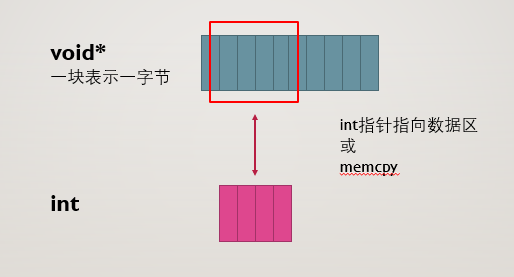
**Buffer Manager 设计笔记**

(c) 2020, Gordon Freeman

**一、 构想**

（灰色的构想表示待定）

1. 放弃为每个块指定唯一的标识符id，id将仅用于内存中块的管理，不写入文件。文件中仅保存文件头和数据。块id从1开始。
2. 使用大数据区保存块的数据，数据区IO通过memcpy实现，如此一来便可以实现定长存储与块整体传输。具体地，通过偏移量定位元组槽，通过memcpy将int, float, char[]数据以二进制形式逐字节拷贝到大数据区，或者反过来将数据区中的内容取出。Record和Index需要Catalog返回表的中的属性信息来实现上一操作。大数据区使用void\*来指向。IndexBlock直接操作内存的复杂度较高，设计一个解析函数，将数据保存在int和T数组中返回；为提高效率，RecordBlock不设计解析函数，直接返回数据区指针。



1. 每条数据仍然保留一个数据位，用于指示该条记录是否被删除。删除标记位于每条元组的起始位置。约定用int表示存在情况，0表示元组槽空闲，1表示元组槽有记录。
2. 为了降低实现难度，内存中的字符串操作以string为主，但在涉及元组时应以char\*为主。
3. 将ID也写入char\*数组，然后写入文件，如此一来可以继续沿用基于ID的实现方法。可以在char数组中保留一部分作为ID的预留位。
4. 数据文件的文件名为“<table>.data”，索引文件的文件名为“<table> - <attribute>.index”。
5. 一些函数分为了record和index两个版本，实际实现时可能会合并参数表不同的对称函数，使之成为重载函数。
6. Index文件相比Record文件具有特殊性，即一张表可能有多个属性建立了索引。为了避免引入二级映射而在openedFiles中直接保存文件名。为了减少在写文件时可能出现的错误，直接使用带有后缀名的文件名。
7. 文件直接产生在.cpp源文件的同级目录下，因为C++无权创建新的目录。
8. 不为RecordFile类和IndexFile类设计析构函数，由程序保证释放句柄时文件所持有的所有块都已被释放。
9. 数据文件使用一个空元组位的偏移量的集合来近似freelist，起节省空间的作用。在删除索引时可能涉及到删除块的操作，为了节省空间，为索引文件也涉及一个空块偏移量的集合。建立新块时，优先在这些位置上建立。数据文件的freelist文件命名为”<table>.freelist”，索引文件的freelist命名为” <table> - <attribute>.freelist”。

**二、 行为**

1. 建表

行为：新建一个数据文件句柄，如果有索引，需要新建索引文件句柄。

相关函数：

1. bool createRecordFile (string tableName, int recordLen);

Manager类外部函数

行为：函数先判断表是否存在，若不存在则抛出异常。其次判断文件是否已打开，或是否文件未打开但已存在。以上两种情况下直接返回true。若以上检查均通过，则新建一个文件句柄，将句柄加入openedRecordFiles映射，但暂时不为其分配块。blockNum，blocksInMemory，freeSlotOffsets将置为默认值。为避免写文件时出现问题，此函数将建立空的数据与freelist文件。

外部依赖：函数依赖Catalog告知表是否存在。

注意：函数默认recordLen是在不带数据位的情况下计算得到的。

为避免创造不存在的表的文件，Manager自身不得调用此函数。

1. bool createIndexFile (string tableName, string attriName, int type, int keyLen);

Manager类外部函数

行为与外部依赖与createRecordFile类似。

1. bool openRecordFile (string tableName);

Manager类内部函数

行为：函数先检查表的存在性以及是否已经打开。函数建立一个RecordFile对象，初始化其成员，并将对象加入openedRecordFiles映射。

外部依赖：依赖Catalog告知表是否存在。

1. bool openIndexFile (string tableName, string attriName);

