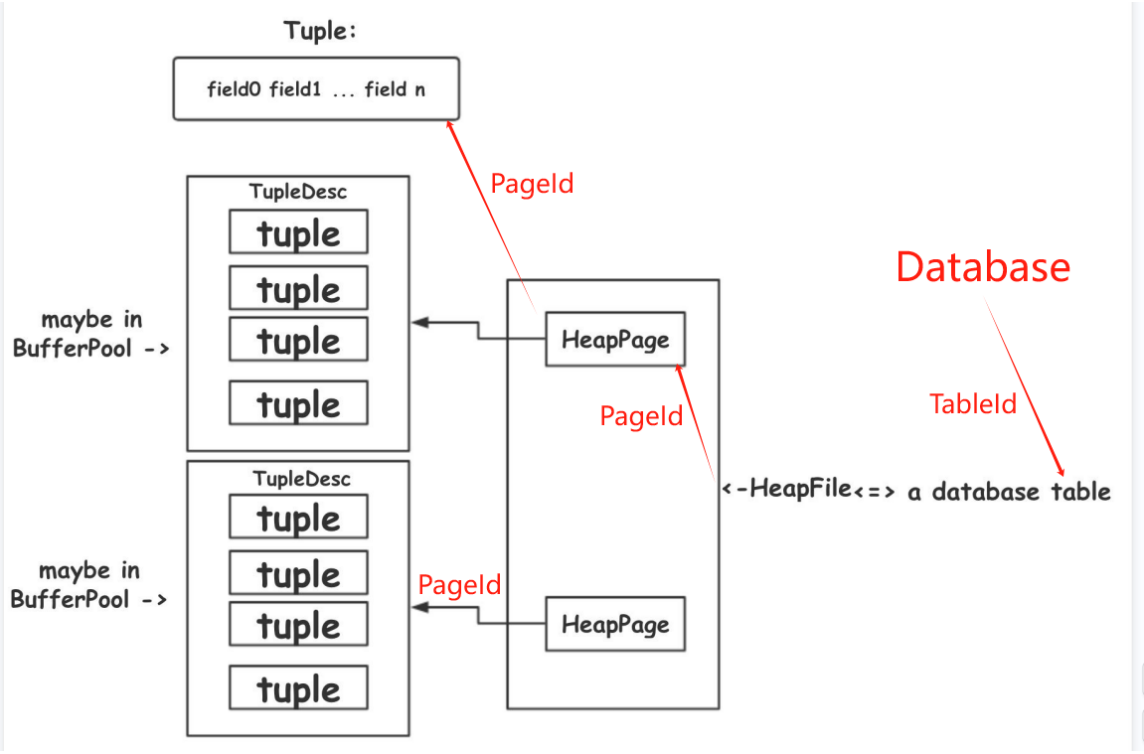
MIT leb1

- **疑问：**

  - **1.**如果是，那么Table太大是分为多个File吗？Table太小是否占用一个File?
  - File暂时没有设置最大字节量，数据有多少File有多大。但是当从File读取数据时，不满足一个Page大小会返回一个Page大小。HeapFile的readPage(PageId pid)方法
  - **2.**代码哪里设置看先从Bufferpool里面读取数据?
  - lec5的Page的迭代器上设置了
  - **3.**再BufferPool中可以根据PageId查找对应数据，在磁盘的数据需要File路径，怎么知道的?
  - File类有**绝对路径**
  - **4.**lec4中HeapPage类的getPageData() 方法，将初始化到该HeapPage的数据序列化到磁盘(利用输出流)，后面怎么找到？根据File类的**绝对路径+页号**作为**PageId**去BufferPool里对应，但是发过来根据PageId能得到File类的**绝对路径吗?**
  - **5.**DbFile类有唯一路径作为ID，File与Table一一对应，那么TbaleId与File对应吗?
  - 是的，在一个Database维护一个*Catalog*，*Catalog的表的TableId唯一对应一个File.*
  - *DbFile == Database.getCatalog().getDatabaseFile(TableId);*

