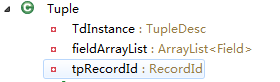
SimpleDB 实验报告

**Tuple class :**

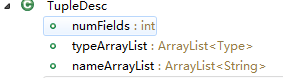
****

此类定义了三个私有变量，TdInstance 变量定义了元组的字段属性，

filedArrayList定义了各字段的value值，tpRecordId定义了元组在那个page和

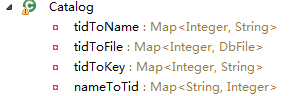
在page里的偏移量。

**TupleDesc class:**



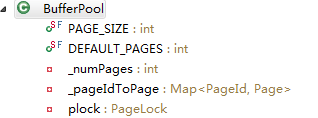
此类定义了三个公有变量，numFields变量定义的字段属性的个数，typeArrayList保存字段属性的类型，nameArrayList保存字段属性的名称。

**Catalog class :**



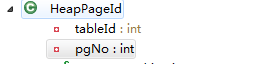
此类定义了四个私有变量，tidToName保存了从表的ID到表名称的映射，tidToFile保存了从表的ID到表文件对象的映射，tidToKey保存了从表的ID到表主关键字的映射，nameToTid保存了从表的名字到表ID的映射。

**BufferPool class ：**

****

PAGE\_SIZE 常量定义了page的大小，DERAULT\_PAGES定义了默认缓冲区页面数量，\_numPages保存缓冲区当时缓存了多少页面，\_pageIdToPage 保存了从PageId到page的映射关系，plock保存了所有缓存页面的锁状态。

**HeapPageId class ：**

****

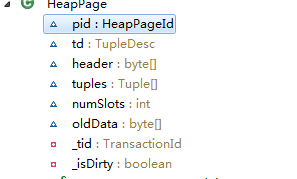
tableId保存属于哪个表的ID，pgNo记录此页在表文件中位置。

**RecordId class ：**

****

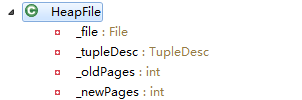
pid记录页的ID，tupleno记录元组在一页中的位置。

**HeapPage class ：**

****

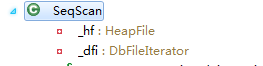
pid保存页面的唯一标示符ID，td 保存页面元组的字段属性，header数组标识某位置是否放置元组，tuples元组放置元组，numSlots记录可以放置元组总量，oldData数组保存页面的旧数据，\_tid 保存操作此页面的事物ID，\_isDirty标识此页面是否被写过。

**HeapFile class ：**

****

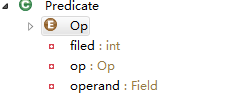
\_file记录此堆文件所对于的实际磁盘文件，\_tupleDesc记录此文件的元组的字段属性，\_oldPage 记录此文件中操作前有多少页面，\_newPages记录操作过程中产生的新页面数量。

**SeqScan class ：**

****

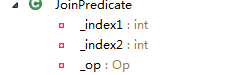
**\_**hf 记录要需要进行元组扫描文件，\_dfi是文件的迭代器。

**Predicate class ：**

****

field保存了元组哪个字段参与比较，op指明比较方式，operand保存被比较值。

**JoinPredicate class :**

****

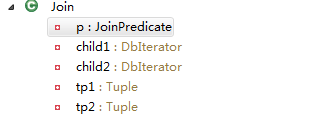
\_index1 定义了第一个元组哪个字段参与比较，\_index2定义了第二个元组哪个字段参与比较，\_op定义了比较方式。

**Filter class :**

****

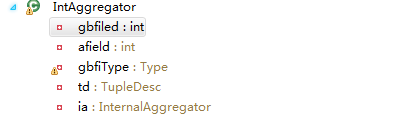
P定义了比较方式，child是一个元组迭代器。

**Join class ：**

****

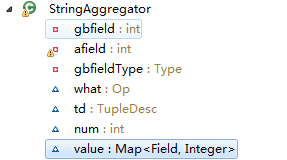
P保存连接的条件，child1第一个元组的迭代器，child2第二个元组迭代器，tp1临时保存child1迭代器中的一个元组，tp2临时保存child2迭代器中的一个元组。

**IntAggregator class ：**

****

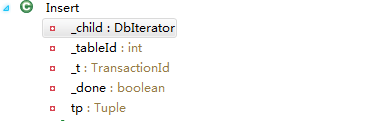
Gbfiled 指示了以哪一个字段进行分组操作，afield指示了以哪一个字段进行聚合操作，td保存了元组的字段属性，InternalAggregator保存了聚集操作的具体类对象。

**StringAggregator class：**

****

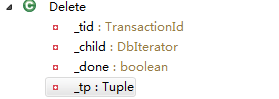
同上一个类同样不再赘述。

**Insert class :**

****

\_child插入元组的迭代器，\_tableId插入元组的表ID，\_t标识插入事物的ID，\_done 标识事物是否成功完成，\_tp保存临时元组。

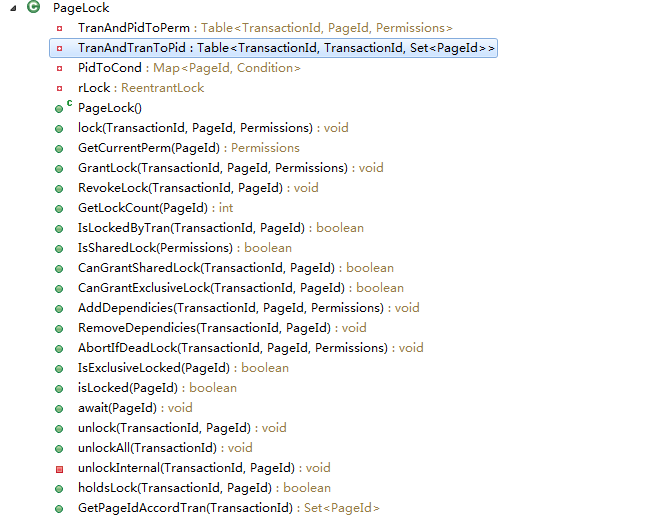
**Delete class ：**

****

同上一个类同样不再赘述。

以上类根据实验文档和已有的框架逐步填写，并没有详细讲解各类的函数功能，只是讲解了各类中定义的变量及含义。接下来详细讲解一个pagelock类及相关的功能，及到此为止SimpleDB所能实现的功能。

**PageLock class ：**

****

此类利用了google开发的一个包com.google.common.collect.\*;此包中有一种很强的数据类型table，是对map数据结构的一种强化，table<R,C,V>,参数介绍如下：

Parameters:

<R> the type of the table row keys

<C> the type of the table column keys

<V> the type of the mapped values

可以从table中按R或C导出C到V的map后R到V的map。

TranAndPidToPerm变量保存事物对某个页所拥有的访问权限，TranAndTranToPid变量保存事物对某个事物在一个页面集合有依赖关系。

PidToCond 保存了页面所对应锁的状态。rLock就是一把对所有页面的锁。

根据如上一些函数可以看出此类实现如下功能：

* 获取某个页面的锁，如果堵塞则等待
* 释放某个页面的锁，同时通过condition来激活其他堵塞的进程
* 释放某个事物的锁，同时通过condition来激活其他堵塞的进程
* 可以通过事物之间的依赖关系来判断是否发生死锁，如果发生就直接释放该事物的所有锁

**总结 ： 到此为止，SimpleDB实现了基于只支持两种类型的简单数据库，支持的操作有，新建表，对表数据的插入和删除，同时还能支持聚合，和简单查找操作。**