

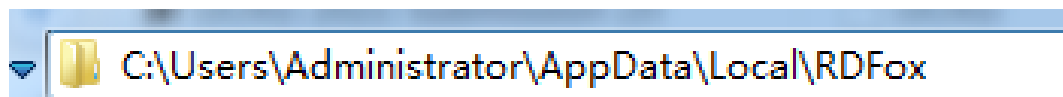
Knowledge Reasoning (II) - RDFox

一、项目导入

导入Licence文件

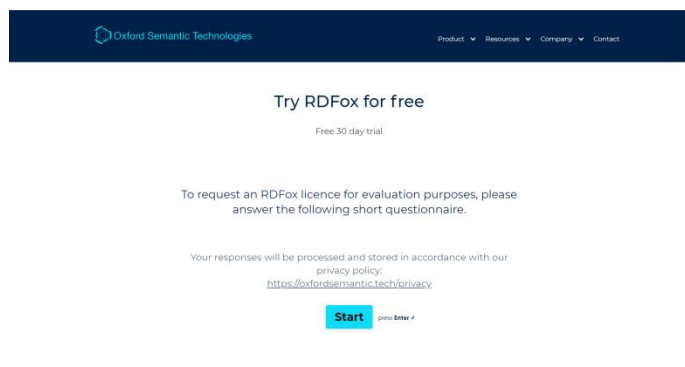
有Licence文件？

- Mac/linux: 将licence文件(RDFox.lic)放置入~/.RDFox/文件夹下
- Windows: 将licence文件(RDFox.lic)放置入
C:\Users\Administrator\AppData\Local\RDFox\文件夹下,
下图以Windows路径为例



