库的系统权限表设计

3.1 ER(Entity Relationship)图

围绕上述需求，我们可以在数据库内进行如下的表设计，下面直接给出ER图：

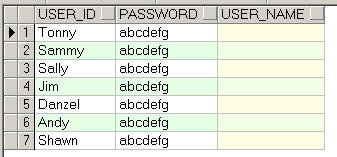


3.2 表关系详解

上述设计有6张表，其中：

T\_User表

用于存放用户信息，此处只存放基础信息



T\_Role表

用于存放系统角色信息



T\_User\_Role表

用于存放系统用户与角色的匹配关系



T\_Sys\_Menu表

这张就是用于存放系统菜单的表了，这张表的设计主要使用了如下的表设计技巧：



注意这边的MENU\_ID与MENU\_PID

如果这个菜单项是一级菜单，那么我们把它的MENU\_PID设为0

如果这个菜单是另一个菜单的子菜单，那么我们就把它的MENU\_PID设为它的父菜单的MENU\_ID。

有了这样的结构，我们一个递归就能把这颗“树”显示出来了，是不是？

此处以Oracle数据库为例，不使用递归，直接把树形结构在数据库中就造型造好（当然，还有更好的方法如：有人喜欢设level或者是deep这样的字段来简化程序解析树型结构菜单，稍后我们会来讲一个根本不需要用递归的树型菜单的设计来最大程度优化设计。）

显示整颗树型菜单结构的Oracle语句：

|  |
| --- |
| SELECT \*FROM T\_SYS\_MENU  STARTWITH MENU\_PID=0  CONNECTBYPRIOR MENU\_ID=MENU\_PID  orderby MENU\_ID |



上述语句，已经用数据查询用句就将我们的这个“树”的层次关系理出来了，如果我们手上有一个控件叫dtree.js，那么一个循环就可以把这个树显示出来了，不是吗？

来看dtree.js的应用

|  |
| --- |
| //公式:  d.add(menu\_id, menu\_pid, ‘menudescr’, ‘menu\_url’);  d = new dTree('d');                     d.add(0,-1,'菜单');                     d.add(1,0,'报表查询','example01.html');                     d.add(2,1,'月报','example01.html');                     d.add(3,1,'季报','example01.html');                     d.add(4,1,'年报','example01.html');                     d.add(5,0,'系统管理','example01.html');                     d.add(6,5,'用户管理','example01.html');                     d.add(7,6,'新增用户','example01.html');                     d.add(8,6,'删除用户','example01.html');                d.add(9,5,'角色管理','example01.html');                     d.add(10,9,'新增角色','example01.html');                     d.add(11,9,'删除角色','example01.html')                     document.write(d); |

大家看上面，这个是dtree.js插件，一个专门用于生成树的js插件的使用方法，那么如果我们附以上述的sql语句在数据库中把数据选出来后，是不是只要一个循环就可以给这个dtree.js插件显示了，不是吗？

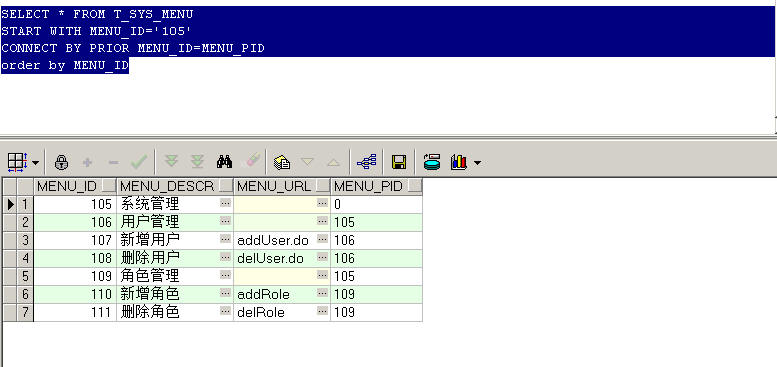
我们如果不想显示整颗树只想显示如：



只显示系统管理菜单有其下列所有的子菜单，那么我们的Oracle中的Sql应该怎么写呢？

经查“系统管理”这个菜单的MENU\_ID=’105’，于是我们的Sql语句如下：

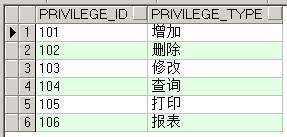
|  |
| --- |
| SELECT \*FROM T\_SYS\_MENU  STARTWITH MENU\_ID='105'  CONNECTBYPRIOR MENU\_ID=MENU\_PID  orderby MENU\_ID |



