封装的还是比较好的，也不存在任何依赖，使用CPP的时候只需要定义两个必须的变量

schema sch;//用于保存数据字典

query\_tree original\_tree;//保存存下来的树

然后把所有代码都复制下来，调用两个函数：

1.init\_schema(); //把数据字典保存到树中

2.query\_tree original\_tree = get\_original\_tree(input);

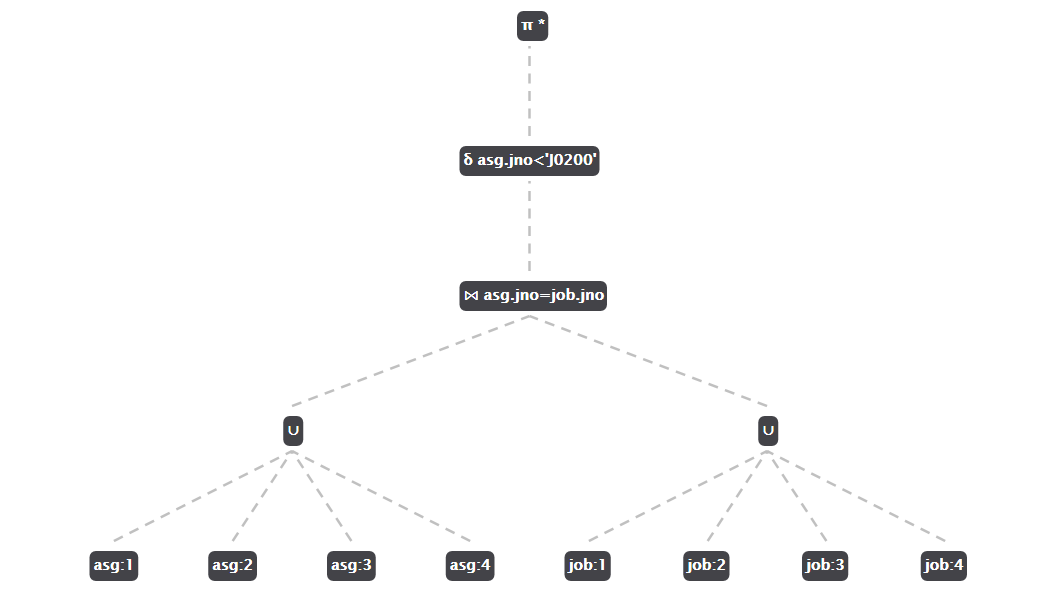
//input是输入的sql，注意这个sql必须被parser过

**具体数据格式参考代码前半部分的数据定义**

如果想要看树的结构，则最后调用一下print\_tree(original\_tree);来把树的结构输入到一个文件中。提供了一个tree.jsp和tree.js用来展示,将他们放在同一文件夹下。



把jsp的这个地方改成输出文件的位置即可。结果大概如下图：

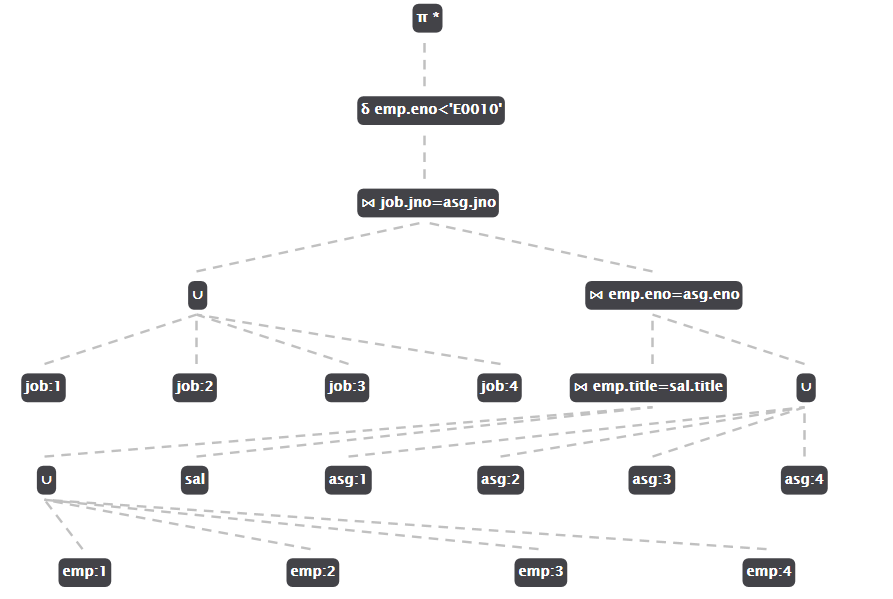


只生成了一个基本的查询树，效果还是可以的，分片如果显示条件的话太长展示不下

，所以这里用编号。对最后一个最复杂的查询，生成的表如下图：

这里输入的sql为：

select \* from emp,job,sal,asg where (emp.title=sal.title) and (emp.eno=asg.eno) and (job.jno=asg.jno) and (emp.eno<'E0010')



其实是正确的，这个展示主要是想把一行都画满，所以显示的比较怪异，之后再调整自适应的画法。

PS:sal表是没有分片的，分片方式参考benchmark里后面的水平+垂直划分