大作业：实现 AQL Subset

编译原理

什么是AQL?

- 全称：Annotation Query Language
- 用于Text Analytics。可以从非结构化或半结构化的文本中提取结构化信息的语言。
- 语法类似于SQL

什么是AQL subset?

- AQL语法复杂，功能强大，实现难度较高
- AQL 子集具有AQL的主要特点

名词定义

- Token：以字母或者数字组成的无符号分隔的字符串，或者单纯的特殊符号，不包含空白符（blank）。
  - Eg: I am strong, and I don’t like eating light-blue lollipop.

I	am	strong	,
and	I	don	'
t	like	eating	light
-	blue	lollipop	.

名词定义

- Span：由正则表达式提取出来的具有位置信息的字符串，可以是完整的Token，也可以是不完整的（半个或者多个）。

- view：类似于数据库的Table，可以分为多个属性列，列中的行单元为Span。

AQL 子集

- 主要操作：
  - create view （创建view）
    - extract regex （利用正则从文本中提取span）
    - extract pattern （利用正则和view在原文本中匹配符合的模式）
    - select （在view中选列）
  - output view （格式化打印view）

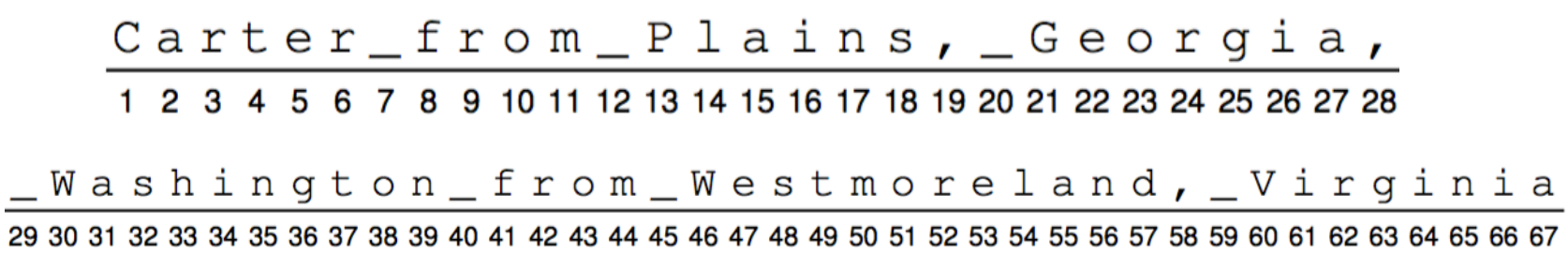
举例

AQL subset 示例

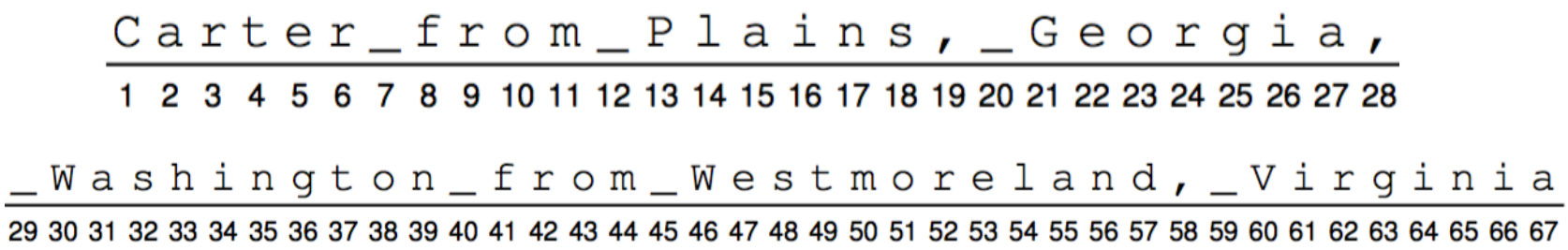
- **任务**：将人名和人名对应的地名从文本中提取出来
  - **人名**: 大写字母开头的Token
  - **地名**: 两个以大写字母开头的Tokens，并且中间以逗号分隔，其中第二个单词是美国的州名
  - 文本: （下划线代表空格，会被忽略）
    Carter_from_Plains,_Georgia,_Washington_from_Westmoreland,_Virginia

