南開大學
Nankai University

南　　开　　大　　学

计　算　机　学　院

数据库系统实验报告

SimpleDB Lab1

姓名：韩佳迅

学号：2012682

年级：2020 级

专业：计算机科学与技术

2022 年 3 月 26 日

摘要

本次实验完成了 SimpleDB 的 Lab1, 并通过了测试。通过完成 Lab1 搭建了数据库的基础架构,重点在于理解数据库、table、tuple、BufferPool、page 等之间的关系。

本报告中给出了每个类的具体设计和实现思路、重难点, 借此实验和报告理清了数据库基础架构关系。

目录

一、 Lab1 总览

GitLab 仓库地址：https://gitlab.com/hanmaxmax/SimpleDB
Lab1 主要是实现数据库的整体架构。分为以下几个部分：

(一) TupleDesc and Tuple

TupleDesc, 用于描述元组 Tuple，记录元组里 field 的个数、每个 field 的类型和名字等。
Tuple，元组，又称记录。一个元组由多个字段（field）组成。

(二) Catalog

Catlog 跟踪数据库中所有可用的表及其关联的模式，用于从磁盘读取表 table。

(三) BufferPool

BufferPool 用于管理从磁盘向内存中读写数据页 Page。它的基本单位是 Page。访问时，系统方法调用它来检索 Page，它从适当的位置获取 Page。

(四) HeapPageId and RecordID and HeapPage

HeapPage 是 Page 接口的实现，是用来组织数据的一张数据页，它的每个实例化对象用于存储一页 HeapFile 的数据，并且 HeapPage 也是 BufferPool 的基本单位。
HeapPageID 用于唯一标识 HeapPage 对象。
RecordId 是对特定 table 的特定 page 上特定 tuple 的标识。

(五) HeapFile

HeapFile 是 DbFile 接口的实现，用于与磁盘的文件交互，一张表的所有数据存到 DbFile 的 File 属性中，即一个磁盘的文件就是一张表的所有数据。HeapFile 不按特定顺序存储 tuple 集合。HeapFile 只是 Page 的集合，tuple 存储在 Page 上，每个 Page 的大小都是固定的。

(六) SeqScan

SeqScan 顺序扫描访问全表，该方法不按特定顺序读取表中的每个 tuple。

二、 设计思路及重难点

(一) Exercise 1

1. 类属性及方法

TupleDesc.java

类的成员属性:

- List<TDItem> tdItemList;

  TDItem 的列表，用于存放整个 tuple 所包含的 field 的属性

类的成员方法:

- class TDItem

  表示单个 field 的性质，包含 fieldType 和 fieldName 两个属性，实现了 toString 的方法。

- iterator()

  返回 tdItemList(TDItem 的列表) 的迭代器，用于遍历 tdItemList。

- TupleDesc 构造函数

  创建一个 TDItem 列表，描述一个 tuple 包括的所有 field 的属性。

- numFields()

  返回 tuple 中 field 个数，即 TDItem 数组的大小。

- getFieldName(int i)

  返回 tuple 中第 i（从 0 开始算的话）个 field 的 name, 即 tdItemList 中下标 i 的 fieldName。

- getFieldType(int i)

  返回 tuple 中第 i（从 0 开始算的话）个 field 的 type, 即 tdItemList 中下标 i 的 fieldType。

- fieldNameToIndex(String name)

  找到对应 fieldName 的 field 在元组中的 index。即通过遍历 tdItemList 找到对应 fieldName 的下标。

- getSize()

  返回与此 TupleDesc 对应的 tuple 的大小（单位：字节)。即通过遍历 TDItem 数组，求每一列 fieldType 大小总和。在 SimpleDB 中，来自给定 TupleDesc 的 tuple 大小是固定的。

- merge(TupleDesc td1, TupleDesc td2)

  把两个 TupleDesc 合二为一。

- equals(Object o)

  判断两个 TupleDesc 的列属性是否相同。

- toString()

  把整个元组所存的所有 field 的 fieldName 和 fieldType 转换成字符串。

Tuple.java:

类的成员属性:

- TupleDesc tupleDesc

  用于表示 tuple 的模式（列属性）

- RecordId recordId

  标识特定 table 的特定 page 上特定 tuple

- List<Field> fields

  用于存储 tuple 中的所有 field

类的成员方法:

- Tuple 的构造函数

  使用指定的类型创建新元组

- getRecordId()、setRecordId(RecordId rid)

  获得、设置当前 tuple 的 RecordId

- getTupleDesc()、resetTupleDesc(TupleDesc td)