对吧？很简单哈！

T\_Privilege表

用于存放系统每个菜单的详细子权限如“增，删，改，查”



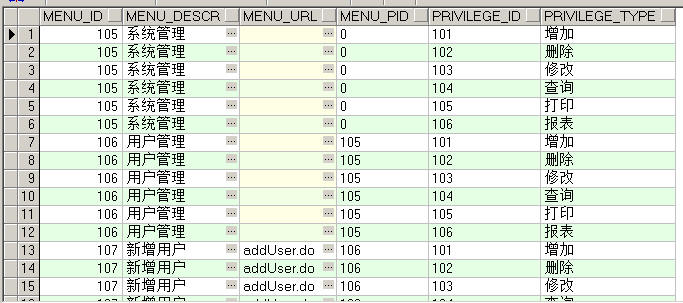
T\_Menu\_Privilege表

这张表就是我们的最终终结大BOSS表，它里面是一个完整的系统权限与角色关系间的对应。

比如说：

我们想要知道“user”这个角色，可以操作哪些菜单，哪些权限，那么我们的SQL语句如下：

|  |
| --- |
| selectdistinct m.menu\_id,m.menu\_descr,m.menu\_url,m.menu\_pid,p.privilege\_id,p.privilege\_type from  t\_menu\_privilege mp,  t\_sys\_menu m,  t\_privilege p,  t\_user\_role r  where  mp.privilege\_id=p.privilege\_id  and mp.role\_id=r.role\_id  and mp.menu\_id=m.menu\_id  and r.role\_id='user'  orderby m.menu\_id |



通过这个结果我们就知道了

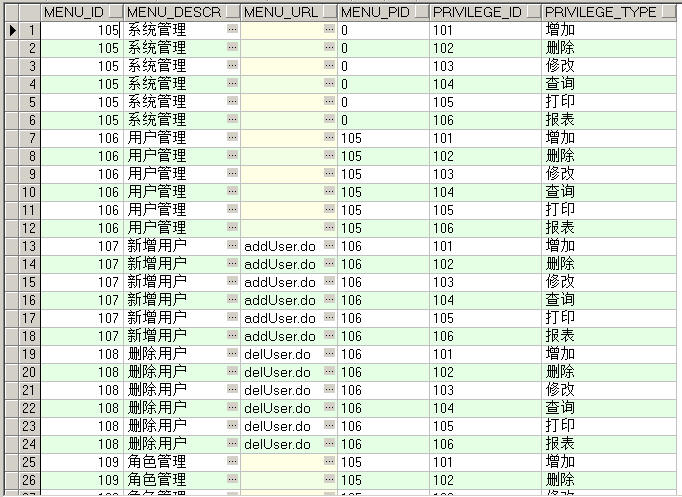
1. 角色“user”能操作哪些菜单

2. 角色“user”对某个菜单具有什么样的权限

进而，我们可以推出：

我们想要知道Danzel这个人，可以操作哪些菜单，以及在哪些菜单上有哪些可供操作的权限，我们使用如下的SQL语句：

|  |
| --- |
| selectdistinct m.menu\_id,m.menu\_descr,m.menu\_url,m.menu\_pid,p.privilege\_id,p.privilege\_type from  t\_menu\_privilege mp,  t\_sys\_menu m,  t\_privilege p,  t\_user\_role r  where  mp.privilege\_id=p.privilege\_id  and mp.role\_id=r.role\_id  and mp.menu\_id=m.menu\_id  and r.user\_id='Danzel'  orderby m.menu\_id; |



通过这个结果我们就知道了

1. Danzel这个人能操作哪些菜单

2. Danzel对某个菜单具有什么样的权限

3.3 界面制作

关于jsp，什么dao层，service层的具体代码这个就不讲了，这个没有意义的哦，我们来讲设计。

登录后如何显示左边的树型菜单：

ü 取得用户名

ü 将该用户名作为参数input进如下的SQL语句得到该用户在登录后可以看到的系统菜单：

|  |
| --- |
| selectdistinct m.menu\_id, m.menu\_descr, m.menu\_url, m.menu\_pidfrom  t\_menu\_privilege mp,  t\_sys\_menu m,  t\_privilege p,  t\_user\_role r  where  mp.privilege\_id=p.privilege\_id  and mp.role\_id=r.role\_id  and mp.menu\_id=m.menu\_id  and r.user\_id='Danzel'  STARTWITH MENU\_PID='0'  CONNECTBYPRIOR M.MENU\_ID=M.MENU\_PID  orderby M.MENU\_ID |