- SimpleDB**存储结构**

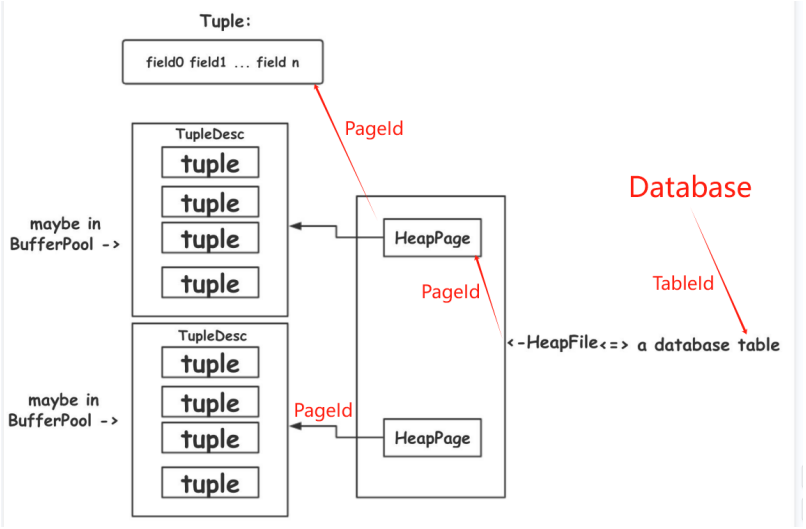


- **实验步骤：**

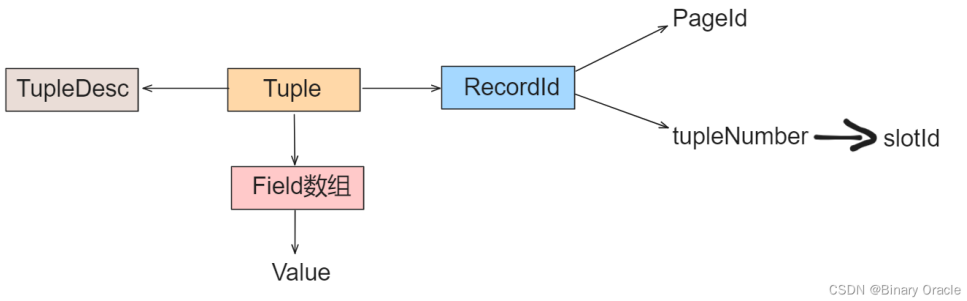
  - 实现管理tuples的类**Tuple**、**TupleDesc**，项目中已经提供了**Field**、IntField、StringField以及Type，我们只需要支持整数和(定长)字符串和固定长度的元组即可

- 实现**Catalog**
- 实现**BufferPool**的构造方法以及getPage()方法
- 实现**HeapPage**、**RecordId**以及**HeapPageId**类中的方法
- 实现**HeapFile**，映射磁盘的数据。
- 实现**SeqScan**方法，完成简单的全局扫描select *
- 本次实验的目标是通过ScanTest系统测试

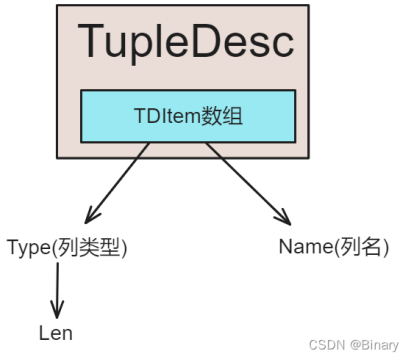
- **lec1**
  - **涉及的类：**
  - Tuple类：1.TupleDesc， 2.List<Field>， 3. RecordId ； **方法**：Field的迭代器；还有一个Tuple的迭代器类?



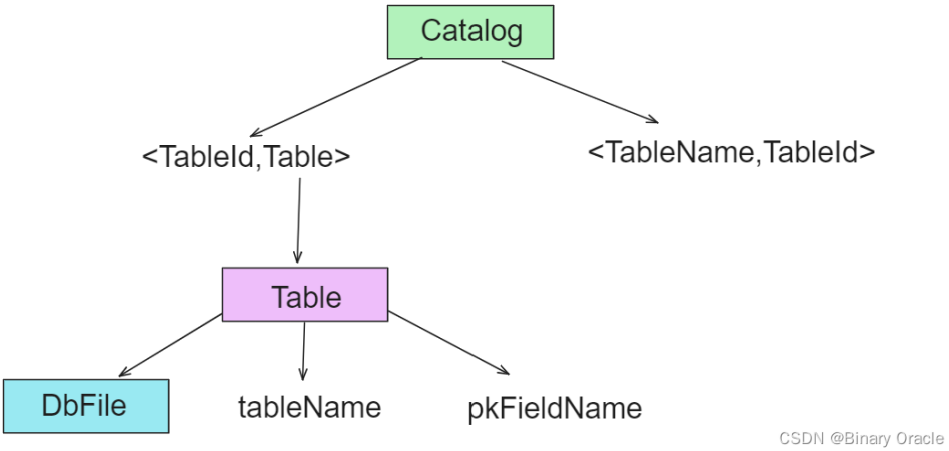
    - RecordId类PageId ; Integer(tupleNumber/slotId); **方法：**没有特别的
  - TupleDesc类: 1.List< Type ， String(fieldName)>。**方法**：List迭代器，Field数量，TupleDesc字节长度
    - Type类:1.*INT_TYPE;2.STRING_TYPE* 。***方法：****各自有getlen();和 parse(dis)将输入流转化为Type对象;*
  -