  获取当前 tuple 的 TupleDesc 结构、重新设置 TupleDesc，而不改变其他内容

- setField(int i, Field f)、getField(int i)

  为 tuple 所存的 field 列表中下标 i 处的 field 赋新值、获得 fields 列表下标 i 处的 field 值

- toString()

  把元组中的包含的内容作为一个字符串返回

- fields()

  返回可以迭代 tuple 内 fields 列表中所有元素的迭代器

2. 思路及重难点

Exercise 1 完成了 tuple 和 tuple 结构的代码。

TupleDecs 的思路：要实现依次存储 tuple 中包含的 field 的属性，并且提供迭代、获取/查询 tuple 结构信息的对外接口。

Tuple 的思路：实现依次存储 tuple 中所存的 field 字段，并且为 tuple 设置唯一标识，还需要提供迭代、设置这些字段，设置、获取 tuple 信息的对外接口。

（具体思路见上面【类属性及方法】小节）

重难点:

1. 一开始在实现 tuple 和 tupledesc 的时候用了数组，结果写到了迭代器那里会很麻烦，所以又改成了 List 列表，利用列表的迭代器即可。

2. 刚开始做 Tuple 的时候不理解 RecordID 是什么作用，不过等到完成 RecordID.java 之后就理解它是一个对 tuple 的标识作用了。

3. 在实现这些接口的时候，要注意判断函数传入的参数是否正确，要善于使用 throw 语句。

(二) Exercise 2

1. 类属性及方法

Catalog.java

类的成员属性:

- HashMap<Integer,DbFile> id2file;
  从表 id(int) 到表文件 file(DbFile) 的哈希表
- HashMap<Integer,String> id2name;
  从表 id(int) 到表名 name(String) 的哈希表
- HashMap<Integer, String> id2pkeyField;
  从表 id(int) 到表的主键名 pkeyField(String) 的哈希表
- HashMap<String, Integer> name2id;
  从表名 name(String) 到表 id(int) 的哈希表

类的成员方法:

- Catalog 构造函数
  创建四个哈希表，用于存储已经实例化的表名、id、表、主键之间的映射关系。
- addTable(DbFile file, String name, String pkeyField)
  在哈希表中添加一个 Table，此时要实现将这个表的四种映射关系存入成员属性的四个哈希表中。
- getTableId(String name)
  根据传入的表名 name，获取表的 ID。
- getTupleDesc(int tableid)
  根据传入的表的 ID，获取该表中的元组结构 TupleDesc。
- getDatabaseFile(int tableid)
  根据传入的表的 ID，获取该表。
- getPrimaryKey(int tableid)
  根据传入的表的 ID，获取该表的主键。
- tableIdIterator()
  返回一个迭代器，可以迭代 Catalog 里存的所有表的 table id。
- getTableName(int id)
  根据传入的表的 ID，获取该表的表名。

- clear()

  从 Catalog 中删除所有的 tables，即清空所有的哈希表。

- loadSchema(String catalogFile)

  从表文件中读取表的 schema，并在数据库中创建相应的表。

2. 思路及重难点

Exercise 2 完成了数据库目录的代码。

Catalog 的思路：Catalog 需要存储数据库中所有表的一些属性之间的各种映射关系，通过底层存储的这种映射关系，实现可以从表的 a 属性获得表的 b 属性/整个表（例：根据表名获取表的 ID），并且还可以实现对整个数据库所有表的迭代。

（具体思路见上面【类属性及方法】小节）

重难点：

1. 难点在于设置什么成员属性，这部分最后通过分析框架已给出的成员函数解决。
2. 理解 Catalog 的作用，才可以明白为什么要用哈希。

(三) Exercise 3

1. 类属性及方法

BufferPool.java

类的成员属性:

- int numPages

  BufferPool 里可以存放的最大页数

- HashMap<PageId,Page> pid2pages;

  从 PageId 到 Page 的哈希表

类的成员方法:

- BufferPool 的构造函数

  根据传入的最大页数创建一个 BufferPool。

- getPageSize()、setPageSize(int pageSize)、resetPageSize()

  获取、设置、重置每页 Page 大小，默认大小为 4096。

- getPage(TransactionId tid, PageId pid, Permissions perm)

  根据 PageId 获取 Page。如果这个 Page 已经在 BUfferPool 里面，则返回对应 Page；如果不在，就添加进 Buffer Pool 里面，如果缓存的 page 数量超过缓存最大 numPages 数量，调用 evictPage() 淘汰一个页。