Manager类内部函数

行为及外部依赖与openRecordFile类似。函数还需检查属性的存在性。

1. bool closeRecordFile (string fileName);

Manager类内部函数，由shutdown函数调用，该函数保证传入文件的句柄一定已经存在。

行为：利用blocksInMemory集合逐个检查当前文件在内存中持有的块，用freeRecordBlock逐个释放（当有块释放失败时，文件关闭失败，函数返回false）。之后函数向文件写入文件头的一些信息，并向另一个单独的文件写入freelist信息。最后函数将对象从openedRecordFiles映射移除并释放文件句柄。

注意：要求传入完整文件名。

1. bool closeIndexFile (string fileName);

Manager类内部函数，由shutdown函数调用，该函数保证传入文件的句柄一定已经存在。

行为类似closeRecordFile。

1. 删除表

行为：使用没有记录的文件代替原文件（包括数据文件、索引文件和freeList），随后释放内存中的块和句柄。

相关函数：

1. bool dropTable (string tableName);

Manager类外部函数

行为：函数先检查表是否存在，不存在将返回false。若表存在，调用clearFile用有文件头但没有实际数据的文件替换原文件，clearFile同时会释放块和句柄。函数将检查clearFile的返回值。删除成功将返回true。

外部依赖：依赖于Catalog告知表是否存在，并返回表上的索引信息来决定需要删除哪些索引。具体地，Catalog需要返回一个string类型（或其他可行类型）的数组，存储建立了索引的属性的名称。

注意：删除表有可能失败，这可能是块未解锁导致的。忽略dropTable返回值可能会导致严重后果。

1. bool clearRecordFile (string fileName);

Manager类内部函数，由dropTable调用，该函数保证文件存在。

行为：先调用forceFreeRecordBlock逐个释放文件在内存中所持的块，文件不在内存中时跳过此步。函数检查forceFreeRecordBlock的返回值，如果有某次调用返回false，则文件删除失败，函数返回false。之后用有文件头但没有实际数据的文件替换原数据文件。最后将文件对象从openedRecordFiles映射中移除并释放文件句柄（如果已打开）。

1. bool clearIndexFile (string fileName);

Manager类内部函数，由dropTable调用，该函数保证文件存在。

行为类似clearRecordFile。

1. bool forceFreeRecordBlock (int id);

Manager类内部函数，由clearRecordFile调用，该函数保证块在内存中。

行为：无视块是否是脏块，强制将其释放，但仍然受到锁限制。函数首先检查块是否加锁，加锁的块直接返回false。上述检查通过后，释放srcData数据区，将id从recordBlocksHeld映射移除，令num\_BlocksHeld自减，从usedTime链表移除id，将id加入freeIDs集合，并将块指针从文件对象的blocksInMemory集合移除。成功释放后返回true。

1. bool forceFreeIndexBlock (int id);

Manager类内部函数，由clearRecordFile调用，该函数保证块在内存中。

行为类似forceFreeRecordBlock。

1. 插入记录

行为： Record会要求Buffer Manager返回空闲块以存入记录。当freeSlotOffsets尚有元素时，取出一个元素，计算它所属块的偏移量，利用块链与每个块的offset成员检查块是否已在内存中，如果有则返回，否则从文件中取出块。当freeSlotOffsets没有元素时，需要为文件新建一个块。

取出块或分配块时，首先检查freeIDs向量中是否有空闲的标识符。

1. 如有，利用标识符新建一个块，初始化后从文件读取信息或直接返回。
2. 如没有，检查当前持有块量是否达到最大值。
   1. 若没有，新分配一个块ID，跳至1）。
   2. 若已达到最大值，利用LRU释放一个块，跳至1）。

此外，当表上有索引时，Index会要求插入新的key。

相关函数：

1. RecordBlock\* newRecordBlock (string tableName);

Manager类内部函数

行为：新建一个数据块。函数先检查表的存在性和文件是否打开，表不存在时输出错误信息并返回空指针，文件未打开则调用openRecordFile。调用allocateID获取一个ID，根据文件对象的blockNum初始化块的offset，再初始化其他成员（包括分配数据区）。函数将对象加入recordBlocksHeld映射。函数还将更新文件对象的blockNum，blocksInMemory，freeSlotOffsets。