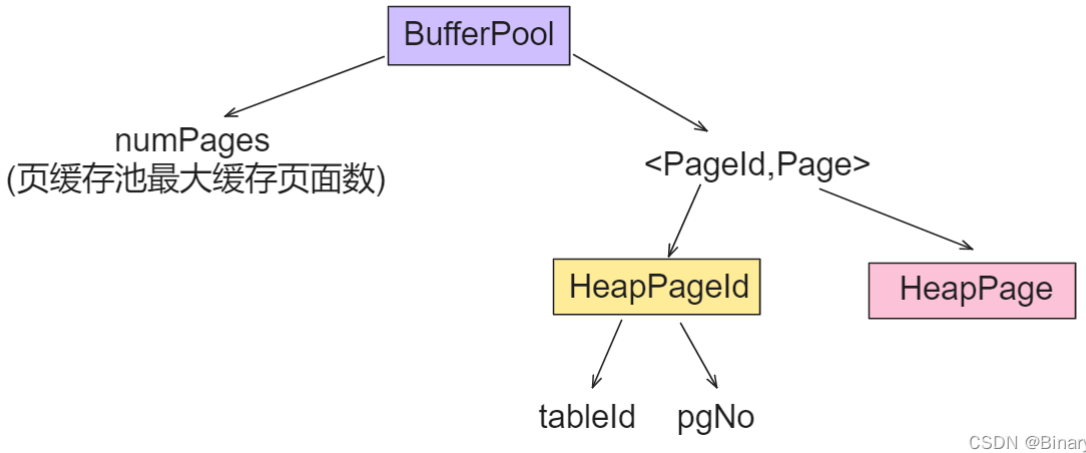
  -



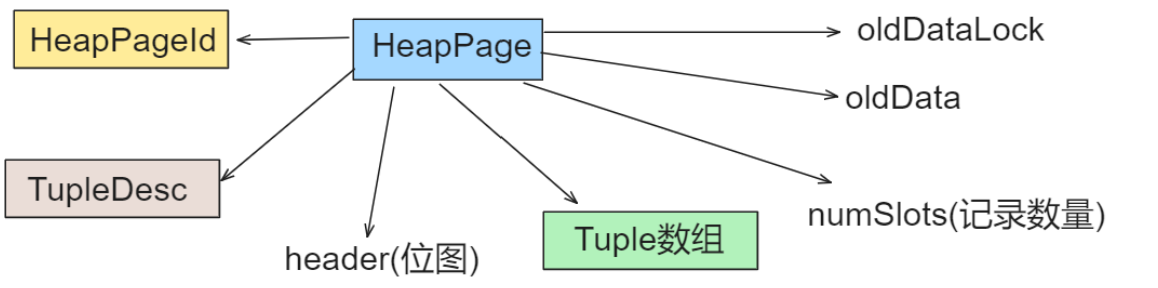
- **lec2**
  - 涉及的类：
  - **Catalog类：List<Table>;,方法：对Table的操作，增删，按tableId、tablename查File**
    - Table内部类：DbFile; String(name); String(pkeyField); **这里Table对应一个File**
      - DbFile类：HeapFile类继承；方法：readpage(); writepage(); insertTuple(); deleteTuple(); iterator(); getId()
  - 这里Table类包含了所有信息，name，id以及File



