任务：基于已经生成好的schema case，生成符合语义要求的随机操作。

方法：将每一个SQL操作分解成若干个子部分，通过随机化的方式确定每一个子部分。

## CRUD操作

### select \*\*\* from \*\*\* where \*\*\* order by \*\*\* lock mode \*\*\*

* FromClause
  + 单表
  + 多表：只考虑基于外键的自然连接（基于视图的定义可以完成）
* Order by
  + 随机选择一些attribute作为排序的key
* ProjectRead
  + 随机选择一些attribute
  + 一定投影出主键，服务于trace分析
* WhereClause
  + 面向的属性列（可以是多列）
    - 基于主键
    - 基于外键
    - 基于非键值
    - 支持没有predicate，即全表扫描（这样子就可以支持AP了）
    - 三者混合，暂不考虑
  + 访问方式
    - 点 =
      * 每一个column都是等值查询，用and连接
    - 范围 between and
      * 每一个column都是范围查询，用and连接
    - 复杂
      * 每一个column随机选择predicate，用and or将多个column进行连接
  + predicate
    - =
    - >
    - <
    - >=
    - <=
    - !=
    - Like
    - In
    - Between and
    - Is null
    - Not
    - **And**
    - ***Or***
  + 生成的时候，还要考虑有些参数的类型和operator之间的兼容性
  + 要提供参数设置的方法和刷新参数的方法，先别管参数的问题
* Lock mode
  + for update
  + lock in shared mode
* *subquery*
* *having*
* *group by*
* *aggregate*

### update \*\*\* set \*\*\* where \*\*\*

* FromClause
* ProjectWrite
  + 要提供参数设置的方法和刷新参数的方法
* WhereClause
  + 仅考虑基于主键的点写即可
  + 要提供参数设置的方法和刷新参数的方法

### delete from \*\*\* where \*\*\*

* FromClause
* WhereClause
  + 仅考虑基于主键的点写即可
  + 要提供参数设置的方法和刷新参数的方法

### insert into \*\*\* values(\*\*\*)

* FromClause
  + 单表
* WhereClause
  + 仅考虑基于主键的点写即可
* ProjectWrite
  + 要提供参数设置的方法和刷新参数的方法

### 每种操作SQL格式输出

包括SQL模板输出和实例化之后的SQL输出

### 每种操作的Trace记录方式

每一个操作都要记录Trace信息，格式如下：（要快速产生和写入，还要方便读取和分析，trace是负载生成和异常分析的媒介）

* transaction ID (tid)
* operation ID (oid)
* operation type(ot)
  + 读，写，提交，回滚，开始
* start timestamp (st)
  + 尽可能细化这个时间
* end timestamp (et)
  + 尽可能细化这个时间，越精细越有利于我们异常检测，jdbc能否精细到逐行记录的时间戳？我们的方法也不是一个精准测试，可以用（et-st）这一指标作为异常分析的误差
* Where clause(pr)
* table and pk of operation object (ooi)
  + 即Key，可能有多个
* operation data (od)
  + 即Value，数目等于Key的数目

### trace写缓冲系统

是实现二进制日志，还是实现文本日志？？

Trace的读写缓冲最好自己实现，不利用操作系统的缓冲，因为trace的领域知识只有我们自己知道，自己做缓冲可以加快trace的读写速度。

使用O\_DIRECT方式写文件，跳过操作系统的缓存，自行维护trace的写缓冲。在内存中维护一个缓冲区trace buffer，用于缓冲trace的写入，trace buffer是一个队列，worker threads产生trace放入trace buffer，trace threads负责刷新trace buffer磁盘中，所以trace的刷新相当于一个生产者和消费者问题，可以用lock-free的同步队列实现trace buffer。这里实现了一个异步IO。

读缓冲系统类似的实现原理，异步读取，预先读取，充分利用多核硬件。

## 常见的操作组合

### Read-Modify-Write

基于一行记录先读后写。主键点读，写可以是update,delete

尽管我们没有基于外键和非键值的写，但是可以通过封装read-modify-write实现基于外键和非键值的写，即先基于外键和非键值进行读，对读到的主键数据，我们基于主键进行写。

也就是说，我们将基于外键和非键值的写拆分成先读后写。（TPPC中的写操作大部分都是这样的，Update都是基于主键的写）

### Read-Twice

同一个的读操作执行两次，中间存在sleep time。

### Update-Reference

基于参考表的主键更新一行记录，该参考表的主键前缀存在外键，那么可以基于这个外键去更新被参考表的一行记录。（最终结果是两个更新操作都是基于主键的点更新）

### Read-your-write

写了一行数据之后，读取这行数据。写可以是Update，Insert，Delete。

### 做出接口方便扩展更多的操作

## *增删索引的操作*

### Alter \*\*\* add index \*\*\*

### Alter \*\*\* drop index \*\*\*

## 将基本操作随机组合成事务模板

先生成一批随机操作，将这些操作SQL格式化输出，然后基于这些操作随机生成一批事务模板，再将这些事务模板格式化输出。

## 编码Tips

* 面向对象编程，要学会把共性的代码抽象出来，降低冗余和耦合
  + 这一块就是基于schema数据结构进行生成，可以回过头去调整schema generation的代码
* 如何设计随机生成的流程和seed
* 要留下方便扩展的接口