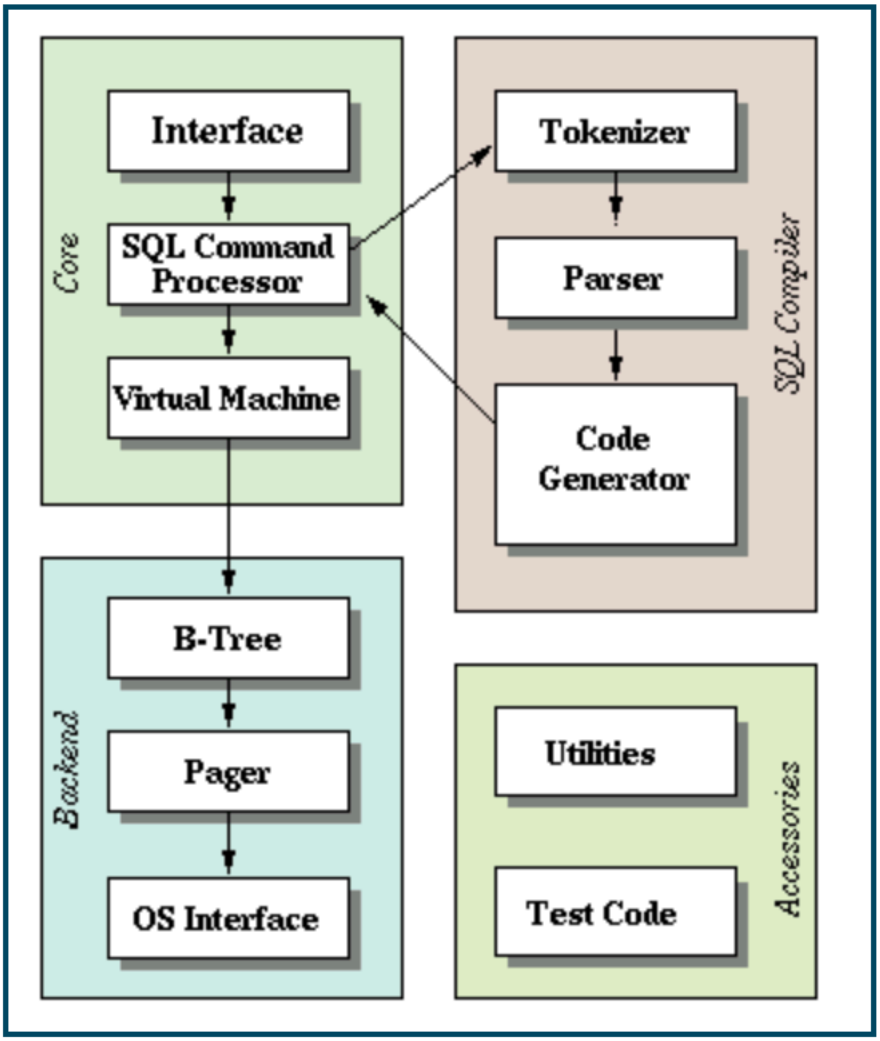
Homework 4 Report

2-A

1.

a.



b.

(1)Interface:

user can make command to the Interface

->main.c, legacy.c, vmbeapi.c

(2)Tokenizer:

Tokenizer breaks the SQL text into several tokens and
hands these tokens one by one to Parser

->tokenize.c

(3)Parser:

Parser assigns meaning to token based on their context.

->parse.y

(4)Code Generator:

every tokens would assemble into a parser tree, and code generator will perform work of the SQL statement, ex: insert, delete, where

->attach.c, auth.c, build.c, delete.c, expr.c, insert.c, pragma.c, select.c, trigger.c, update.c, vacuum.c, where.c, wherecode.c, whereexpr.c

(5)Bytecode Engine(Virtual machine):

after translating SQL statement into bytecode, we have to run this code through Bytecode Engine, which is a virtual machine

->vdbe.c, vdbe.h

(6)B-tree:

every index and table has a individual b-tree to store data

->btree.c, btree.h

(7)Page cache(Pager):

the B-tree will request some pages from the Page cache and notify the Page cache when it want to modify pages or commit or rollback changes.