AQL subset 示例

- 输入：



General Method

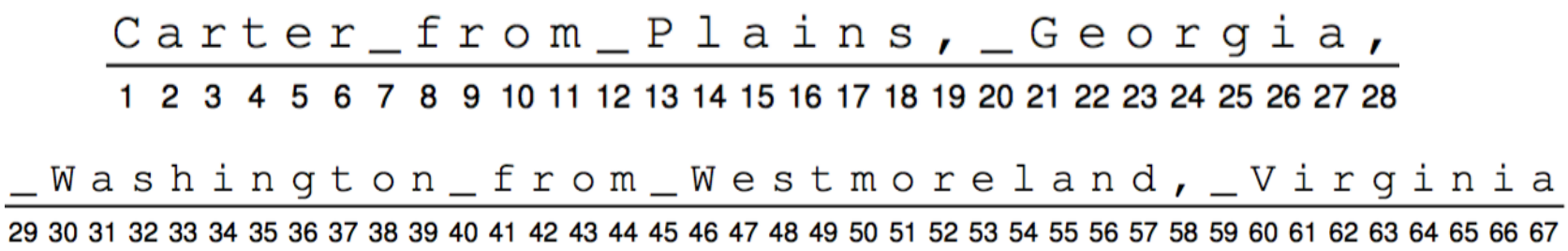


从文本中提取出大写字母开头的单词

Eg:

Carter(1,7)
Plains(13,19)
Georgia(21,28)
Washington(30,40)
Westmoreland(46,58)
Virginia(60,68)

General Method

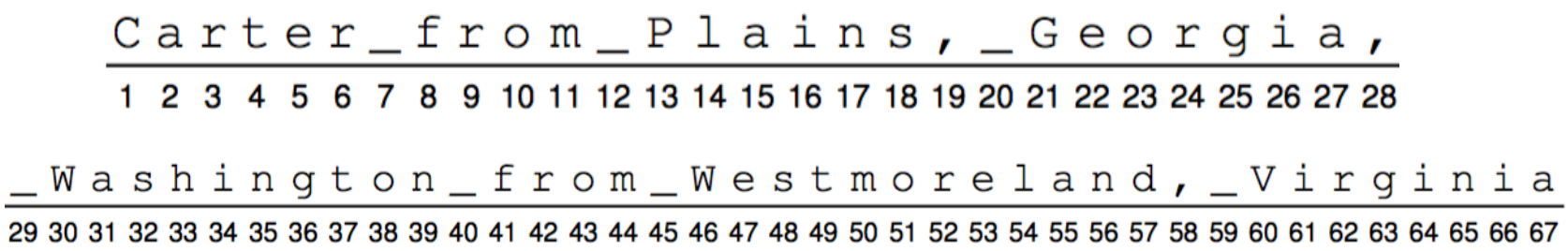


从文本中提取出美国的州名

Eg:

Georgia(21,28)
Washington(30,40)
Virginia(60,68)

General Method

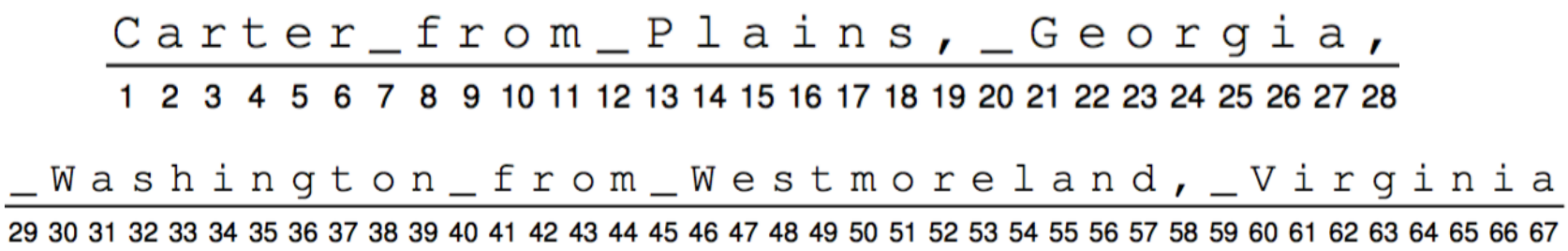


按照中间只**隔了一个逗号，且后一个单词为州名**即为一个地名的规则，
拼接上述两部操作得到的字符串列表

Eg:

Plains, Georgia(13,28)
Georgia, Washington(21,40)
Westmoreland, Virginia(46,68)

General Method



大写字母开头的为人名

Eg:

Carter(1,7)
Plains(13,19)
Georgia(21,28)
Washington(30,40)
Westmoreland(46,58)
Virginia(60,68)

