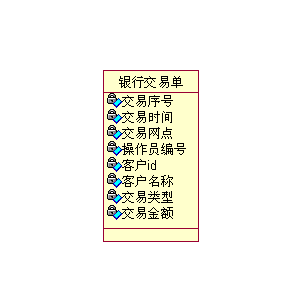
1. 背景介绍
   1. 大数据处理的挑战

XML 文件是一种具有自描述特性的文件，非常适合表达一些复杂的信息结构，所以也是一种非常重要的文件格式。 如何将XML文件转换为方便阅读的CSV格式，并从中提取需要的信息结构， 是非常重要的一种数据结构。

在实施大数据项目的时候，我们常常需要处理多种类型的XML文件，所以要想参与网优的各项工作，XML文件的处理是一个最基本的处理文件处理的技能。

* 1. 用例背景

在假设一个财富银行，他在一个城市有多个营业网点， 在一个营业网点中，有客户来存钱和取钱，每一次交易会生成一张ticket，其中包含了交易时间，流水号，操作员，客户，交易类型，交易金额的资讯。 交易完成以后，交易的ticket 会生成一个文件，放置在FTP接口上。



由于银行交易清单的格式会经常发生变化，所以我们会考虑使用xml格式来记录相关的银行交易单， 当这些银行交易清单，传递到中央银行以后，就需要能够将xml格式的清单，转换为CSV格式的清单，并用于后续的处理。

* 1. 用例描述
     1. 输入数据分析 （I）

在这里，我们假设银行的交易记录是XML格式的数据，相关的组成结构见下图所显示：

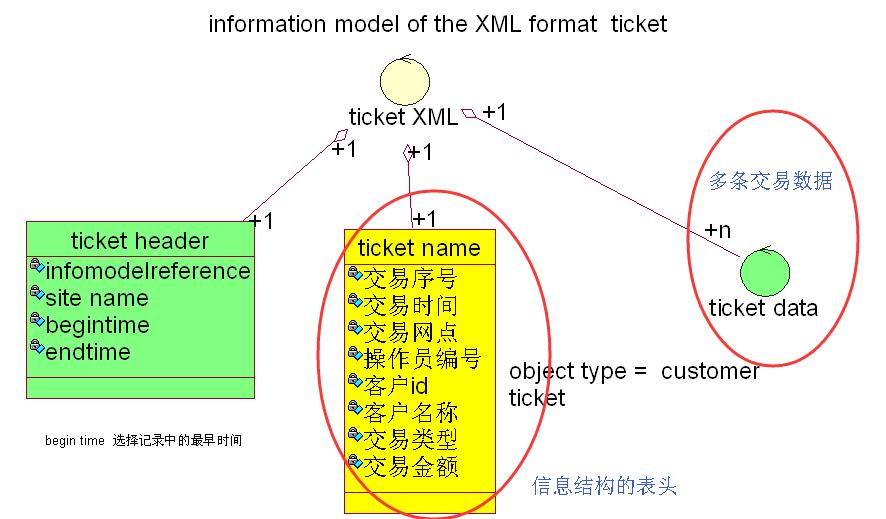


图 4-1 XML 格式的交易清单的组成结构

从上图可见， 相关的XML的数据包含以下的组成部分。

1， XML 的header 部分： 包含信息模型参考索引， 产生xml文件的营业网点名称， XML中包含的交易清单的开始时间和结束时间。

2，交易清单的表头部分（信息结构的名称）: 描述交易ticket的全部信息结构的名称，以list列表的方式来呈现（信息结构的名称在”value”部分进行表达）。

3，交易清单的数据部分（休息结构的取值）：描述交易ticket的全部信息结构的取值，以list列表的方式来呈现（信息结构的取值在“value”部分进行表达）。

根据上面的格式，我们可以提供三个xml 文件。

文件的命名为：

QLR2019020710-001.xml

QLR2019020710-002.xml

QLR2019020710-003.xml

。。。。。

关于输入数据的描述，参考以下文件。



* + 1. XML 解析处理过程（P）

1，用户界面向处理组件传递参数，入参包括：

**入参1**： 存放xml文件的文件路径，

**入参2**：存放解析后CSV的文件路径。

2，处理组件接收入参以后， 首先根据入参1：存放xml文件的文件路径，识别需要处理的xml文件，建立一个待解析的文件清单。

3、处理组件读取一个配置文件， ticket\_infofield.py 配置文件，确定需要提取的信息结构。

4，处理组件提取第一个文件，根据文件的后缀名称确定为xml格式，就调用xml处理组件，首先识别出xml文件的ticket\_name部分的信息结构取值，将相关的信息结构传入到一个list当中。