->pager.c, pager.h

(8)OS interface:

provide portability between across operating systems

->os_unix.c, os_win.c

(9)Utilities:

such as Memory allocation, caseless string comparison routines, portable text-to-number conversion routines, etc

->util.c
c.

(1)比較sqlite和DBMS：

一般的DBMS有DDL compiler 和DML preprocessor，但是sqlite只有一個tokenizer

(2)比較sqlite和database manager：

一般的database manager要有權限才能讀取，因此有authorization control和command processor，但是sqlite沒有

2.

a.

(1)Tokenizer:

user can make command to the Interface

(2)Parser:

Parser assigns meaning to token based on their context.

(3)Code Generator:

every tokens would assemble into a parser tree, and code generator will perform work of the SQL statement, ex: insert, delete, where

(4)Bytecode Engine(Virtual machine):

after translating SQL statement into bytecode, we have to run this code through Bytecode Engine, which is a virtual machine

b.

(1)Where clause:

如果一個query包含Where那會根據And被拆成數個部分，Or不受影響，然後去檢查能不能使用index，判斷的標準是

(I)看在兩個連續columns中有沒有某個column沒有限制(gap)，如果有，之後（右方）的column就不能用

ex: create index idx on table(a,b,c,d)

where a=5 and d=10 => no b, c => only a column is usable

(II)如果其中一個column的限制是inequality，那右方的column都不能使用

ex: create index idx on table(a,b,c,d)

where a=1 and b>2 and c=10 =>b>2 =>only a column is usable

(2)Between clause:

如果query是 a between b and c，那會把query改成 a>=b and a<=c的形式，因此可以不需要跑完整個table

(3)Like clause:

Like也可以使用index的搜尋，不過有一些限制：

(I)Like的左邊是一個index column 的名字，同時符合TEXT affinity

(II)Like的右邊是字串

(III)Escape不能出現在Like query中

如果都符合上述條件，我們可以修改query，

ex: n like xy%

我們可以改寫為 n>='xy' and n < 'xz'

3.

a.

(1)pages

database file 主要由pages構成，就如同array和index的關係。pages根據功能可以分成The lock-byte page, The freelist page, The b-tree page, The pointer map page。freelist page 用來儲存暫時沒用到的page，lock-byte page負責shared lock和exclusive lock等，lock byte page用來幫助移動pages時速度加快（透過指向parent），而b-tree page可以說是最重要的部分，分為internal, leaf和overflow，internal指向接下來要去的地方（為search navigate），leaf存真正的資料，overflow則是在存不下的時候使用

(2)header

每個page大小約在512(2^9)到65536(2^16)之間，而page最前面100byte就是header，功能大概是

Offset	Size	Description
0	16	Header string: “SQLite format 3”
16	2	Page size in bytes.
18	1	File format write version
19	1	File format read version
20	1	Bytes reserved at the end of each page
21	1	Max embedded payload fraction
22	1	Min embedded payload fraction
23	1	Min leaf payload fraction
24	4	File change counter
28	4	File size in pages
32	4	First freelist page
36	4	Number of freelist pages
40	60	Fifteen 4-byte meta values

(3)format

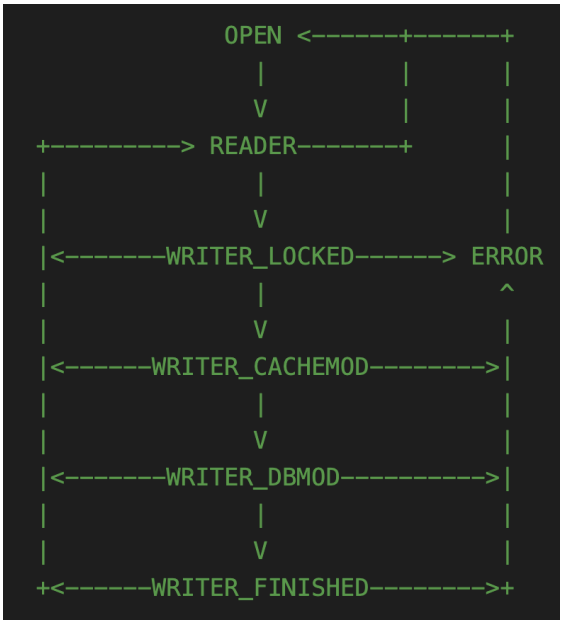
offset 18和19用來決定file format 是write還是read

b.