General Method

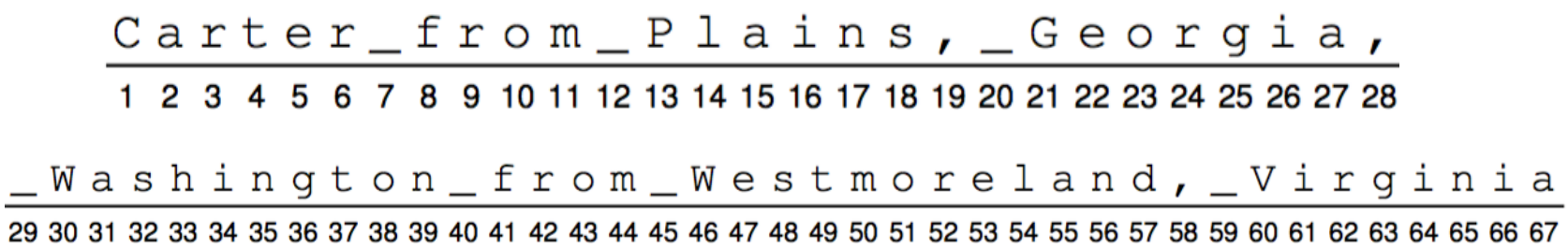
```
C a r t e r _ f r o m _ P l a i n s , _ G e o r g i a ,
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

_ W a s h i n g t o n _ f r o m _ W e s t m o r e l a n d , _ V i r g i n i a
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
```

人名与地名相对应的规则为相隔**1-2个tokens**,
根据此规则拼接人名和地名的列表
Eg:

人名	地名	人名-地名
Carter	Plains, Georgia	Carter from Plains, Georgia
Plains	Georgia, Washington	Plains , Georgia, Washington
Georgia	Westmoreland, Virginia	Georgia 无匹配
Washington		Washington from Westmoreland,Virginia
Westmoreland		Westmoreland 无匹配
Virginia		Virginia 无匹配

General Method



最终我们得到三个列表

Eg:

人名	地名	人名-地名
Carter	Plains, Georgia	Carter from Plains, Georgia
Plains	Georgia, Washington	Plains, Georgia, Washington
Georgia	Westmoreland, Virginia	Washington from Westmoreland,Virginia
Washington		
Westmoreland		
Virginia		

利用AQL

create view：regex语句

- 提取第一个字母大写的token

```
create view Cap as
  extract regex /[A-Z][a-z]*/
    on D.text as Cap
  from Document D;
```

```
View: Cap
+----------------------+
| Cap                  |
+----------------------+
| Carter:(0,6)         |
| Plains:(12,18)       |
| Georgia:(20,27)      |
| Washington:(29,39)   |
| Westmoreland:(45,57) |
| Virginia:(59,67)     |
+----------------------+
6 rows in set
```

注意：

1、view与数据库table类似，是可以具有多个属性列的表格。

2、view后的Cap是view的名字。

3、'/' 包括起来的是正则表达式，内部含有需转义。

4、Document是默认的view，指代当前输入文本。D是Document的别名。text是Document的一个属性列，存储的文本。

create view：regex语句

- 提取美国州名

```
create view Stt as
  extract regex /Washington|Georgia|Virginia/
    on D.text
    return group 0 as Stt
  from Document D;
```

```
View: Stt
+----------------------+
| Stt                  |
+----------------------+
| Georgia:(20,27)      |
| Washington:(29,39)   |
| Virginia:(59,67)     |
+----------------------+
3 rows in set
```

注意：

1、return group 给属性列命名。

2、group 0是匹配的全部，即整个正则所匹配的内容。

3、没有写return语句,只有as ID 默认为return group 0 as ID（如上一张）。

create view : pattern语句

```
create view Loc as
  extract pattern (<C.Cap>) /,/ (<S.Stt>)
  return group 0 as Loc
    and group 1 as Cap
    and group 2 as Stt
  from Cap C, Stt S;
```

```
View: Loc
+----------------------+--------------------------------+--------------------+
| Cap                  | Loc                            | Stt                |
+----------------------+--------------------------------+--------------------+
| Plains:(12,18)       | Plains, Georgia:(12,27)        | Georgia:(20,27)    |
| Georgia:(20,27)      | Georgia, Washington:(20,39)    | Washington:(29,39) |
| Westmoreland:(45,57) | Westmoreland, Virginia:(45,67) | Virginia:(59,67)   |
+----------------------+--------------------------------+--------------------+
3 rows in set
```

注意：Cap和Stt为之前两步建立的view。group1和group2为Loc的两个属性列。pattern的作用是利用正则和现成的span在文本中匹配符合的模式。

create view : regex语句

```
create view Per as
  extract regex /[A-Z][a-z]*/
    on D.text
    return group 0 as Per
  from Document D;
```

```
View: Per
+----------------------+
| Per                  |
+----------------------+
| Carter:(0,6)         |
| Plains:(12,18)       |
| Georgia:(20,27)      |
| Washington:(29,39)   |
| Westmoreland:(45,57) |
| Virginia:(59,67)     |
+----------------------+
6 rows in set
```