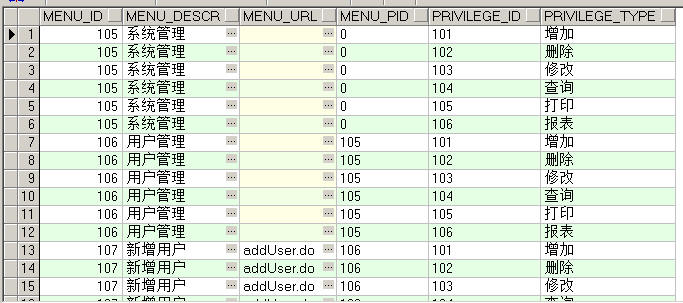


将该结果直接给于index.jsp页面上的dtree.js组件，一个循环，所有菜单曾树形显示。

**知道用户登录后能够对哪些菜单，并且在相关界面操作时有哪些子权限如：增、删、改、查、打印、报表的设计：**

ü   在登录时得到用户名等信息，然后将该用户名作为参数input进入如下的sql语句：

|  |
| --- |
| selectdistinct m.menu\_id,m.menu\_descr,m.menu\_url,m.menu\_pid,p.privilege\_id,p.privilege\_type from  t\_menu\_privilege mp,  t\_sys\_menu m,  t\_privilege p,  t\_user\_role r  where  mp.privilege\_id=p.privilege\_id  and mp.role\_id=r.role\_id  and mp.menu\_id=m.menu\_id  and r.user\_id='Danzel'  orderby m.menu\_id; |



ü 得到上述结果后，使用：Haspmap<String menuId, List menuList>这样的结构将该用户所属的角色分对每个菜单有哪些操作（增、删、改、查、打印、报表）进行存储，放入用户的session中，在以后用户在每个界面进行点击动作时进行判断，或者可以写个filter来进行判断，是不是就可以作到：

知道该登录用户在登录后可以对哪些菜单进行操作，并且拥有什么操作权限啦？

相应的我们还需要制作如下的界面：

ü 用户的管理界面

ü 角色的管理界面

ü 用户与角色的分配界面

ü 系统菜单的管理界面

ü 具体权限项的管理界面

ü 系统菜单与角色间具体的权限分配界面

好了，有了这些界面，一个完整的基于数据库引擎的权限系统算是完成了。

严重注意：

在制作“系统菜单与角色间具体的权限分配界面”时，如果在界面上把某个角色对该条菜单的“查看”这个选项disable后，那么该角色将不拥有任何该菜单的所有权限，举例：

某角色对菜单A拥有如下权限：

增、删、改、打印

但是这个“查看”权限没有，也有可能是管理员误操作，但是从真实情况我们来说，这个角色连“查看”的权限都没有，连菜单都进不了，他能做什么“增、删、改。。。”等其它的操作啊？操作个头啊！是吧？

所以一旦界面上该角色对某个系统菜单没有了查看权限后，它对这个菜单的其它权限也必须从T\_MENU\_PRIVILEGE这个表中删除。

四、改进T\_SYSTEM\_MENU表的设计

前面我们用的是Oracle特有的递归SQL将树形菜单在从数据库中选取出来时就已经是一颗树的结构了，但是像MySQL，SQL SERVER， DB2等可能不带有这样的特SQL，那就需我们自己动手去写递归，还有就是很多工程用的是jQuery的tree或者是其它相关的ajax tree，这些tree都需要用到一个字段叫level（此处指深度、层次的意思），如果按照原来的表结构，要取得这个level，恐怕是要写递归算法了。就算有些数据库有类似的语句，那也需要你去修改你的SQL语句从未影响了性能与通用性。

