Code Instruction

依赖第三方库

- pebble: 一个带有 Timeout 超时设置的线程并行库
- sqlparse: sql query statement ⇒ ast
- sql_metadata: 对 sqlparse 的进一步封装，便于使用 ast
- beautifulsoup4: 使用其内部方法：UnicodeDammit 来检测 SQL script 的编码

项目结构

- data/
  - s3_sql_files_crawled_all_vms/ ⇒ 存放了所有解压后的 sql scripts 文本
  - s4_sql_files_parsed/ ⇒ 存放 parse 输出的 .pkl file
  - samples/ ⇒ 存放用于 sample 分析时的 .pkl file
  - variants_cases/ ⇒ 存放多种经由 grep 后匹配到的结果
- log/
- sql_parse/
  - cls_def.py
    - 定义了包括：Key, ForeignKey, Index, Column, Table, Pipeline 在内的多个基础类与各自的类方法。
  - display.py
    - 读入一个 .pkl 文件，统计并打印 pkl 中的主要内容
  - dump_tables.py
    - 读入一个 .pkl 文件，将内容输出为 Language Modeling 所需的 sequence 形式
  - exceptions.py (unused)
    - 自定义异常类
  - parse_query.py

        - join 与 non-join query 的解析类，模块下定义了 TableInstance, BinaryJoin, Query, QueryNode, QueryTree, TokenVisitor, QueryParser 类与各自的类方法。
        - 每个输入 query statement 将在主调方构造一个 QueryParser 对象，并调用该对象下的 parse 方法解析传入的 query statement
        - 对每个 query statement 构造出一棵 Query Tree，tree 中的每个 node 对应 query statement 中的每个 query scope，query node 内记录了指向其父节点(如果有)，子节点(如果有)的成员变量，对应 sub query 的 statement 以及 sub query ast。
        - 对 query tree 按：join condition ⇒ projection ⇒ aggregation ⇒ selection ⇒ groupby 进行解析并保存结果到 QueryParser 下相应的成员变量下。
        - 对于 join condition, projection, aggregation, selection 和 groupby 而言，五者中只需任意一个 parse 并 check 成功则认为该 query statement 解析成功，为之将构造出一个 query object 并保存之。
    - query_sample.py
        - 该模块下定义了用以分析计算 query parse 过程中 table fail, column fail, fk 与 join condition 重合以及同名 table 的重合比例等实用函数。
    - repo_parse_sql.py
        - 该模块定义了 Repository 类与对 repo 做聚合处理的相关函数，一般来说，该模块作为 shell 脚本调用 Python 的入口模块，在并行处理条件下，该模块作为 master thread 负责对 fork 出的每个 worker thread 进行调度。
    - s4_parse_sql.py
        - 在并行处理条件下，模块下定义的 作为每个 worker 的调用函数 parse_repo_files。
    - s4c_post_process_tables_for_training.py
        - 输入一个 pickle 文件，对数据做后处理
    - sample.py
        - 该模块下定义了一系列用于打印 table object, column object, etc. 对象内容的实用函数
    - utils.py