注意：

1、这里与前面的Cap一模一样。

2、再次强调：提取出来的每个字符串叫做span。span存储在view的属性列中。

create view：pattern语句

- 拼接人名与地名，要求只相隔1-2个Token

```
create view PerLoc as
  extract pattern (<P.Per>) <Token>{1,2} (<L.Loc>)
  return group 0 as PerLoc
    and group 1 as Per
    and group 2 as Loc
  from Per P, Loc L;
```

```
View: PerLoc
+--------------------------------+--------------------+-----------------------------------------------+
| Loc                            | Per                | PerLoc                                        |
+--------------------------------+--------------------+-----------------------------------------------+
| Plains, Georgia:(12,27)        | Carter:(0,6)       | Carter from Plains, Georgia:(0,27)            |
| Georgia, Washington:(20,39)    | Plains:(12,18)     | Plains, Georgia, Washington:(12,39)           |
| Westmoreland, Virginia:(45,67) | Washington:(29,39) | Washington from Westmoreland, Virginia:(29,67) |
+--------------------------------+--------------------+-----------------------------------------------+
3 rows in set
```

注意：

1、pattern中用圆括号包含的部分类似于「子表达式捕获」，group 0捕获整个pattern，group 1捕获第一个括号包含的部分作为属性列，group 2同理。

2、**Token** 是一个关键字，用来表达中间相隔了token。{min,max}表达重复。

create view：select语句

- 从现成的view中选列

```
create view PerLocOnly as
  select PL.PerLoc as PerLoc
  from PerLoc PL;
```

```
View: PerLocOnly
+-----------------------------------------------+
| PerLoc                                        |
+-----------------------------------------------+
| Carter from Plains, Georgia:(0,27)            |
| Plains, Georgia, Washington:(12,39)           |
| Washington from Westmoreland, Virginia:(29,67) |
+-----------------------------------------------+
3 rows in set
```

注意：

1、PerLoc这个view有三个列Loc、Per以及PerLoc。这里选择了PerLoc列。

create view：output语句

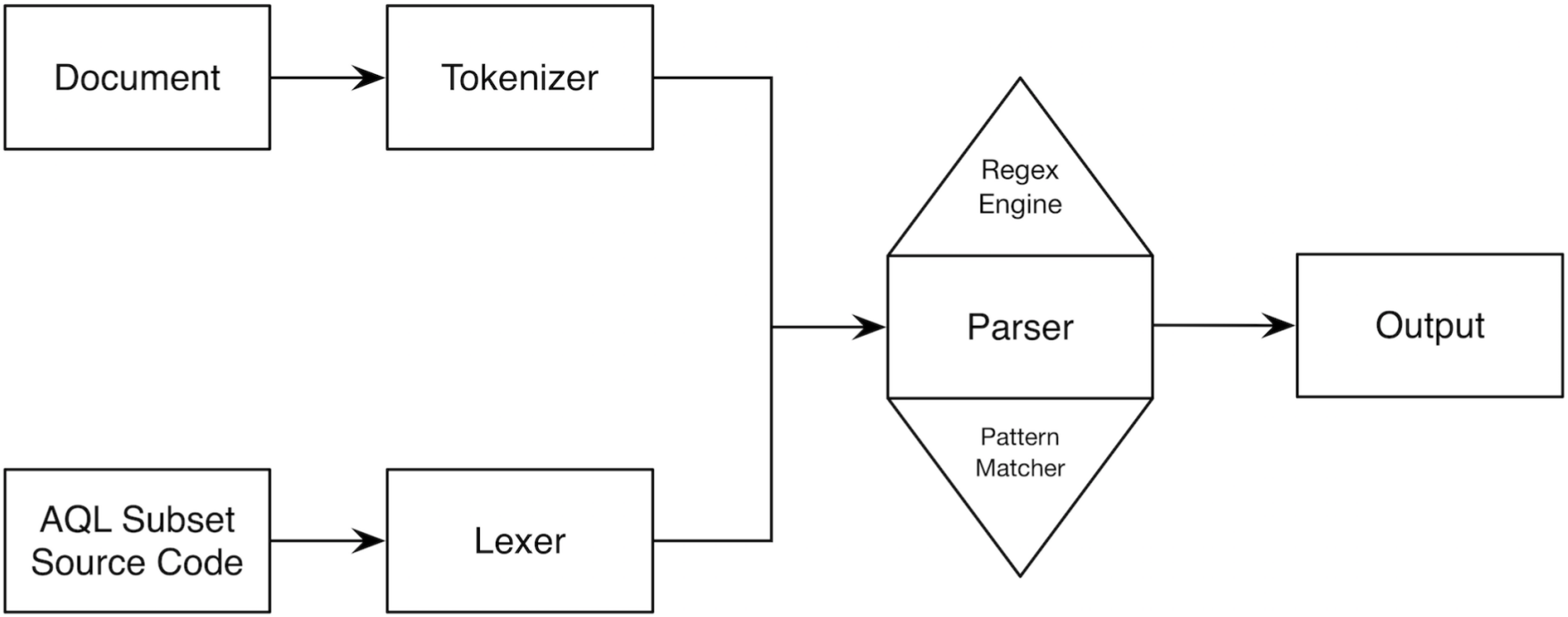
- 输出语句

```
output view Cap;
output view Stt;
output view Loc;
output view Per;
output view PerLoc;
output view PerLocOnly;
```