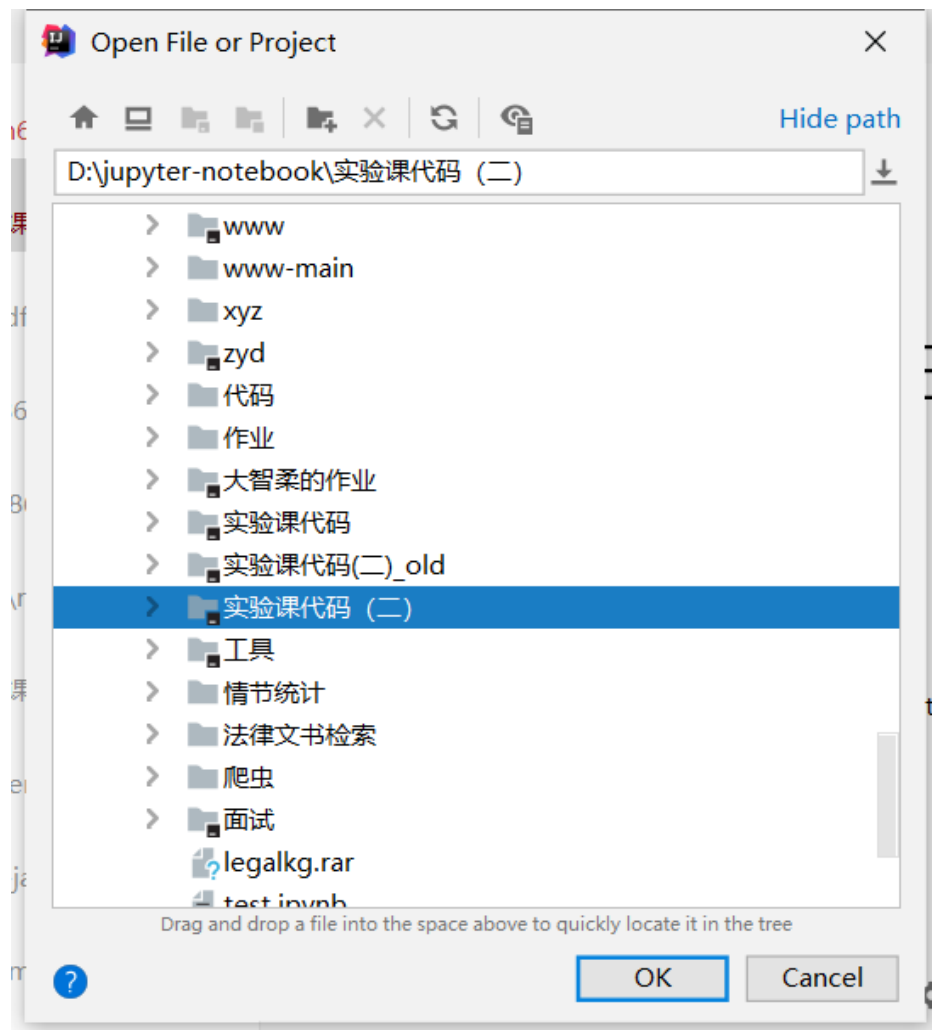
无Licence文件？

- 申请地址: <https://www.oxfordsemantic.tech/tryrdfoxforfree>
- 填写**组织邮箱** (如zhangsan@seu.edu.cn) 并申请通过, 可获得30天免费试用期



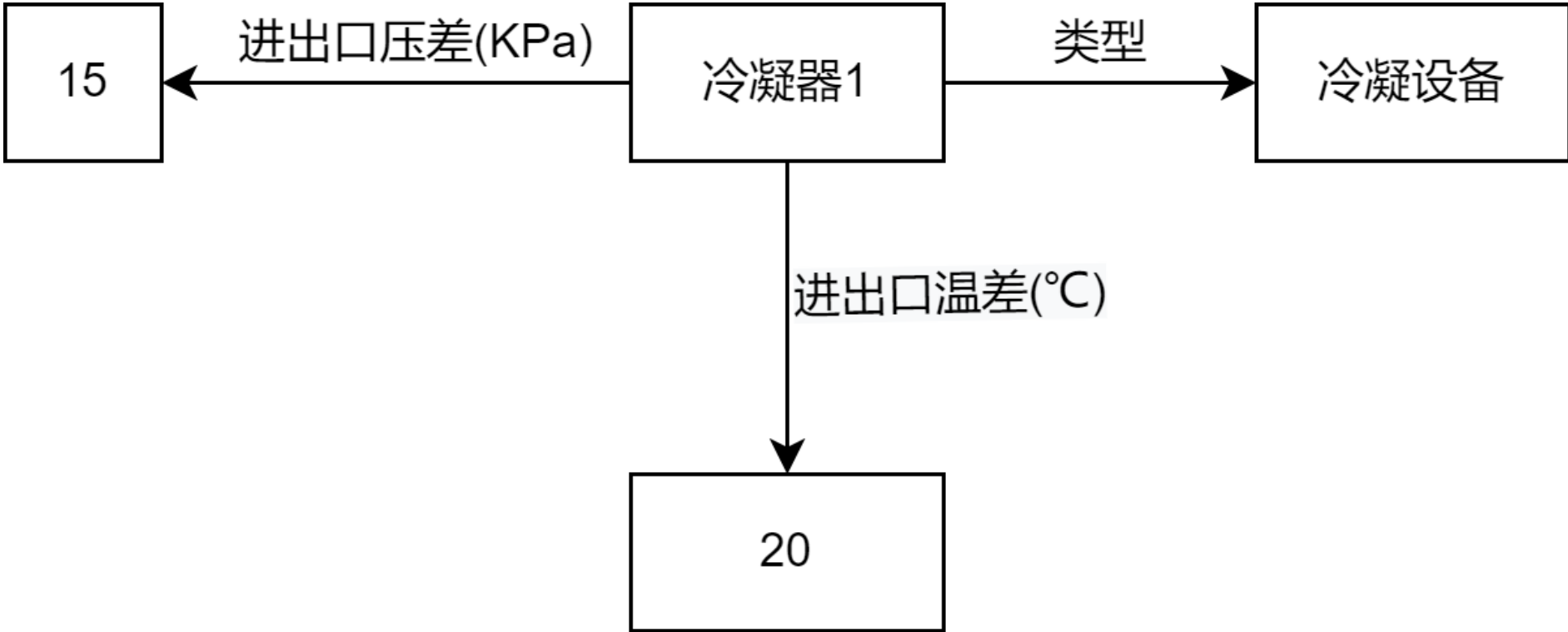
导入java项目

点击java项目“实验课代码（二）”，点击“OK”确认；



二、知识推理示例

示例一：故障诊断领域知识推理



RDFox实践-程序输入

规则：描述具体情况，以三元组的形式给出

对应：“实验课代码（二）\RDFox-win64-

3.1.1\examples\Java\tech\oxfordsemantic\jrdfox\data\diagnosis_rule.txt”

