lab2

将一个字符串：SQL语句，解析为AST的流程如下：

- 词法分析：分析SQL语句中的每个单词，得到每个单词的token。其核心是利用正则关系式，来匹配和对应token相符的单词
- 语法分析：利用语法规则归约，构建语法树。其核心便是构造AST归约的语法规则

语法分析的本lab的核心便是构造JOIN语句的语法规则，根据SQL官网，JoinTable的语句为：

```
    table_reference [INNER | CROSS] JOIN table_factor
[join_specification]
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference
join_specification
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN
table_factor
```

根据上述规则就可以在parser.y文件写出JoinTable的语句，主要处理Join、Join on、Left Join on 这三种规则：

```
    TableRef CrossOpt TableRef %prec tableRefPriority
    {
        $$ = &ast.Join{Left: $1.(ast.ResultSetNode), Right: $3.
(ast.ResultSetNode), Tp: ast.CrossJoin}
    }
|   TableRef CrossOpt TableRef "ON" Expression %prec
tableRefPriority
    {
        condion := &ast.OnCondition{Expr: $5.(ast.ExprNode)}
        $$ = &ast.Join{Left: $1.(ast.ResultSetNode), Right: $3.
(ast.ResultSetNode), Tp: ast.CrossJoin, On: condion}
    }
|   TableRef JoinType CrossOpt TableRef "ON" Expression %prec
tableRefPriority
    {
        condion := &ast.OnCondition{Expr: $6.(ast.ExprNode)}
        $$ = &ast.Join{Left: $1.(ast.ResultSetNode), Right: $4.
(ast.ResultSetNode), Tp: $2.(ast.JoinType), On: condion}
    }
```

实验结果如下：

```
(base) [xiejian@xiejian-optiplex3070 parser]$ go test parser_test.go
ok      command-line-arguments  0.011s
```