注意：

之前所有的语句的输出都是利用输出语句格式化打印出来的。

Procedure


Document
Tokenizer
AQL Subset
Source Code
Lexer
Regex
Engine
Parser
Pattern
Matcher
Output

语法

AQL Subset Grammar

aql_stmt → create_stmt **;** | output_stmt **;**

create_stmt → **create view ID as** view_stmt
view_stmt → select_stmt | extract_stmt

output_stmt → **output view ID** alias
alias → **as ID** | ε

Eg:

```
output view Cap;
output view Stt;
output view Loc  as L;
output view Per  as P;
```

AQL Subset Grammar

Select Statement

select_stmt → **select** select_list **from** from_list
select_list → select_item | select_list **,** select_item
select_item → **ID . ID** alias
from_list → from_item | from_list **,** from_item
from_item → **ID ID**

Eg:

```
create view PerLocOnly as
    select PL.PerLoc as PerLoc
    from PerLoc PL;
```

AQL Subset Grammar

Extract Statement—Regular Expression

extract_stmt → **extract** extract_spec **from** from_list
extract_spec → regex_spec | pattern_spec

regex_spec → **regex REG on** column name_spec
column → **ID . ID**
name_spec → **as ID** | **return** group_spec
group_spec → single_group | group_spec **and** single_group
single_group → **group NUM as ID**

Eg:

```
create view Per as
    extract  regex  /[A-Z][a-z]*/
        on D.text
        return group 0 as Per
    from Document D;
```

AQL Subset Grammar

Extract Statement—Sequence Pattern

pattern_spec → **pattern** pattern_expr name_spec
pattern_expr → pattern_pkg | pattern_expr pattern_pkg
pattern_pkg → atom | atom **{ NUM , NUM }** | pattern_group
atom→ **<** column **>** | **< Token >** | **REG**
pattern_group → **(** pattern_expr **)**

Eg:

```
create view PerLoc as
    extract pattern (<P.Per>) <Token>{1,2} (<L.Loc>)
        return group 0 as PerLoc
            and group 1 as Per
            and group 2 as Loc
    from Per P, Loc L;
```

关键字

关键字如下，大小写敏感。

create，view，as，output，select，from，extract，regex，on，return，group，and，Token，pattern

实现细节

Implementation

- Interpreter
  - Lexer
  - Parser

- Text Tokenizer
- Pattern Matcher
- Regular Expression Engine (已提供)

实现细节：词法分析器 lexer

- 建议AQL的每个token都记录行号、列号，方便进行词法错误的提示
- extract regex /正则表达/ ， 使用了 ’/’ 来包含正则表达式，可以在词法分析时忽略前后的’/’，将正则表达式整体作为一个token保存并返回给语法分析

实现细节：语法分析器 parser

- 递归下降的语法分析，禁止自底向上
- 在写代码时，语法的产生式可以自行调整，只要符合规定的语法规则
- 建议阅读《编译原理》课本的附录A

实现细节：正则表达式引擎

- 无需写，提供regex.cpp及接口函数findall。
- regex.cpp实现了NFA方式的正则引擎
- 支持特性：
  - 连接、选择、任意
    - ab, a|b, .
  - 重复(贪婪与非贪婪模式)
    - *, +, ?, *?, +?, ??
  - 转义
    - \r, \n, \t
  - 字符类
    - [a-zA-Z0-9_]
  - 捕获与非捕获
    - a(b)c, a(?:b)c
  - 全文多次匹配
    - [a-zA-Z0-9_]能够找到文本中所有的标识符，而不仅仅是匹配第一个找到的