例如

- 已知华氏温度到摄氏温度转换公式，求进出口温差对应的摄氏温度。形式化规则为：
进出口温差($^{\circ}\text{F}$)[X,Y], 华氏度转摄氏度[Y,Z]->进出口温差($^{\circ}\text{C}$)[X,Z]
- 某冷凝设备进出口温差小于20摄氏度,该冷凝设备存在脏堵故障。形式化规则为：
类型[X, 冷凝设备], 进出口温差($^{\circ}\text{C}$)[X, Z], 小于[Z,20]->故障(X, 冷凝设备脏堵)

```
JRDFoxDemo_finance.java × diagnosis_rule.txt × diagnosis_data.nt × JRDFoxDemo_diagnosis.java × lubm1-new.ttl ×
1 PREFIX p: <http://www.example.org/kse/diagnosis#>
2
3 p:进出口温差( $^{\circ}\text{C}$ )[?X,?Z] :- p:进出口温差( $^{\circ}\text{F}$ )[?X,?Y], BIND ((?Y - 32) / 1.8 AS ?Z) .
4 p:故障[?X,p:冷凝设备脏堵] :- p:类型[?X,p:冷凝设备], p:进出口温差( $^{\circ}\text{C}$ )[?X,?Z], FILTER(?Z < 20) .
```

利用数据中不存在的三元组参与推理，实现数值转换、数值大小比较

RDFox实践-原子类型

基本概念：规则的主体公式可以是原子、否定等

- 形如（主语，谓词，宾语）的三元组称为元组表原子，又称通用原子；
- 形如`BIND(exp AS v)`的原子称为绑定原子，exp是一个表达式，表达式中的所有变量必须事先被声明，v是一个事先没有被声明的变量。例如华氏度转摄氏度[Y,Z]的实现方式为`BIND ((?Y - 32) / 1.8 AS ?Z)`；

RDFox实践-原子类型

- **过滤原子**的形式为`FILTER(exp)`，exp是一个表达式。表达式中的所有变量必须事先被声明。如果当前变量绑定exp值为true，则过滤器原子的求值成功。例如小于[Z,20]的实现方式为`FILTER(?Z < 20)`；
- 注意，绑定原子和过滤原子中的算术表达式对数据类型有限制，例如`BIND ((?Y - 32) / 1.8 AS ?Z)`中的变量Y和`FILTER(?Z < 20)`中的变量Z绑定的三元组宾语为以纯数字表示的数值类型。

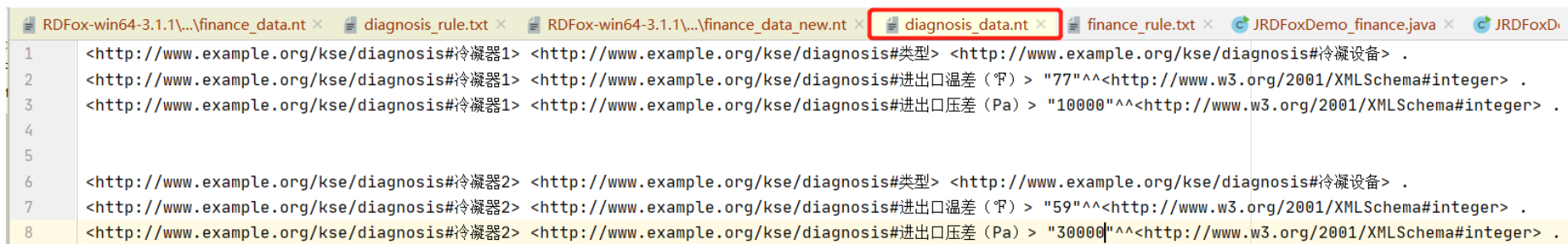
RDFox实践-程序输入

数据：记录事实信息，以三元组的形式给出；

对应“实验课代码（二）\RDFox-win64-

3.1.1\examples\Java\tech\oxfordsemantic\jrdfox\data\diagnosis_data.nt”；

前缀：<http://www.example.org/kse/diagnosis#> .