        - 定义了一份在抽取中用于管理正则表达式的自定义类 RegexDict，用于列类别映射转换的自定义类，此外还实现了一些 parse 过程的常用函数。

- tools/
    - grep_join_like_stmts.py: 计算所有 SQL scripts 中 join query 语句的数量

parse 功能介绍

```
通过向 run.sh 脚本传递不同的参数执行解析过程

OPTIONS
  -h, --help    show list of command-line options
  -t, --test    unit test for shell script functions
  -p, --parse   fork a SQL parse process
  -d, --debug   debug SQL parse scripts with pudb

SQL parse 分为两种模式，并行处理与串行调试模式。

通过 -p / --parse 选项 进入(并行)处理模式：
    开始解析后，将在 /datadrive/yang/exp/data/s4_sql_files_parsed
    下产生一个新的目录(以任务启动时的时间格式 yyyy_mm_dd_hr:mi:se 为结尾)，
    为避免解析过程中潜在的内存溢出问题，将会在该目录下分批保存解析完成的 .pkl 文件，
    注意：分批保存的文件以 _d 数字编号为文件名后缀；
    当解析全部完成后，程序会将所有 .pkl 文件合并存储到该目录下统一的 .pkl 文件中。

    P.S. 1) 该合并后的文件无 _d 数字编号文件名后缀；
         2) 如需解析全量数据，需提前取消 repo_parse_sql.py:321 处注释；
         3) 如需解析 1/100 全量的随机采样数据，需提前注释掉 repo_parse_sql.py:321，
            并且取消 repo_parse_sql.py:322 处注释。

通过 -d / --debug 进入(串行)调试模式：
    调试模式解析输出的说明同上
```

数据分析

数据集统计分析

- 对输出的 pkl file 整体进行分析：

```
通过 display.py 脚本完成对整个 parse 流程所输出的 pkl 文件的统计分析，可统计的数据有：
- 非空 repo 总数，
- 解析出的 table 总数，
- 非空 table 总数，
- 所有 table 的 column 总数，
- 所有表的主键的总数，
- 所有表的外键的总数，
- unique 约束的数目，
```

```
- candidate key 的数目，
- 拥有 data type 的 column 的数目，
- query 总数，
- binary join 总数，
- join condition 总数，
- index 总数，
- 包含 projection 的 query 的数目，
- 包含 aggregation 的 query 的数目，
- 包含 selection 的 query 的数目，
- 包含 groupby 的 query 的数目，
```

- 重名表分析方法：
  - 在 s4_parse_sql.py 中使用 print_name2tab 函数在跑一份 repo 数据的同时完成对重名表格的输出(并行 or 串行均可完成这一过程)。
- ForeignKey 与 BinaryJoin Condition 重合分析方法：
  - 在 query_sample.py 中借助 calc_fk_jq_overlap 函数，输入一份处理后的 repo_list (由 pkl file 读入)，函数执行后的打印结果即为二者重叠与否的分析结果。
- 同一 user 下在 repository 中寻找缺失 table 的比例：
  - 在 query_sample.py 中使用 calc_missing_table_in_other_repo 函数，输入一份处理后的 repo_list (由 pkl file 读入)，打印出 join query 中缺失的 table 在同用户下其它 repo 中的出现比例。
- 统计 table check fail 与 column check fail 数量：
  - 在 query_sample.py 中使用 calc_failed_cases_num 函数，输入一份处理后的 repo_list (由 pkl file 读入)，将分别打印出所有 check failed case, table check failed case 以及 column check failed case 的数量。
- 输出 pkl file 内容为序列化的 schema sequence:
  - 运行 dump_tables.py 脚本，将 pkl 文件包含的数据输出为 natural language sentence 形式的 table schema sequence。
- 检查 pkl 文件中保存的实体信息：
  - 打印 pkl 文件中各层级对象所存内容，可借助 sample.py 模块中 print_query_obj , print_table_obj , etc. 函数将对应层级的对象传入并打印出所保存的实体信息。

调试方法

> 本项目的调试借助 Python 3rd party → PuDB 完成

- 输入特定的 user 进行调试：
  - 在 `repo_parse_sql.py:288` 处取消注释，对属于同一 user 下的 repo 做聚合后在 `/datadrive/yang/exp` 目录下启动调试
- 输入特定的 repository 进行调试：
  - 在 `repo_parse_sql.py:291` 处更改需要调试的 repo url，打断点后在 `/datadrive/yang/exp` 目录下通过 `sh run.sh -d` 启动调试。
- 输入特定的 sql file 进行调试：
  - 在 `s4_parse_sql.py:1554` 处更改需要调试的 sql file 文件路径，打断点后在 `/datadrive/yang/exp` 目录下通过 `sh run.sh -d` 启动调试。
- 输入特定的 statement 进行调试：
  - 对于 query 而言，在 parse_query.py:2219 下插入需要 parse 的语句，打断点后在 `/datadrive/yang/exp` 目录下通过 `sh run.sh -d` 启动调试即可。