外部依赖：依赖Catalog告知表是否存在。

1. RecordBlock\* retrunRecordBlock (string tableName, int offset);

Manager类外部函数

行为：用于依据传入的表名和offset返回块。函数不要求offset恰为块的起始偏移量，而是可以依据传入偏移量推算出所属块的偏移量，再将其取回。函数先检查表的存在性，再检查文件是否已打开，之后检查传入的位置上是否确实有块存在（依据blockNum）。之后，函数检查块是否已在内存中，若存在则直接返回，否则调用readRecordBlock从文件读取。

函数自动为块加锁。

外部依赖：由于函数不明确调用者是否修改块内数据，因而依赖调用者调用setDirty函数将块设为脏块。

依赖Catalog告知表是否存在。

依赖函数调用者来更改recordNum和文件对象的freeSlotOffsets。

注意：Record可以依赖RecordFile类的returnNextFreeSlot方法来返回下一个空闲槽位的偏移量，但请注意动态更新freeSlotOffsets集合。Buffer建议Record先计算返回的空槽偏移量是否在当前块内，再考虑调用returnRecordBlock，避免函数调用导致的效率损失。

1. RecordBlock\* readRecordBlock (string tableName, int offset);

Manager类内部函数

行为：函数从文件中读取块。函数需检查表是否存在、文件是否打开、传入位置上是否真的有块存在、块是否已在内存中。调用者保证offset恰为块的起始地址。调用allocateID得到一个可用的ID，新建一个对象，为ID赋值，并加入recordBlocksHeld映射。函数为offset，fileName，MAX\_RECORD\_NUM赋值，为data分配空间并读入数据，读取recordNum，其余赋默认值。函数还需将块的指针加入文件对象的blocksInMemory向量。

1. bool writeRecordBlock (int id);

Manager类内部函数，由freeRecordBlock调用

行为：函数将向磁盘写入块。函数由freeRecordBlock调用，该函数保证传入的块是未加锁的脏块。函数先为块加锁，更新数据区中的recordNum数据位，然后依据块的fileName和offset信息，调用fwrite向文件中写入块，但不会释放块。随后，函数将isDirty设置为false，将块解锁。写入成功将返回true.

1. bool freeRecordBlock (int id);

Manager类内部函数，可能由LRU，closeFile调用，这些函数保证块一定在内存中。

行为：函数释放内存中的块，如果是脏块，函数自动将其写回。函数检查块是否在内存中、是否是脏块、是否加锁，不在内存中的块直接返回true，加锁的块直接返回false，未加锁的脏块调用writeRecordBlock写回。若成功写回或无需写回（未加锁的非脏块），函数将id加入freeIDs向量，将id从usedTime链表和recordBlocksHeld映射移除，令num\_BlocksHeld自减，释放块对象（析构函数确保数据区被释放），并将块指针从文件的blocksInMemory向量移除。

注：该函数只会被LRU，closeFile等函数调用，这些函数保证id一定存在。

1. RecordBlock\* returnFreeRecordBlock (string tableName);

Manager类外部函数

行为：函数返回一个空闲块。函数先检查表的存在性。其次检查记录文件是否已打开，若未打开调用openRecordFile打开文件。利用openedRecordFiles映射定位RecordFile对象，检查freeSlotOffsets中是否有元素。若有，逐个检查blocksInMemory向量中的各个块，看是否有块的offset符合，以及该块是否有锁，若有符合条件的块则直接返回。否则调用readRecordBlock函数从文件中读取块并返回。如果freeSlotOffsets中已无元素，调用newRecordBlock建立一个新块并返回。

函数自动为块加锁（返回块的外部函数都将自动加锁），并自动设为脏块。

外部依赖：函数依赖调用者手动解锁（调用unpin）并调用updateUsedTime来更新使用时间。

当调用者向块中写入数据时，函数依赖调用者手动调用removeFreeSlot来清除一个空位偏移量。

依赖Catalog告知表是否存在。

1. IndexBlock\* newIndexBlock (string tableName, string attriName);

Manager类内部函数

行为：新建一个数据块。函数先检查属性的存在性和文件是否打开，属性不存在时输出错误信息并返回空指针，文件未打开则调用openIndexFile。调用allocateID获取一个ID，优先检查文件的freeBlockOffsets，如果无空块，根据文件的blockNum初始化块的offset，再初始化其他成员（包括分配数据区）。函数将对象加入indexBlocksHeld映射。函数还将更新文件对象的blockNum，blocksInMemory，emptyBlockOffsets。