我们在这边说，我们无论什么数据库，如果都用相同的SQL就能把我们需要的东西在数据库中就排好树形结构然后一次性选取出来，那应该有多好啊。答案是有的，在原来的T\_SYSTEM\_MENU表中改动也不大，只需要增加两个字段即可，即：lft与rgt（left, right），这种设计其实已经有了，我在此只不过结合实际例子把它应用到实际上，并且进一步详细描述如果来实现它，它就是被称为：

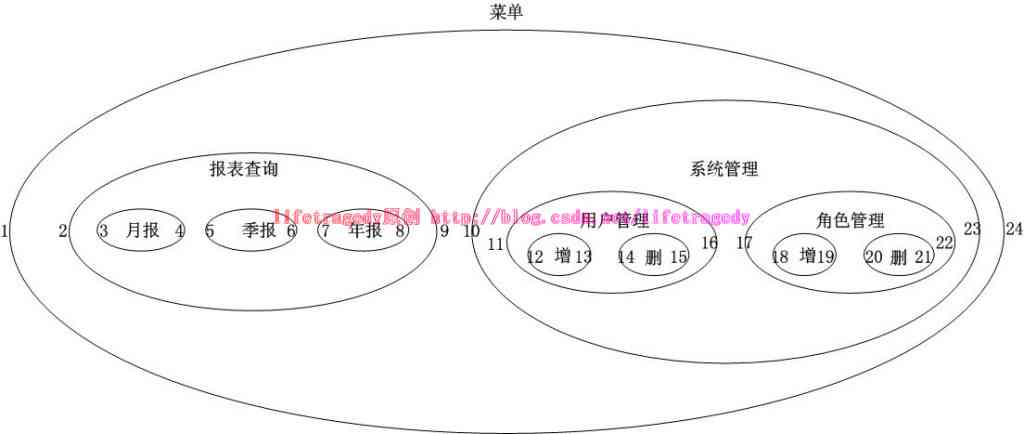
左右值无限分类实现算法也称为预排序遍历树算法，对于这种层次型数据（Hierarchical Data）一般我们有两种设计方法：

ü 毗邻目录模式(adjacencylist model)

ü 预排序遍历树算法(modifiedpreorder tree traversal algorithm)

4.1 基于lft, rgt的无限分类算法

我们来看一个图，下面我们把我们原有的菜单画成下面这样的层次关系：



我们把原有的系统菜单画成了一个个的椭圆，最外层的就是我们的菜单，然后在每个椭园的两个端点即left与right，按照从左->右，开始用数字来标号，上面这个图中可以看到最外层这个大椭园的lft(左)为1，它的rgt(右)为24。

那么我们可以用一条标准的SQL，而非什么数据库自带的特有的、特殊的SQL来显示出这个树形菜单，来看下面的SQL：

|  |
| --- |
| SELECT         node.menu\_id menuId,         node.menu\_descr menuDescr,         node.lft,         node.rgt,         node.menu\_url menuUrl,         (COUNT(parent.menu\_id)-1) menuLevel,         node.menu\_pid pid  FROM t\_sys\_menu node,         t\_sys\_menu parent  WHERE          node.lftBETWEEN parent.lftAND parent.rgt          AND node.menu\_descr!='菜单'  GROUPBY node.menu\_id,node.menu\_descr,node.lft,node.rgt,node.menu\_url,node.menu\_pid  ORDERBY node.lft |