2. 思路及重难点

Exercise 3 完成了数据库缓冲池的一部分代码。

BufferPool 的思路：BufferPool 主要用于存储缓冲区的 Pages。在 Lab1 中，主要需要实现对外的获取/设置 PageSize 的接口，以及获取数据页的操作。

（具体思路见上面【类属性及方法】小节）

重难点：

Lab1 实现 BufferPool,java 的难点在于实现其中的 getPage 函数时，一开始不知道如何在 page 数已经满了的时候再加新的 page，最后通过分析下面（非 Lab1 所需实现的函数）的 evictPage() 函数解决如何删除 Page 的问题。

（四） Exercise 4

1. 类属性及方法

HeapPageId.java（实现 PageId 接口）

类的成员属性:

- int tableId;

  用于标识 page 所在的 table 的 table id

- int pgNo;

  page 在其所在 table 中的 number 号码

类的成员方法:

- HeapPageId 的构造函数

  根据 page 所在 table 的 table id 和 page 在其所在 table 中的 number 号码来构造 HeapPageId。

- getTableId()、getPageNumber()

  获取 page 所在 table 的 table id（tableId）、page 在其所在 table 中的 number 号码（pgNo）。

- hashCode()

  为 PageId 创建 hashcode，来唯一表示这个 PageId。实现时利用了 String 的 hashcode 方法，因此只需要构建一个能够唯一表示 PageId 的 String，然后利用它的 hashcode 即可。

- equals(Object o)

  比较两个 PageId 是否相等。

RecordID.java

类的成员属性:

- PageId pid;

  用于标识 record 所在的 page 的 page id

- int tupleno;

  (record 所在的) tuple 在其所在 page 中的 number 号码

类的成员方法:

- RecordId 的构造函数

  根据 record (tuple) 所在 page 的 page id 和 record (tuple) 在其所在 page 中的 number 号码来构造 RecordId。

- getTupleNumber()、getPageId()

  获取 record 在其所在 page 中的 number 号码 (tupleno)、record 所在的 page 的 page id (pid)。

- hashCode()

  为 RecordId 创建 hashcode，原理同 HeapPageId。

- equals(Object o)

  比较两个 RecordId 是否相等。

HeapPage.java (实现 Page 接口)

类的成员属性:

- HeapPageId pid; HeapPage 的 PageId 唯一标识

- TupleDesc td;

  HeapPage 中所存的 tuple 的 tupledesc

- byte header[];

  存储在数据页的头部，header 使用 bitmap 来保存各个 slot 的占用情况，从最低位开始数

- Tuple tuples[];

  Page 中所存的元组们，即具体的数据记录

- int numSlots;

  一共的 slot 数目，即该数据页一共有多少条记录

类的成员方法:

- HeapPage 的构造函数

  有两个参数，第一个是 HeapPageId，第二个是 byte[] 类型的 data。在构造函数中为 data 分配空间，并将空间中开始部分用于 header，header 中保存了此页 slots 的 bitmap 信息。

- getNumTuples()：

  返回此页中能够存储的 tuple 数量，用于构造函数中分配 slot。

  公式：floor((BufferPool.getPageSize()*8) / (tuple size * 8 + 1))

- getHeaderSize()

  返回此页需要多少 bytes 作为 header。公式：ceil(getNumTuples() / 8);

- getId()

  返回该 Page 的 PageId。

- readNextTuple(DataInputStream dis, int slotId)

  返回下一个存储的 tuple，即寻找到下一个被占用的 slot，返回读取的 tuple。

- getPageData()

  返回 byte[] 类型的此页数据。

- getNumEmptySlots()

  返回此页中为空的 slot 数量。

- isSlotUsed(int i)

  判断第 i 个 bit 的 slot 是否为空。具体实现利用了位运算，先在 header 找到 bit 所在位置，然后把该 bit 移位到字节最右边，之后与 1=[00000001] 做与运算，若该 bit 是 1，则运算结果为 1，否则为 0。

- Iterator<Tuple> iterator()

  返回一个能够迭代 Page 里所有 tuple 的迭代器，并且可迭代的 tuple 的 slot 不能为空。

2. 思路及重难点

Exercise 4 完成了数据库 HeapPage 及其 Id、RecordId 的代码。

HeapPageId 和 RecordId 的思路：两者都是一种标识，因此代码思路非常相似。都是存储了被标识的数据的所在位置的 Id（如 HeapPageId 存储了 table id，RecordID 存储了其所在的 PageId）和本身编号。之后提供相关信息对外接口，并创建哈希值等。