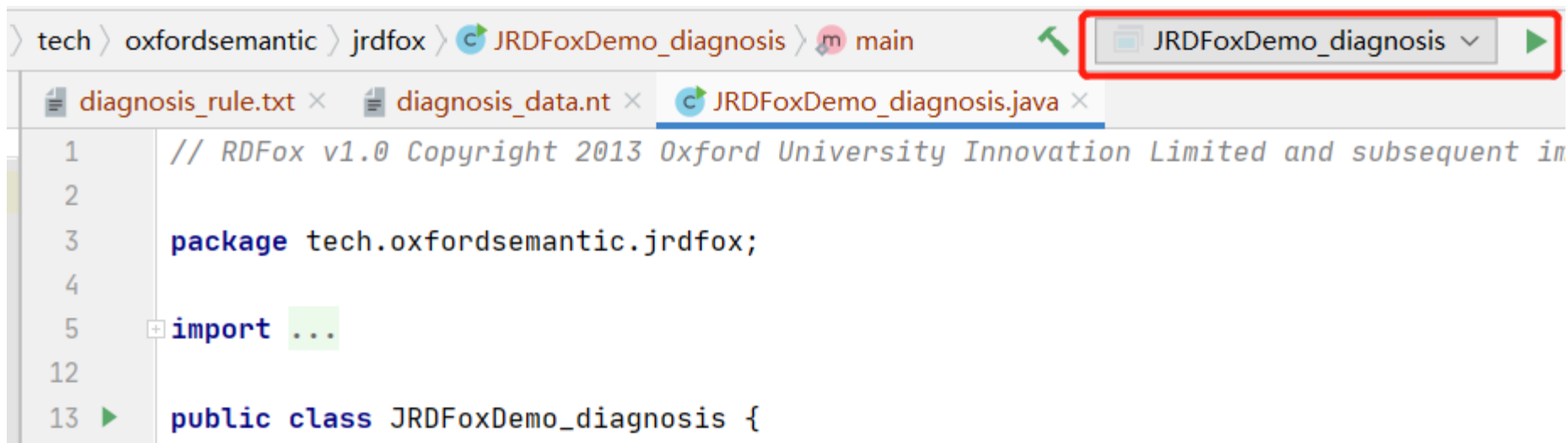


```
1 <http://www.example.org/kse/diagnosis#冷凝器1> <http://www.example.org/kse/diagnosis#类型> <http://www.example.org/kse/diagnosis#冷凝设备> .
2 <http://www.example.org/kse/diagnosis#冷凝器1> <http://www.example.org/kse/diagnosis#进出口温差 (°F)> "77"^^<http://www.w3.org/2001/XMLSchema#integer> .
3 <http://www.example.org/kse/diagnosis#冷凝器1> <http://www.example.org/kse/diagnosis#进出口压差 (Pa)> "10000"^^<http://www.w3.org/2001/XMLSchema#integer> .
4
5
6 <http://www.example.org/kse/diagnosis#冷凝器2> <http://www.example.org/kse/diagnosis#类型> <http://www.example.org/kse/diagnosis#冷凝设备> .
7 <http://www.example.org/kse/diagnosis#冷凝器2> <http://www.example.org/kse/diagnosis#进出口温差 (°F)> "59"^^<http://www.w3.org/2001/XMLSchema#integer> .
8 <http://www.example.org/kse/diagnosis#冷凝器2> <http://www.example.org/kse/diagnosis#进出口压差 (Pa)> "30000"^^<http://www.w3.org/2001/XMLSchema#integer> .
```

RDFox实践-执行程序

选择执行文件为JRDFoxDemo_diagnosis，对应路径为“实验课代码（二）\RDFox-win64-3.1.1\examples\Java\tech\oxfordsemantic\jrdf\JRDFoxDemo_diagnosis.java”

点击运行键执行代码



```
// RDFox v1.0 Copyright 2013 Oxford University Innovation Limited and subsequent in
package tech.oxfordsemantic.jrdf;
import ...
public class JRDFoxDemo_diagnosis {
```