实现细节：正则表达式引擎

- 正则文法

```
regex → alt
alt → concat | alt '|' concat
concat → repeat | concat repeat
repeat → single
          | single '*' | single '*' '?'
          | single '+' | single '+' '?'
          | single '?' | single '?' '?'
single → '(' alt ')'
          | '(' '?' ':' alt ')'
          | '.' | CHAR | CHARCLASS
```

实现细节：正则表达式引擎

- **正则函数接口：**
- vector<vector<int> > findall(const char *regex, const char *content)
- 函数功能
  - 返回正则表达式在文本中的所有匹配
- 函数参数
  - regex：　正则表达式
  - content：文本
- 返回结果
  - 返回正则表达式在文本中的所有匹配（**外层vector**）
  - 每一次匹配（**内层vector**）包含所有捕获块的文本匹配范围，其中每相邻两个整数表示一个捕获块范围(捕获块为**圆括号**包含起来的**子表达式**匹配到的内容)
  - 捕获块从0开始计数，第0个捕获块表示整个正则表达式，默认总是存在
  - 捕获块范围为左闭右开区间

实现细节：正则表达式引擎

```
extern vector<vector<int> > findall(const char *regex, const char *content);
int main()
{
    char regex[] = "([a-z]+)([^a-z]+)";
    char content[] = "ab2c1def3g12ui";
    vector<vector<int> > result;
    result = findall(regex, content);
    for (int i=0; i<result.size(); i++) {
        for (int j=result[i][0]; j<result[i][1]; j++)
            printf("%c", *(content+j));
        printf(": ");
        for (int j=0; j<result[0].size(); j+=2)
            printf("(%d,%d)", result[i][j], result[i][j+1]);
        printf("\n");
    }
    printf("\n");
}
```

输入：

正则：([a-z]+)([^a-z]+)

文本：ab2c1def3g12ui

输出：

ab2: (0,3)(0,2)(2,3)
c1: (3,5)(3,4)(4,5)
def3: (5,9)(5,8)(8,9)
g12: (9,12)(9,10)(10,12)

实现细节: Pattern

- 可以当做对若干表进行拼接。
- Eg：

  extract pattern (<C.Cap>) /,/ (<S.Stt>)

  这里 <C.Cap>是一个表中的列，<S.Stt>同理。

  /,/ 匹配Cap和Stt列中元素之间是否含有**逗号**。

实现细节: Pattern

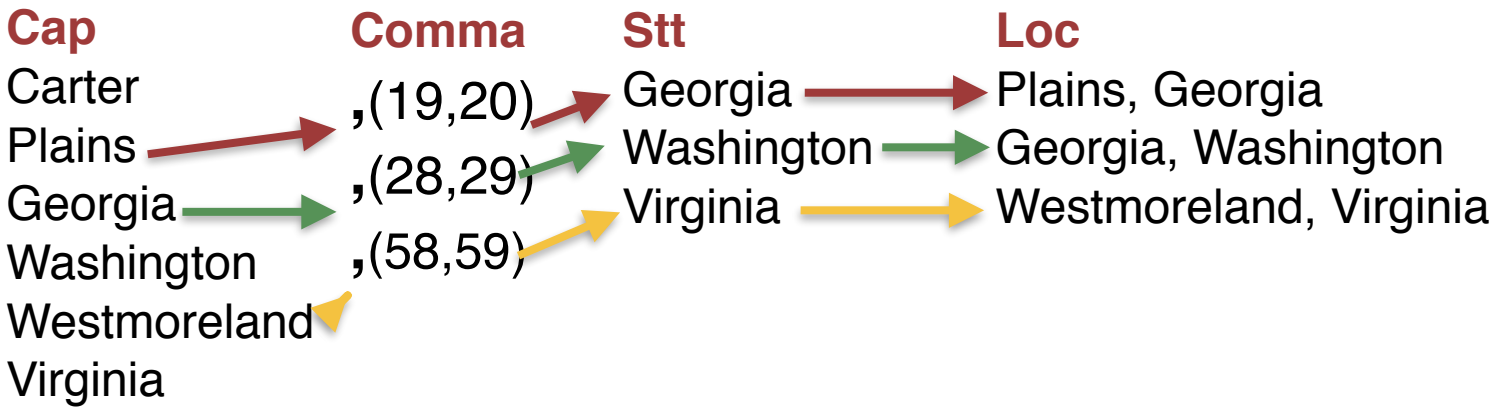
```
C a r t e r _ f r o m _ P l a i n s , _ G e o r g i a ,
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

_ W a s h i n g t o n _ f r o m _ W e s t m o r e l a n d , _ V i r g i n i a
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
```

按照中间只**隔了一个逗号，且后一个单词为州名**即为一个地名的规则。

pattern (<C.Cap>) /,/ (<S.Stt>)