看看上面这个结果，怎么样？

ü 树形结构也有了（可以用于dtree来显示）；

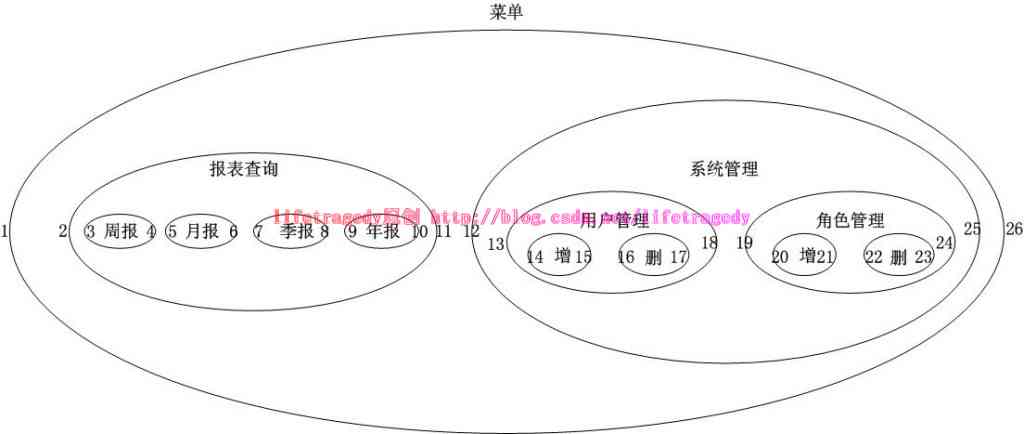
ü 层次level也有了（可以用于ajax的一些tree）；

ü 我们用的SQL又是最标准的所有的数据库都能用到的SQL；

尝到甜头了是吧？那我们下面来看如何对这样的基于t, rgt的数据结构来作插入操作？

4.2 如何在现有节点中插入新的子节点

如果现在我们要在“报表查询”这个圆里加入一个菜单，假设我们就叫“周报”，那么再来看这个原有的图发生了什么样的改变，来看：



看到么，原有的最外层椭园的rgt+2，原有的报表查询这个园的右边界呢？是不是也加了2啊？而原有的“月报”这个圆的lft加了多少？也是+2！

那么来看“周报”这个圆的lft与rgt关系：

“周报”的lft= “报表查询”这个圆的lft+1

“周报”的rgt=“报表查询”这个圆的lft+2

于是我们就可以整理出在原有叶子中插入child的公式：

第一步：选取要被插入new child的外面这个圆的lft的值

第二步：原有的数据中所有的rgt如果>第一步中得到的lft的值，那么全部+2

第三步：原有的数据中所有的lft如果>第一步中得到的lft的值，那么全部+2

第四步：将插入的节点的lft与rgt的设计，新节点的lft =第一步中的lft+1，新节点的rgt=第一步中

的lft+2

来看一个具体的例子：

我们要在“报表查询”即menu\_id=’101’ 中插入一个新的菜单，叫“周报”，下面是按照上面四步算法的相关SQL语句：

第一步

SELECT lftFROM t\_sys\_menuwhere menu\_id='101';

这一步我们得到的值为：2

第二步：

UPDATE t\_sys\_menuSET rgt = rgt +2WHERE rgt >2;

第三步：

UPDATE t\_sys\_menuSET lft = lft +2WHERE lft >2;

第四步：

INSERTINTO t\_sys\_menu(menu\_id, menu\_descr, menu\_url, lft, rgt)VALUES('113','周报','周报的url', (2+1), (2 +2));

插完后我们运行查询SQL：

SELECT

node.menu\_id menuId,

node.menu\_descr menuDescr,

node.lft,

node.rgt,

node.menu\_url menuUrl,

(COUNT(parent.menu\_id)-1) menuLevel,

node.menu\_pid pid

FROM t\_sys\_menu node,

t\_sys\_menu parent

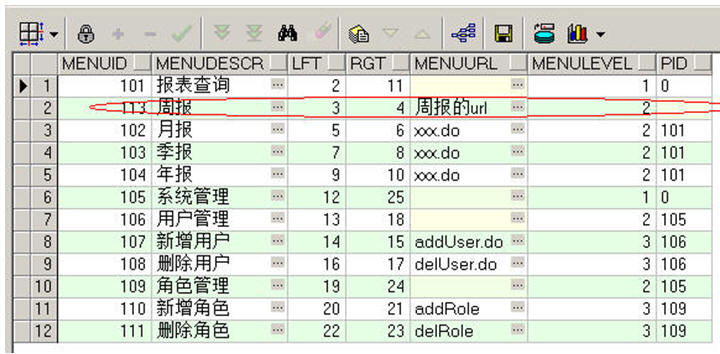
WHERE

node.lft BETWEEN parent.lft AND parent.rgt

AND node.menu\_descr!='菜单'

GROUPBY node.menu\_id,node.menu\_descr,node.lft,node.rgt,node.menu\_url,node.menu\_pid