RDFox实践-程序输出

- 推理结果显示推理出了新的三元组,推理之后的三元组被导出到.ttl文件;
- 规则被触发。

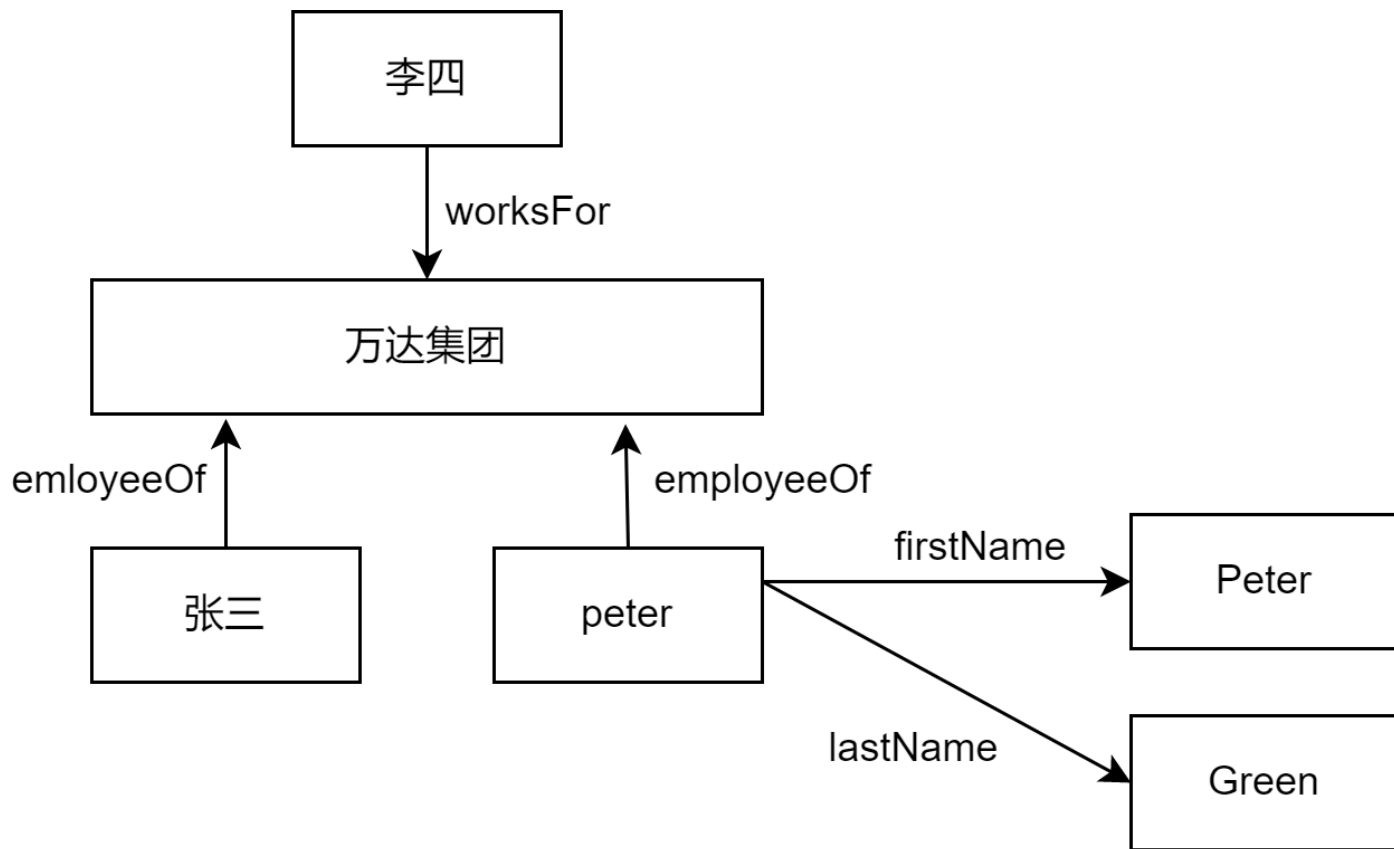
```
D:\JDK\bin\java.exe ...
Setting the number of threads...
Importing RDF data...
Number of tuples after import: 6
{ "head": { "vars": [ "Y" ] },
  "results": { "bindings": [
    { "Y": { "type": "uri", "value": "http://www.example.org/kse/diagnosis#类型" } },
    { "Y": { "type": "uri", "value": "http://www.example.org/kse/diagnosis#进出口温差 (°F)" } },
    { "Y": { "type": "uri", "value": "http://www.example.org/kse/diagnosis#进出口压差 (KPa)" } }
  ] }
}
Importing rules from a file...
Number of tuples after materialization: 9

=====
<http://www.example.org/kse/diagnosis#类型> * 1
<http://www.example.org/kse/diagnosis#进出口温差 (°F)> * 1
<http://www.example.org/kse/diagnosis#进出口压差 (KPa)> * 1
<http://www.example.org/kse/diagnosis#进出口温差 (°C)> * 1
<http://www.example.org/kse/diagnosis#故障> * 1
=====

The number of rows returned: 5
=====

Exporting facts to file '.\final-facts1983651441511867650.ttl' ... done.
This is the end of the example!
```