（具体思路见上面【类属性及方法】小节）

重难点：

1. 理解 table（file）、BufferPool、HeapPage、tuple（record）之间的关系。

一个 table 存在一个 file 里；一个 file 由几个 Page 存储；（由于 tuple 存在 table 里，而 table 被分为几个 page，所以）page 里面存了很多 tuple（record），而 tuple（record）里存储的是 field 字段。BufferPool 是对其中一些 page 的缓存。

2. RecordID 和 HeapPageId 的 hashcode：难点在于如何为每个 record/HeapPage 创建唯一的 hashcode。解决办法是利用 String 封装好的 hashcode 函数，只需创建唯一的 String 即可，而创建唯一的 String 可以利用标识 record/HeapPage 的唯一的 pid+tupleno / tableId+pgNo 即可。

3. 理解 HeapPage 的存储结构，HeapPage 在存储的时候先要存一个头部 header，来记录哪些 slot 不为空。并且要掌握如何通过位运算判断某比特位为 1 的情况。

（五） Exercise 5

1. 类属性及方法

HeapFile.java

类的成员属性:

- File file;

  磁盘实际储存的文件

- TupleDesc tupleDesc;

  一个 file 存的是一个 table，使用一套 tupleDesc。

类的成员方法:

- HeapFile 的构造函数

  通过特定的 file 文件和 TupleDesc 构建一个 HeapFile。

- getFile()

  返回磁盘中支持此 HeapFile 的 File 类型文件。

- getId()

  返回唯一标识此 HeapFile 的 ID。（据提示：We suggest hashing the absolute file name of the file underlying the heapfile, i.e. f.getAbsoluteFile().hashCode().）

- getTupleDesc()

  返回存储在这个 DbFile 中的 table 的 TupleDesc。

- readPage(PageId pid)

  读取 pid 对应的 Page。先找到 File 内要读取的 Page Number，读取整个 Page 返回。

- numPages()

  返回这个 HeapFile 中包含的 page 数量。

  公式：file.length() / BufferPool.getPageSize()

- iterator(TransactionId tid)

  实现 DbFileIterator 的接口，对整个表中所有的元组进行了迭代操作，基本思路是对 File 中的每个 Page 进行 tuple 迭代，需要重写 open()、hasNext()、next()、rewind()、close() 等方法。

2. 思路及重难点

Exercise 5 完成了数据库 HeapFile 的一部分代码。

HeapFile 的思路：一个 HeapFile 就是一张表/一个文件，在这个类中需要实现正确计算文件中的偏移量，随机访问该文件，以便以任意偏移量读取和写入 Page。并且需要实现对外访存 page 的接口和迭代的接口。

（具体思路见上面【类属性及方法】小节）

重难点：

1. readPage：在读取 page 的时候，需要先读取 page 的 pid 以及 pid 的 getTableId()、getPageNumber()，并依此计算出 page 的位置，然后通过 java 的 RandomAccessFile 类实现随机访问。注意要在类里写好 throw，以防特殊情况。

2. iterator：HeapFileIterator 对整个表中所有的元组进行了迭代操作，需要对 File 中的每个 Page 进行元组迭代，难点在于要注意访问完当前 page 之后还需访问下一个 page，因此需要注意自己维护当前页面的游标。

(六) Exercise 6

1. 类属性及方法

SeqScan.java

类的成员属性:

- TransactionId transactionId;
- int tableId;

  要被扫描的 table
- String tableAias;

  表的别名，用于解析
- DbFileIterator it;

  迭代器

类的成员方法:

- SeqScan 的构造函数

  创建一次特定的 transaction 对特定 table 内的数据进行的扫描
- getTableName()、getAlias()

  返回要扫描的表对应的表名、表的别名。
- reset(int tableid, String tableAlias)

  重置要扫描的表的 id 和别名
- open()、hasNext()、next()、close()、rewind()

  迭代器操作，具体实现是对 DbFileIterator it 的操作

- getTupleDesc()

  返回 table 加工后的 TupleDesc，其中包含基础 HeapFile 中的字段名，前缀为构造函数中的 tableAlias 字符串

2. 思路及重难点

Exercise 6 完成了对数据库全表扫描的代码。

SeqScan 的思路：顺序扫描全表，并提供表内数据的迭代。

（具体思路见上面【类属性及方法】小节）

重难点：

理解为表添加别名，这是因为当连接包含同名字段的表时，别名作为前缀将非常有用。因此在实现 getTupleDesc() 时，要注意要返回加工后的 TupleDesc，其中包含基础 HeapFile 中的字段名，前缀为构造函数中的 tableAlias 字符串。具体实现的时候要先把原来的 tupleDesc 和 field 个数提取出来，然后重新生成 TupleDesc。