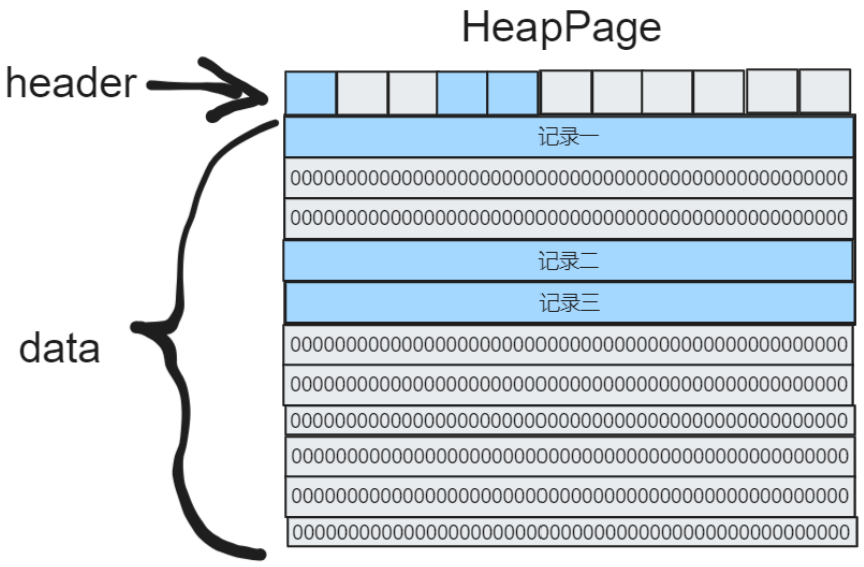
- **lec3**
  - 涉及的类：
  - **BufferPool**类：
  - 常量：Page大小，Page数量，即BufferPool大小固定；
  - 变量：int(**numPages**); LRUCache<**PageId,Page**>； PageId与Page一一对应(下一节实现)
  - 方法：主要实现getPage()，LRU缓存 + **HeapFile.**readPage()(下下一节实现)
  -



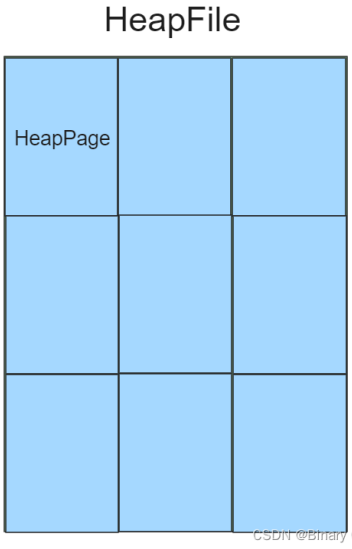
- **lec4**
  - 涉及的类：
  - **HeapPageId**；**RecordId**；**HeapFile**；
  - **HeapPageId**类：int**(tableId)**; int (**pageNum**) 方法：get，set，hashcode，equal,serialize();
  - **RecordId**类 **PageId**; Integer(**Tupleno**); 方法：get，set，hashcode，equal;
  - **HeapFile**类：HeapPageId; TupleDesc ; byte[] (header); Tuple[] ; int (numSlots，可计算得到); 其他的变量后面实验用
  - 方法：
  - 初始化输入为数据**byte[] data**，页号**HeapPageId** ，将数据 **Tuple[]**和槽**byte[] header**都初始化 (利用for)。
  - 计算Tuple数量和槽大小的方法.
  - **readNextTuple**()方法定位到Page中下一个有数据(槽为1)的Tuple
  - **getPageData**() 方法，将初始化到该HeapPage的数据序列化到磁盘(利用输出流)，后面怎么找到？没有id
  - 迭代器Iterator<Tuple> 对该页的Tuple迭代
  - 其他方法，对Tuple的操作等，后面实现
  -



  -



- **lec5**
  - 涉及的类：
  - HeapFile类，继承DbFile类： **File**; **TupleDesc** (一个File类代表一个Table,所以**TupleDesc** 也是唯一的。)
    - File类是java的io包的类，有路径和一些操作方法。
  - **readPage(PageId pid)**，把变量File路径的文件数据读取出一个Page大小，并封装成Page对象返回。
  - **getId()**, 根据变量File文件的绝对路径生成唯一id
  - 迭代**iterator()**, 重写了迭代器的方法
    - 1.open（num初始化为0，先根据getId()+num去定义唯一PageId，然后BufferPool里找Page，找到就返回一个page迭代器(实际是操作Tuple),没有先返回错误）
    - nextPage, num++,然后重复open操作。
    - hasNext，判断该page还有数据吗，没有就执行nextPage，没有page就返回false.（注，File的iterator逻辑上是迭代Page的，实际使用的是page的迭代，即Iterator<Tuple> 对Tuple迭代）
  - File类有唯一路径作为ID



- **lec6**
  - 涉及的类：**SeqScan**
    - int(**tableId)**; String(**tableAlias/自定义name**); **DbFileIterator**; **DbFile** ;
    - DbFile = Database.getCatalog().getDatabaseFile(TableId);
    - DbFileIterator = DbFile.iterator();
    - 方法：利用 DbFile.iterator();对File/Table的数据进行迭代读取
  - int(tableId); String(tableAlias); DbFileIterator; DbFile ;
  - DbFile == Database.getCatalog().getDatabaseFile(TableId);