ORDERBY node.lft



Look, 数据正确无误，我们来看整个t\_sys\_menu表里的数据：



Look，整个最外层的“圆”，右边界增加了2，从原有的24变成了26。

1.3 如何插入一个新的节点

上面讲的是在原有的节点中插入一个子节点，现在来讲，如何插入一个新的节点，比如说：

我们现在有：报表查询，系统管理两大菜单，我们还想加一个菜单：保单审核，怎么来做？

我们把4.2节中“如何在现有节点中插入新的子节点”里四步公式，稍稍改动一下

第一步：选取要被插入新的节点左边节点的rgt的值

第二步：原有的数据中所有的rgt如果>第一步中得到的rgt的值，那么全部+2

第三步：原有的数据中所有的lft如果>第一步中得到的rgt的值，那么全部+2

第四步：将插入的节点的lft与rgt的设计，新节点的lft =第一步中的rgt+1，新节点的rgt=第一步中

的rgt+2

下面来看我们在“报表查询”与“系统管理”中间，插入一个菜单叫“保单审核”。

第一步

SELECT rgtFROM t\_sys\_menuwhere menu\_id='101';

这一步我们得到的值为：11

第二步：

UPDATE t\_sys\_menuSET rgt = rgt +2WHERE rgt >11;

第三步：

UPDATE t\_sys\_menuSET lft = lft +2WHERE lft >11;

第四步：

INSERTINTO t\_sys\_menu(menu\_id, menu\_descr, menu\_url, lft, rgt)VALUES('114','保单审核','', (11+1), (11 +2));

运行下面的SQL语句我们来检查一下插入的效果：

SELECT

node.menu\_id menuId,

node.menu\_descr menuDescr,

node.lft,

node.rgt,

node.menu\_url menuUrl,

(COUNT(parent.menu\_id)-1) menuLevel,

node.menu\_pid pid

FROM t\_sys\_menu node,

t\_sys\_menu parent

WHERE

node.lftBETWEEN parent.lftAND parent.rgt

AND node.menu\_descr!='菜单'

GROUPBY node.menu\_id,node.menu\_descr,node.lft,node.rgt,node.menu\_url,node.menu\_pid

ORDERBY node.lft



怎么样，结果对了吧，呵呵！

看看整个菜单的右边界吧，从原来的26变成了28了，是不是哦？

1.3 如何删除一个节点

来看公式

第一步：选取要被删除的菜单的lft的值，rgt的值，以及宽度（width=rgt-lft+1）；

第二步：删除所有的位于第一步中得到的lft与rgt之间的节点；

第三步：将所有的右边界大于第一步中得到的rgt的所有节点的rgt的值减去第一步中得到的width

第四步：将所有的左边界大于第一步中得到的rgt的所有节点的lft的值减去第一步中得到的width

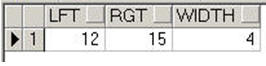
来看实际例子，我们有下面这样的数据：



我们想将menu\_id=114的保单审核删除，当然，这是一个父节点，如果把它删了，其子节点115即手工审核也必须被一起删除，要不然它就成为脏数据了是不是？套用上述四步公式：

第一步：

SELECT lft, rgt, (rgt - lft +1) widthFROM t\_sys\_menuWHERE menu\_id ='114'



第二步：

DELETEFROM t\_sys\_menuWHERE lftBETWEEN12AND15

第三步：

UPDATE t\_sys\_menuSET rgt = rgt -4WHERE rgt >15

第四部：

UPDATE t\_sys\_menuSET lft = lft -4WHERE lft >15

全部步骤完成后，我们来运行检验的SQL：

SELECT

node.menu\_id menuId,

node.menu\_descr menuDescr,

node.lft,

node.rgt,

node.menu\_url menuUrl,

(COUNT(parent.menu\_id)-1) menuLevel,

node.menu\_pid pid

FROM t\_sys\_menu node,

t\_sys\_menu parent

WHERE

node.lftBETWEEN parent.lftAND parent.rgt

AND node.menu\_descr!='菜单'

GROUPBY node.menu\_id,node.menu\_descr,node.lft,node.rgt,node.menu\_url,node.menu\_pid

ORDERBY node.lft



结果正确，再来看整个“菜单”的边界，从原来的28缩减成了26了，结果正确。

上述这种基于lft, rgt左右值无限分类实现算法的个菜单的好处在于：

ü SQL语句不受特定的数据库的限制

ü SQL语句通用

ü 直接从数据库中远取出来的结构化的数据即可满足需要pid的如：dtree.js这样的JS控件的需要也可以满足需要level的ajax tree控件的需要。