Eg:



实现细节: Pattern

- 关于 <Token>{min, max}，这里min和max指 中间相隔了min到max个Token符合的模式。
- Eg：

  extract pattern (<P.Per>) <Token>{1,2} (<L.Loc>)

  指Per和Loc之间至少间隔1个token，至多间隔两个tokens。

实现细节：文本 Tokenizer

- 需要一个分词器，对文本划分token。
- 划分规则：以非数字与非单词为间隔，除了空白符的一切字符或字符串都可以作为token。空白符被忽略。
- [跳转]到幻灯片4查看例子

实现细节: output 格式化输出

- output的功能是将view的内容进行格式化输出
- (参考下图)

```
Processing sample1.input
View: Cap
+--------------------+
| Cap                |
+--------------------+
| Nick:(0,4)         |
| Atlanta:(10,17)    |
| Georgia:(19,26)    |
| Aaron:(31,36)      |
| Olympia:(42,49)    |
| Washington:(51,61) |
+--------------------+
6 rows in set

View: Stt
+--------------------+
| Stt                |
+--------------------+
| Georgia:(19,26)    |
| Washington:(51,61) |
+--------------------+
2 rows in set

View: Loc
+-----------------+----------------------------+--------------------+
| Cap             | Loc                        | Stt                |
+-----------------+----------------------------+--------------------+
| Atlanta:(10,17) | Atlanta, Georgia:(10,26)   | Georgia:(19,26)    |
| Olympia:(42,49) | Olympia, Washington:(42,61) | Washington:(51,61) |
+-----------------+----------------------------+--------------------+
2 rows in set
```

```
Processing sample2.input
View: Cap
+----------------------+
| Cap                  |
+----------------------+
| Carter:(0,6)         |
| Plains:(12,18)       |
| Georgia:(20,27)      |
| Washington:(29,39)   |
| Westmoreland:(45,57) |
| Virginia:(59,67)     |
+----------------------+
6 rows in set

View: Stt
+--------------------+
| Stt                |
+--------------------+
| Georgia:(20,27)    |
| Washington:(29,39) |
| Virginia:(59,67)   |
+--------------------+
3 rows in set
```

实现细节: output 格式化输出

- 格式:
  - 正在处理的文本，开头打印Processing sample.input
  - 打印view的名字，如果output带有别名，只打印view的别名
  - 以表格形式打印view的内容：列名和列的内容（注意对齐）
  - 表格后打印行数，若行数为0则打印「Empty set」
  - 列的内容包含span的实际字符和span的位置下标
  - 行、列的打印顺序无要求

实现细节：错误处理

- 可选做，无具体要求
  - 词法分析时，出现异常token，打印错误token的行号、列号以及错误的token和错误类型，并停止运行
  - 语法分析时，出现匹配异常，打印错误token的行号、列号以及错误的token和错误类型，并停止运行
  - （错误类型即词法错误或语法错误）

作业说明

作业说明

- 组队完成（人数<=5, 允许跨方向）
- 使用C/C++编写, 允许使用STL
- 代码干净整洁，变量命名合理
- 严禁抄袭

附加：数据集（必做）

- 任务：
  - 提供相应的AQL Subset的程序 *.aql ， 和对应数据集 *.txt，以及此程序对文章处理的输出结果 *.output
- 文本来源：国外的新闻站点、推特等
- 以一篇文章一个txt文本文件的形式存储

注意：

至少需提供一种类型的数据集，即适用于同一个 *.aql 程序的数据集。
同种类型的数据集需提供至少2篇文章。
(将会给出示例程序和数据集)

附加：阅读论文（必做）

- 论文： 『Spanners: A Formal Framework for Information Extraction 』
- 任务：
  - 阅读部分：Abstract、Introduction 和 Spanner

附加：中英文提取（选作，总分加5分）

- 问题1：本AQL Subset只适用于英文的信息提取。若加上中文则需对正则表达式引擎进行改造。
- 问题2：与单纯的提取英文内容不同，中文难以提取固定的有效信息。需要对 *.aql 的写法进行思考。

- 任务：1、提供有效的AQL Subset的实现
  2、提供相关数据集（一篇文章存为一个txt）
  3、提供数据集对应的 *.aql 程序
- 文本来源：国内微博、微信等

输入文件

- *.aql 为AQL程序文件
- *.input 为输入文本

PerLoc.aql

```
create view Cap as
  extract regex /[A-Z][a-z]*/
    on D.text as Cap
  from Document D;

create view Stt as
  extract regex /Washington|Georgia|Virginia/
    on D.text
    return group 0 as Stt
  from Document D;

create view Loc as
  extract pattern (<C.Cap>) /,/ (<S.Stt>)
  return group 0 as Loc
    and group 1 as Cap
    and group 2 as Stt
  from Cap C, Stt S;
```