外部依赖：依赖Catalog告知属性的存在性。

1. IndexBlock\* returnIndexBlock (string tableName, string attriName, int offset);

Manager类外部函数

行为与外部依赖类似returnRecordBlock，但index没有依据块内位置推算块偏移量的需求。函数需要检查属性的存在性。

外部依赖：除了维护指针和key之外，Index还需维护parent和keyNum变量。

1. IndexBlock\* returnRoot (string tableName, string attriName);

Manager类外部函数

行为：返回索引的根节点。函数会检查属性的存在性，但打开文件的操作由底层函数进行。向returnIndexBlock传入文件对象保存的rootOffset，返回returnIndexBlock所返回的结果。

1. IndexBlock\* readIndexBlock (string tableName, string attriName, int offset);

Manager类内部函数

行为类似readRecordBlock，在读入原始数据后还需要调用parseData解析数据，将数据填充到数组中，然后parseData函数将释放原始数据区以节省空间。函数需要检查属性的存在性。

外部依赖：依赖Catalog告知key类型，如果是字符串型，还需告知长度（parseData需要字符串长度来正确解析数据）。

1. bool writeIndexBlock (int id);

Manager类内部函数，由freeIndexBlock调用

行为类似writeRecordBlock，调用packData将数据写回srcData数据区。packData在写入数据之前先向srcData数据区写入parent和keyNum，然后将int和T数组的数据写回srcData。向srcData写回数据后，packData将释放ptrs和key数据区。

1. bool freeIndexBlock (int id);

Manager类内部函数，可能由LRU，closeFile调用

行为类似freeRecordBlock。

1. IndexBlock\* returnFreeIndexBlock (string tableName, string atrriName);

Manager类外部函数

行为：对应于returnFreeRecordBlock，供外界调用的函数。函数调用newIndexBlock，并将块加锁，设置为脏块。

外部依赖：依赖Index调用setRoot将块设置为根。此函数将更新文件对象的rootOffset成员。

1. void setRoot ();

IndexBlock类外部函数

行为：函数访问块对应的文件对象，更新其rootOffset成员。

1. void unpin ();

BlockInfo类外部函数

行为：函数为块解锁。

1. bool updateUsedTime (int id);

Manager类外部函数

行为：函数先检查块是否存在，其次检查是否在usedTime链表中。若块不存在则直接返回false；若确实在链表中，则将其移至usedTime链表的起始位置；不存在于链表中则直接在链表头部插入此id。更新完成后返回true。

注意：当BlocksHeld映射中不存在id为传入值的块时，更新会失败。

1. void setDirty ();

BlockInfo类外部函数

行为：函数将块设置为脏块。

1. int LRU ();

Manager类内部函数

行为：返回一个可用的块ID. 函数从usedTime映射的开头向后检查，找到第一个未加锁的非脏块，调用freeBlock函数将块释放。freeBlock失败则continue. freeBlock函数会将块id加入freeIDs向量并让num\_BlocksHeld自减，因而LRU函数需要再逻辑撤销这些操作。如果遍历usedTime链表都无法找到未加锁的非脏块，则重新遍历，此时只寻找未加锁的块，调用freeBlock将块释放。freeBlock失败则continue直到成功释放一个块。如所有块都加锁将返回-1. 该函数仅用于返回一个可用的块ID，不会在堆上建立新块。

注意：LRU可能失败，可能因为所有8192个块都加锁，此时返回-1.