三、 Git Commit History

GitLab 仓库地址：https://gitlab.com/hanmaxmax/SimpleDB

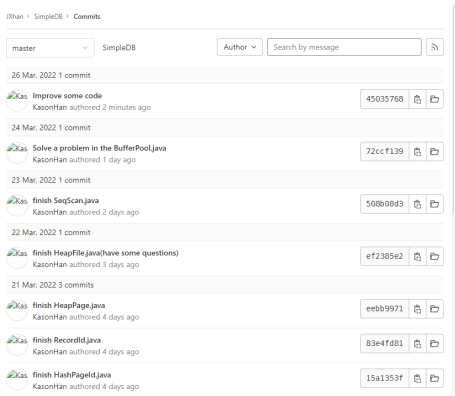

图 1

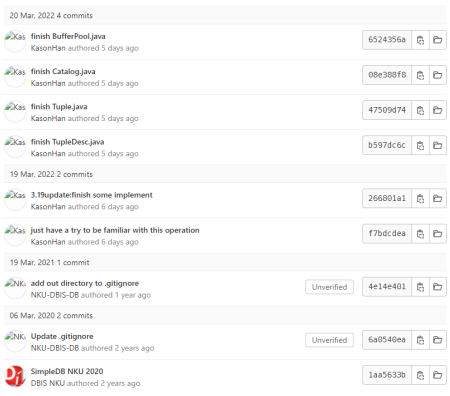

图 2

四、 改动部分

未改动 API。

五、 总结

Lab1 实现了数据库的基础架构。重点在于理解数据库的组织结构，一个 table 存在一个 file 里；一个 file 由几个 Page 存储；（由于 tuple 存在 table 里，而 table 被分为几个 page，所以）page 里面存了很多 tuple（record），而 tuple（record）里存储的是 field 字段。BufferPool 是对其中一些 page 的缓存。而 Catalog 是跟踪数据库中所有可用的表及其关联的模式，对整个数据库的哈希索引。总结 Lab1 关系如图3

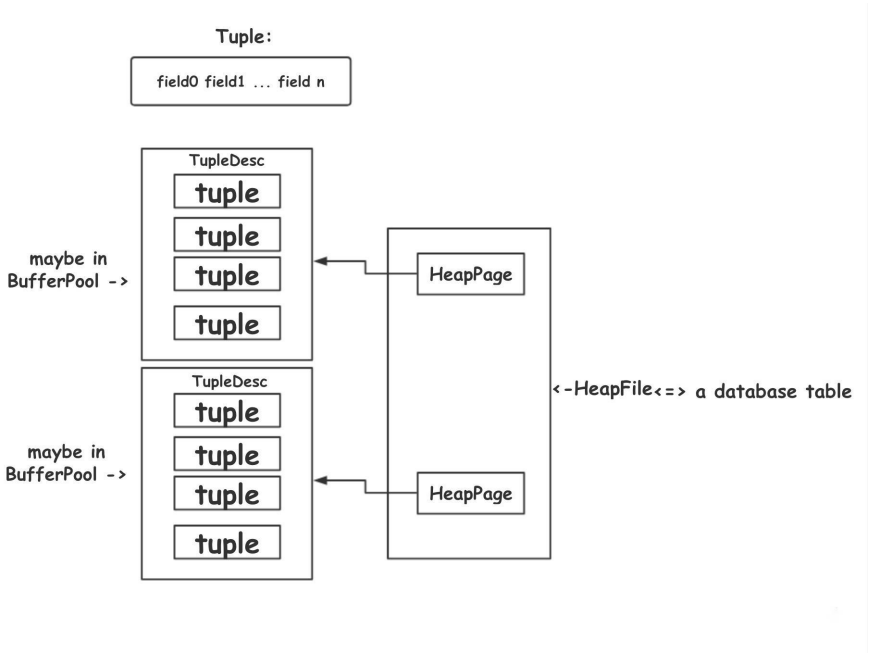
Tuple:
field0 field1 ... field n
TupleDesc
tuple
tuple
tuple
tuple
maybe in BufferPool ->
HeapPage
TupleDesc
tuple
tuple
tuple
tuple
maybe in BufferPool ->
HeapPage
<-HeapFile<=> a database table

图 3