示例二：金融领域知识推理



RDFox实践-程序输入

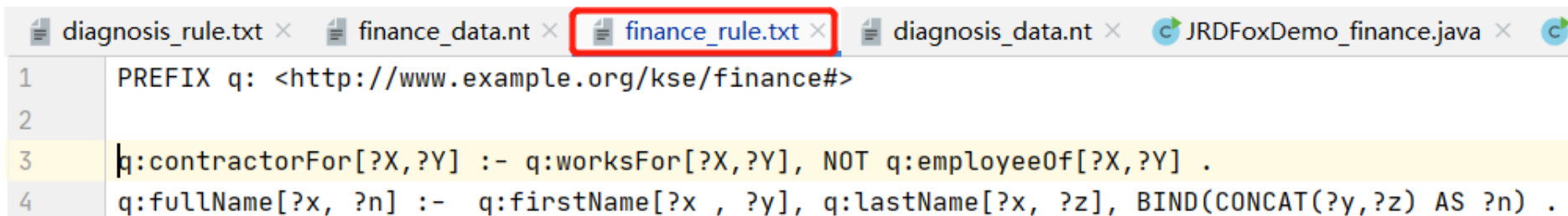
规则：描述具体情况，以三元组的形式给出；

对应“实验课代码（二）\RDFox-win64-

3.1.1\examples\Java\tech\oxfordsemantic\jrdfox\data\finance_rule.txt”。

例如

- 为一家公司工作但不是该公司雇员的人是外部供货商，形式化规则为
• **worksFor (X,Y), NOT employeeOf (X,Y) ->contractorFor (X,Y)**
- 某个人的全称是名字加姓氏,形式化规则为
firstName (X,Y), lastName(X,Z), 全名转换(连接(Y,Z), N) -> fullName(X,N)



```
diagnosis_rule.txt × finance_data.nt × finance_rule.txt × diagnosis_data.nt × JRDFoxDemo_finance.java ×
1 PREFIX q: <http://www.example.org/kse/finance#>
2
3 q:contractorFor[?X,?Y] :- q:worksFor[?X,?Y], NOT q:employeeOf[?X,?Y] .
4 q:fullName[?x, ?n] :- q:firstName[?x , ?y], q:lastName[?x, ?z], BIND(CONCAT(?y,?z) AS ?n) .
```