1. int allocateID ();

Manager类内部函数

行为：用于返回一个可用的块ID. 函数先检查freeIDs向量中是否还有元素，如有则选取一个返回，否则需要检查num\_BlocksHeld是否已达到最大值。如果没有，新分配一个ID（= num\_BlocksHeld + 1）然后返回。以上两种情形都需要让num\_BlocksHeld自增。如果已经达到最大值，调用LRU来释放一个块，并返回LRU的返回值。

1. 选择记录

下面逐条讨论选择算法中Buffer Manager承担的任务。由于我们的实现基于二级索引，一部分算法未在下面列出。

1. A1（线性搜索）：线性扫描各元组。Buffer Manager需要逐个返回文件中的块。
2. A4（二级索引，等值）：返回符合条件的属性值对应的唯一元组或不返回（索引一定建立在主码或unique属性上）。Buffer Manager依据Index传入的索引块的偏移量返回索引块，或根据元组的偏移量，计算出元组所在块的偏移量，然后返回该条元组所在的块。
3. A6（二级索引，比较）：顺序扫描叶节点，返回符合条件的元组。不建议采用这种算法。
4. 关于合取选择：可以直接使用线性扫描。如果时间充裕，可以检查是否有建立在索引上的等值条件，先使用A4取回唯一的元组（如果无符合条件元组则直接返回），然后进一步判断其他条件。

综上，Buffer Manager考虑在A1与A4算法下的行为。

相关函数：

1. RecordBlock\* returnFirstRecordBlock (string tableName);

Manager类外部函数

行为：函数返回文件中的第一个块。函数先检查表的存在性，其次检查文件是否在内存中。当文件已在内存中时，检查其是否有任何块。文件没有块返回nullptr。若文件有块，调用returnRecordBlock来返回。

外部依赖：依赖Catalog告知表的存在性。依赖调用者解锁块。

1. RecordBlock\* returnNextRecordBlock (int curID);

Manager类外部函数

行为：函数返回同文件中的下一个块。函数先检查块是否在内存中（即块ID是否合法），不在内存中则返回nullptr；在内存中，说明表一定存在，且文件已经打开。函数计算出需返回块的offset，调用returnRecordBlock返回块。

外部依赖：依赖调用者解锁块。

其他函数已在3.中说明。

1. 删除记录

行为：Record在处理删除和选择时的行为类似，后者是将元组输出，而前者是将元组前的标识符置为0，因而Buffer Manager的行为完全类似。Index可能需要删除整个块。

相关函数：

1. static bool removeIndexBlock(vector<int> ids);

行为：函数先检查id的合法性，如id已在内存中，说明块在内存且文件已打开。函数遍历向量，每一轮将块解锁后调用forceFreeIndexBlock来释放。由于removeIndexBlock会将块解锁，因而forceFreeIndexBlock函数不会失败。同时需要调整文件的emptyBlockOffsets和blockNum。

此函数的实现基于这样的假设：整棵树中的删除完成后，所有从树中移除的块必须被pin住来保证不被写回。

1. 建立索引

行为：Index会要求Buffer建立索引文件，随后会要求返回新块用以存储数据。在处理元组指针时，Index会要求Record返回元组在文件中的位置，这一需求可以通过Record调用块的returnOffset函数，再进一步计算实现。

相关函数已在3.中说明。

1. 删除索引

行为：删除索引文件。Buffer Manager将调用clearIndexFile实现这一行为。

相关函数：

1. static bool dropIndex(string tableName, string attriName);

Manager类外部函数

行为：先判断属性的存在性以及是否加了索引，之后调用clearIndexFile来清除文件。清除成功返回true，属性不存在、未加索引、清除文件失败返回false。

外部依赖：依赖Catalog告知属性是否存在以及是否有索引。

1. 其他函数
2. bool shutdown ();

Manager类外部函数

行为：遍历openedIndexFile映射和openedRecordFile映射，调用closeFile来关闭各个文件。函数将检查closeFile的返回值，若有文件关闭失败，则Buffer Manager关闭失败，函数将返回false。成功关闭所有文件，返回true。

注意：Buffer Manager可能关闭失败，这可能是有块尚未解锁所致。此时忽略shutdown的返回值而直接结束程序可能导致更新的数据丢失，因而请使用块的Manager注意调用unpin来解锁块。

**三、待办任务**

1.（如有时间）简化合法性检查。一个想法：上层函数负责输入合法性检查，下层函数负责文件是否打开等与调用者无关的检查。

2. Bug Fix