承接format內容，如果read-version比2大，那datafile就不能讀也不能寫，反之即可被讀，如果write-version比2小（包含2），datafile就可以被寫

4.

a.



(1)open state:

最初始的state

(2)reader state:

如果有read的requirement，這個state會讀file，並套上shared lock，然而如果現在是exclusive lock，就無法讀（前往）error state

(3)writer locked state:

如果有write的requirement，會觸發reserved-lock，但是還不會動file，journal file不會被寫也不會被打開

(4)writer cachemod state:

如果要write的內容在page的上層，就會進入這個state，會觸發reserved-lock，此時journal file會被開啟，並且將header寫進journal file，page cache會被動，但是在disk上的內容不會被動

(5)writer DBmod state:

如果要write的內容是database file的content，就會進入這個state，並且觸發exclusive-lock，此時journal file會被開啟，並且將header寫進journal file，page cache會被動，在disk上的內容有可能會被動

(6)writer finished state:

成功寫入後，會進入這個state，同時不會再做任何寫入，維持exclusive lock，並finalize journal file

(7)error state:

一旦進入error state，任何read或write都會回傳error

b.

ex1:

假設今天有五個query，(1)read file1, (2)read file1, (3)write file1, (4)read file1, (5)write file1，執行(1)read file1後，進入reader state，此時套上shared lock，回到open state，執行(2)read file1，進入reader state，此時因為是shared lock，所以還是可以讀，接著執行(3)write file1，進入writer locked state，升級為reserved-lock，(4)進入reader state，(5)無法執行，(1)(2)都繼續執行，進入writer cachemod state，將內容寫入cache中，狀態升級為pending，此時沒有任何query可以取得shared lock（但是在這個例子中也沒有query需要使用shared lock），但擁有shared lock的(1)(2)(4)都還繼續執行，因此持續等，直到(1)(2)(4)都執行結束，此時都回復為unlocked，只有一個reserved-lock，可以進入writer DBmod state了，將資料寫入disk，寫完後進入writer finished state，做最後的完善後恢復為unlocked，現在可以執行(5)write file1了，就重複以上動作，(i)在(3)write的時候，雖然(1)(2)(4)都在read，但是他們所讀取的都是的資料，這時候(3)所更改的資料對(1)(2)(4)就是invisible，(ii)當有兩個或以上在同時update同一個資料庫，因為我

們不知道誰會是最後修改的，會發生race condition，因此nondeterministic

2-B

a.component: Update

b.description:

在Update.c中，主要有sqlite3ColumnDefault、sqlite3Update和updateVirtualtable，被parser用來分析update的query

c.大約流程圖:

(1)sqlite3ColumnDefault:

建立table後，有可能插入某個column時會miss部分資料，此時錯誤了，如果有default值，就回傳default值，如果沒有，用P4代替

```
void sqlite3ColumnDefault(Vdbe *v, Table *pTab, int i, int iReg){
  assert( pTab!=0 );
  if( !pTab->pSelect ){
    sqlite3_value *pValue = 0;
    u8 enc = ENC(sqlite3VdbeDb(v));
    Column *pCol = &pTab->aCol[i];
    VdbeComment((v, "%s.%s", pTab->zName, pCol->zName));
    assert( i<pTab->nCol );
    sqlite3ValueFromExpr(sqlite3VdbeDb(v), pCol->pDflt, enc,
                         pCol->affinity, &pValue);
    if( pValue ){
      sqlite3VdbeAppendP4(v, pValue, P4_MEM);
    }
  }
#ifndef SQLITE_OMIT_FLOATING_POINT
  if( pTab->aCol[i].affinity==SQLITE_AFF_REAL && !IsVirtual(pTab) ){
    sqlite3VdbeAddOp1(v, OP_RealAffinity, iReg);
  }
#endif
```