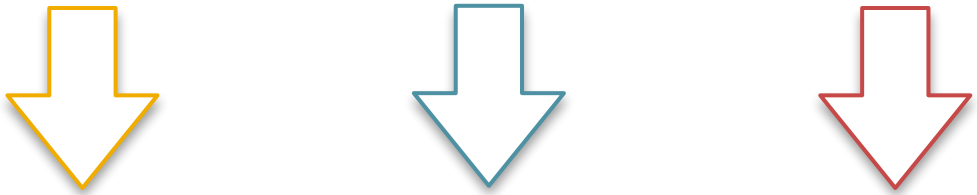
1.input

```
Carter from Plains, Georgia, Washington from Westmoreland, Virginia
```

执行过程

- ./AQL　sample.aql　sample.input　(单一文件)

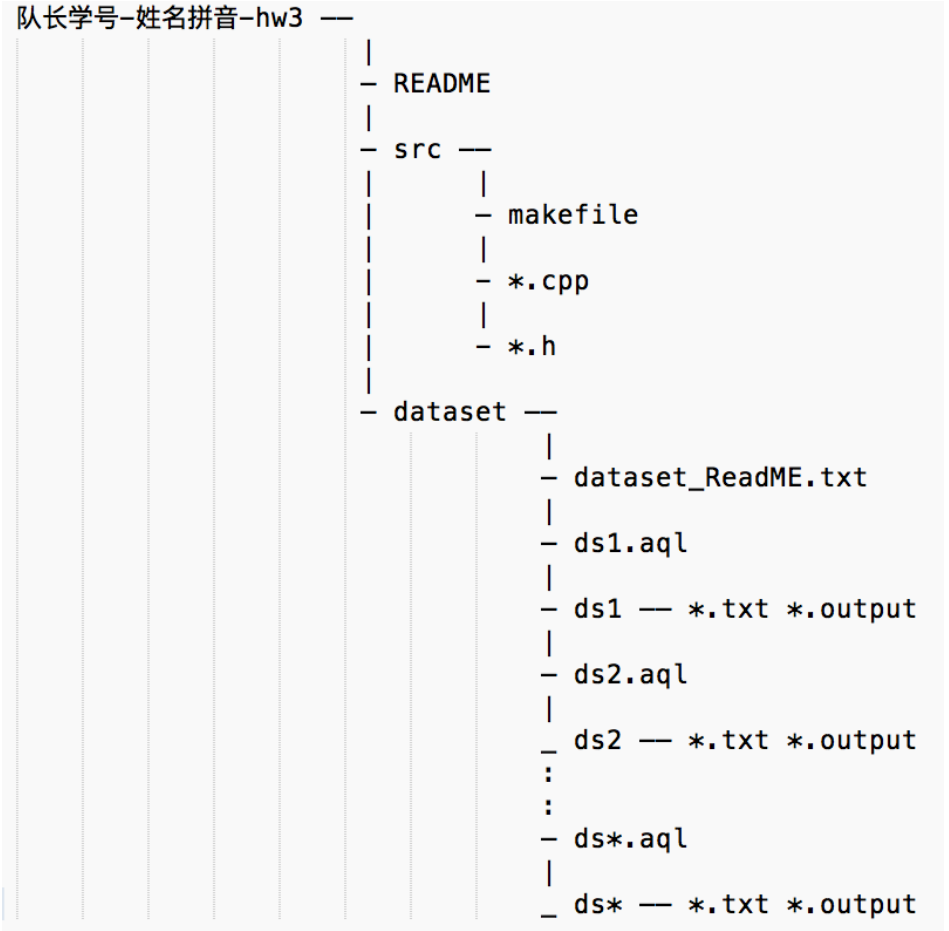
- 可执行文件　AQL程序　输入文本



- ./AQL　sample.aql　sampleTextDir（多个输入文件放入目录）

作业提交

- 提交文件：
  - AQL Subset的源文件
  - makefile
  - 说明文档：README
  - 数据集: dataset



作业提交

- 截止时间：2015年12月31日 晚上 24：00
- 提交方式：ftp://115.28.17.113，账号密码sdcs
- 命名规范: 队长学号-名字拼音-hw3.zip
  - 二次提交: 队长学号-名字拼音-hw3-v2.zip

References

- AQL Reference

  http://www-01.ibm.com/support/knowledgecenter/#!/SSPT3X_2.1.2/com.ibm.swg.im.infosphere.biginsights.aqlref.doc/doc/aql-overview.html