RDFox实践-程序输入

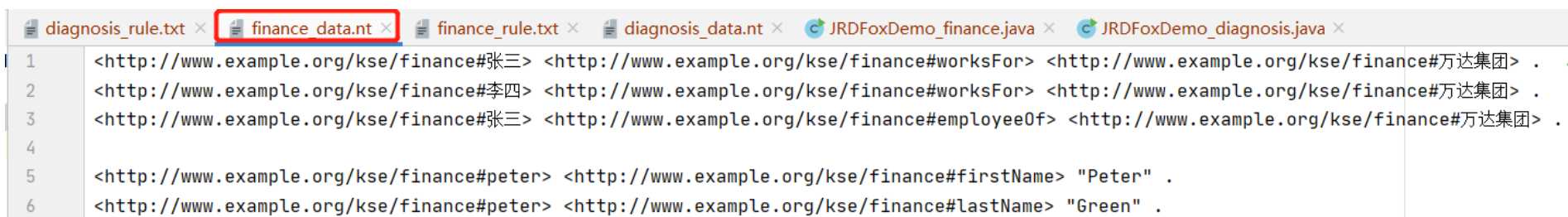
数据：记录事实信息，以三元组的形式给出；

对应“实验课代码（二）\RDFox-win64-

3.1.1\examples\Java\tech\oxfordsemantic\jrdfox\data\finance_data.nt”；

前缀：<http://www.example.org/kse/finance#>；

注意字符串的写法。



```
1 <http://www.example.org/kse/finance#张三> <http://www.example.org/kse/finance#worksFor> <http://www.example.org/kse/finance#万达集团> .
2 <http://www.example.org/kse/finance#李四> <http://www.example.org/kse/finance#worksFor> <http://www.example.org/kse/finance#万达集团> .
3 <http://www.example.org/kse/finance#张三> <http://www.example.org/kse/finance#employeeOf> <http://www.example.org/kse/finance#万达集团> .
4
5 <http://www.example.org/kse/finance#peter> <http://www.example.org/kse/finance#firstName> "Peter" .
6 <http://www.example.org/kse/finance#peter> <http://www.example.org/kse/finance#lastName> "Green" .
```

RDFox实践-否定失败

否定失败允许根据不存在的信息进行推论。否定通常意味着“缺乏信息”，而不代表三元组本身的错误。本质上是非单调的，这意味着新信息可能会使之前的否定失败无效，从而使推理出的三元组消失。

否定失败的使用方法

- 在要否定的三元组前加关键词“NOT”，当三元组存在时否定失败，这条规则不会被触发。例如“为一家公司工作但不是该公司雇员的人是外部供货商”的形式化规则为：

`worksFor (X,Y), NOT q:employeeOf (X,Y)->contractorFor (X,Y)`

RDFox实践-内置函数

Datalog 可以通过使用规则体中的内置函数进行扩展*。这里使用字符串连接函数**CONCAT**演示规则体中内置函数的使用。以下规则计算一个人的全名，将名字和姓氏进行串联。

规则：

```
q:fullName[?x, ?n] :- q:firstName[?x, ?y], q:lastName[?x, ?z],  
BIND(CONCAT(?y, ?z) AS ?n) .
```

数据：

```
<http://www.example.org/kse/finance#peter> <http://www.example.org/kse/finance#firstName> "Peter" .  
<http://www.example.org/kse/finance#peter> <http://www.example.org/kse/finance#lastName> "Green" .
```

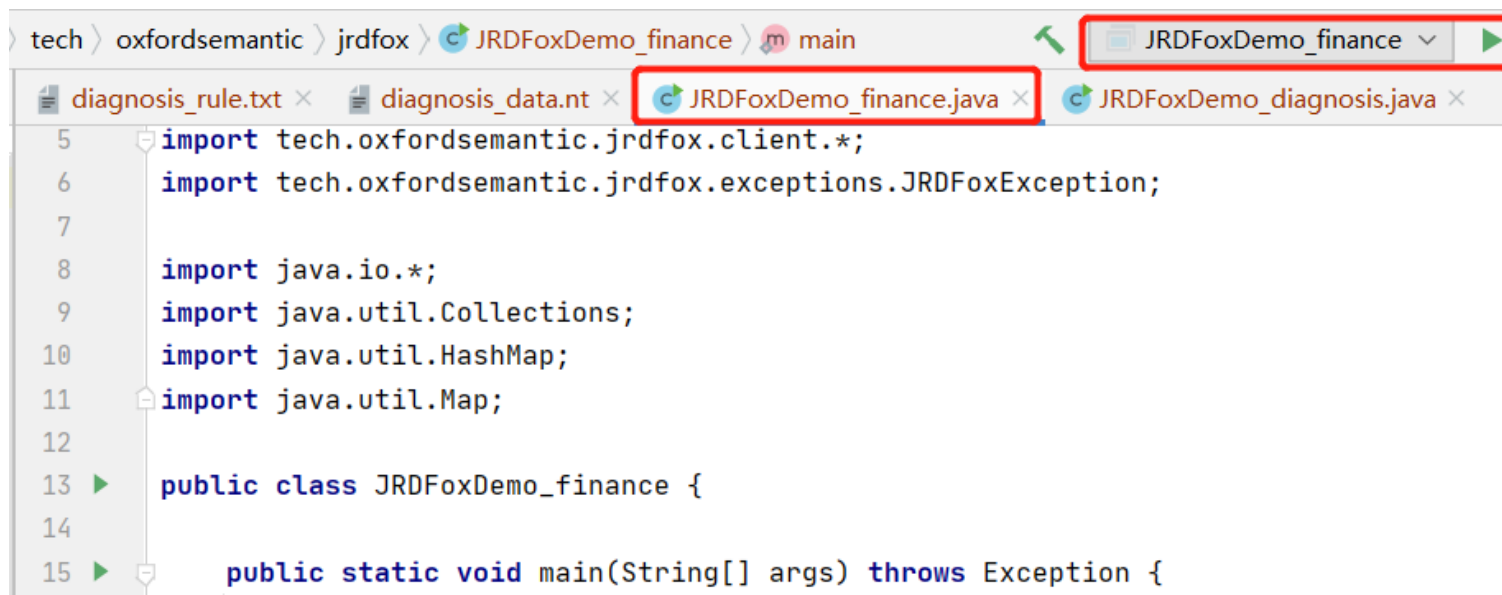
*：更多内置函数详见官方文档<https://docs.oxfordsemantic.tech/5.4/querying.html#built-in-functions>。