5，处理组件将ticket\_infofield.py 中的定义的需要提取的信息结构名字的列表（list），和当前xml文件中识别出来的ticket\_name部分的信息结构（list）进行比较，如果后者包含前者，这说明文件可以解析，转入后续的环节；否则，**则提示当前文件错误**。

6，处理组件根据**入参2**：存放解析后CSV的文件路径，打开一个CSV文件的句柄， CSV文件的文件名和xml文件相同；

并根据ticket\_infofield.py 中的定义的需要提取的信息结构名字的列表（list）顺序，写入CSV的表头数据。

7，处理组件接下来。从当前的xml文件中识别出ticket\_data部分的信息结构的取值， 根据并根据ticket\_infofield.py 中的定义的需要提取的信息结构名字的列表名字，匹配出需要提取的ticket\_data数据，处理完一条ticket交易记录，写入步骤6中打开的文件句柄中，写入一行数据；直到把所有的行记录处理完毕。

7，当前的文件处理完毕以后， 处理组件会打开一个结果list，记录本次的处理结果。包含6个信息结构，[process result, xml-file-size，input xml-path，output-csv-path，start time，end time],

其中，process result=0，处理成；=1，处理失败。

Xml-file-size 记录处理的xml文件的大小。

input xml-path： 输入xml文件的决对路径；

output-csv-path：输出csv文件的决对路径；

start time ：开始处理xml文件时间

end time ：生成csv文件的时间

8， 处理组件，遍历待处理文件清单，继续后续文件的处理，并生成返回值。

9， 全部文件处理完毕以后，整合所有文件的处理记录保存为一个CSV文件，其中包含7个字段。

[sequence id，process result, xml-file-size，input xml-path，output-csv-path，start time，end time],

其中sequence id 为自增序列。

CSV的文件名为 xml-handle-log-uuid.csv

存放路径，**入参2**：存放解析后CSV的文件路径。

10、输出本次用例执行的结果。包括 三个信息结构

[process result, number of xml handled，handle-log-path]

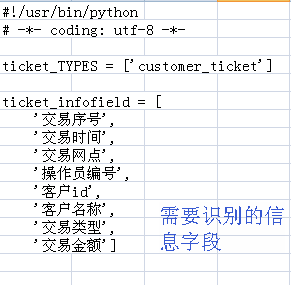
Process result：本次xml解析的结果，0 成功， 1，失败， 2，部分成功。

number of xml handled：本次解析的xml文件的数量。

handle-log-path：本次解析的日志文件的决对路径。（步骤9总生成的文件）

* + 1. ticket\_infofield.py

ticket\_infofield.py 是一个python的配置文件，用于配置需要提取的信息结构的字段。





* + 1. 返回数值

输出本次用例执行的结果。包括 三个信息结构

[process result, number of xml handled，handle-log-path]

Process result：本次xml解析的结果，0 成功， 1，失败， 2，部分成功

number of xml handled：本次解析的xml文件的数量。

handle-log-path：本次解析的日志文件的决对路径。（步骤9总生成的文件）

* 1. 设计约束

1、Xml解析使用组件

ElementTree

2、 Pthon使用python3

* 1. 设计考虑

针对以下用例需求。

9， 全部文件处理完毕以后，整合所有文件的处理记录保存为一个CSV文件，其中包含7个字段。

[sequence id，process result, xml-file-size，input xml-path，output-csv-path，start time，end time],

其中sequence id 为自增序列。

CSV的文件名为 xml-handle-log-uuid.csv

请思考： 在内存中用一种什么样的结构来暂存相关的信息？ 可以考虑python内建的组件，也可以考虑基于第三方的python包，或者第三方的独立组件。

实现方式不限制，但是请充分思考。