RDFOx实践-执行程序

选择执行文件为JRDFoxDemo_finance,对应路径为“实验课代码（二）\RDFOx-win64-

3.1.1\examples\Java\tech\oxfordsemantic\jrdfox\JRDFoxDemo_finance.java”

点击运行键执行代码



```
tech > oxfordsemantic > jrdfox > JRDFoxDemo_finance > main
diagnosis_rule.txt x diagnosis_data.nt x JRDFoxDemo_finance.java x JRDFoxDemo_diagnosis.java x
5 import tech.oxfordsemantic.jrdfox.client.*;
6 import tech.oxfordsemantic.jrdfox.exceptions.JRDFoxException;
7
8 import java.io.*;
9 import java.util.Collections;
10 import java.util.HashMap;
11 import java.util.Map;
12
13 public class JRDFoxDemo_finance {
14
15     public static void main(String[] args) throws Exception {
```

RDFox实践-程序输出

- 推理结果显示推理出了新的三元组,推理之后的三元组被导出到.ttl文件;
- 规则被触发。

```
D:\JDK\bin\java.exe ...
Setting the number of threads...
Importing RDF data...
Number of tuples after import: 5
{ "head": { "vars": [ "Y" ] },
  "results": { "bindings": [
    { "Y": { "type": "uri", "value": "http://www.example.org/kse/finance#worksFor" } },
    { "Y": { "type": "uri", "value": "http://www.example.org/kse/finance#employeeOf" } },
    { "Y": { "type": "uri", "value": "http://www.example.org/kse/finance#firstName" } },
    { "Y": { "type": "uri", "value": "http://www.example.org/kse/finance#lastName" } }
  ] }
}
Importing rules from a file...
Number of tuples after materialization: 7

=====
<http://www.example.org/kse/finance#worksFor> * 1
<http://www.example.org/kse/finance#employeeOf> * 1
<http://www.example.org/kse/finance#firstName> * 1
<http://www.example.org/kse/finance#lastName> * 1
<http://www.example.org/kse/finance#contractorFor> * 1
<http://www.example.org/kse/finance#fullName> * 1
-----

The number of rows returned: 6
=====

Exporting facts to file '.\final-facts466306021780804014.ttl' ... done.
This is the end of the example!
```

三、课堂总结

课堂总结

- 原子类型
 - 1) 通用原子
 - 2) 绑定原子
 - 3) 过滤原子
- 函数扩展
 - 1) 否定失败
 - 2) 内置函数

四、课堂作业

故障诊断领域知识推理

撰写规则，观察新的推理结果（对应“实验课代码（二）

\RDFox-win64-

3.1.1\examples\Java\tech\oxfordsemantic\jrdfox\data\diagnosis_rule.txt”）：

- 1) 已知Pa转换为Kpa的转换公式（ $1\text{KPa}=1000\text{Pa}$ ），求设备的进出口压差为多少Kpa？（对应diagnosis_data.nt中的谓词“进出口压差（Pa）”）
- 2) 某冷凝设备进出口压差大于 20KPa ，该冷凝设备存在“冷凝设备压差过大”故障。（对应diagnosis_data.nt中的谓词“进出口压差（KPa）”和“type”）

金融领域知识推理

1) 阅读程序源码，解除JRDFoxDemo_finance.java中141-146的注释，观察在推理过程中新插入三元组后推理结果的变化，理解否定失败非单调的性质。

参考资料

- RDFox官方文档
<https://docs.oxfordsemantic.tech/index.html>