一個column的default有兩種可能，有可能是建立table時user提供的也有可能是Alter table的時候，前者不需要p4值，後者回傳p4值，因為

Alter table的時候產生的default value只會是number, string or null。因此只要是這三種就需要p4值，sqlite3ValueFromExpr()這個function的用意就是要將這三種轉成sqlite3_value。

(2)sqlite3Update:

```
/*
** Process an UPDATE statement.
**
**   UPDATE OR IGNORE table_wxyz SET a=b, c=d WHERE e<5 AND f NOT NULL;
**          \_______/ \________/     \______/       \________________/
*            onError   pTabList      pChanges             pWhere
*/
void sqlite3Update(
  Parse *pParse,         /* The parser context */
  SrcList *pTabList,     /* The table in which we should change things */
  ExprList *pChanges,    /* Things to be changed */
  Expr *pWhere,          /* The WHERE clause.  May be null */
  int onError,           /* How to handle constraint errors */
  ExprList *pOrderBy,    /* ORDER BY clause. May be null */
  Expr *pLimit,          /* LIMIT clause. May be null */
  Upsert *pUpsert        /* ON CONFLICT clause, or null */
){
```

主要負責update的函數，由上而下分別是傳給parse的值，要被改的table，要改的內容，where clause的內容，錯誤處理，根據clause排序，被限制的clause，衝突的clause，

傳入query後，update之後接的就是onError，後面是要被改的table，set之後接上要更改的內容，由於範例中有where所以pwhere不能是NULL

過程：

```
  /* Locate the table which we want to update.
  */
  pTab = sqlite3SrcListLookup(pParse, pTabList);
  if( pTab==0 ) goto update_cleanup;
  iDb = sqlite3SchemaToIndex(pParse->db, pTab->pSchema);

  /* Figure out if we have any triggers and if the table being
  ** updated is a view.
  */
```

確認table->檢查triggers

```
/* Allocate a cursors for the main database table and for all indices.
** The index cursors might not be used, but if they are used they
** need to occur right after the database cursor.  So go ahead and
** allocate enough space, just in case.
*/
iBaseCur = iDataCur = pParse->nTab++;
iIdxCur = iDataCur+1;
pPk = HasRowid(pTab) ? 0 : sqlite3PrimaryKeyIndex(pTab);
testcase( pPk!=0 && pPk!=pTab->pIndex );
```

->table是不是view->動態建立足夠的空間

```
/* Resolve the column names in all the expressions of the
** of the UPDATE statement.  Also find the column index
** for each column to be updated in the pChanges array.  For each
** column to be updated, make sure we have authorization to change
** that column.
*/
chngRowid = chngPk = 0;
for(i=0; i<pChanges->nExpr; i++){
  if( sqlite3ResolveExprNames(&sNC, pChanges->a[i].pExpr) ){
```

->開始generate code->找出所有update後要使用得column，確保我們有足夠的權限

```
/* The SET expressions are not actually used inside the WHERE loop.
** So reset the colUsed mask. Unless this is a virtual table. In that
** case, set all bits of the colUsed mask (to ensure that the virtual
** table implementation makes all columns available).
*/
pTabList->a[0].colUsed = IsVirtual(pTab) ? ALLBITS : 0;

hasFK = sqlite3FkRequired(pParse, pTab, aXRef, chngKey);

/* There is one entry in the aRegIdx[] array for each index on the table
** being updated.  Fill in aRegIdx[] with a register number that will hold
** the key for accessing each index.
*/
if( onError==OE_Replace ) bReplace = 1;
```

->如果不是virtual table，就直接reset

```
  aRegIdx[nAllIdx] = ++pParse->nMem;  /* Register storing the table record *
  if( bReplace ){
    /* If REPLACE conflict resolution might be invoked, open cursors on all
    ** indexes in case they are needed to delete records.  */
    memset(aToOpen, 1, nIdx+1);
  }

  if( pParse->nested==0 ) sqlite3VdbeCountChanges(v);
  sqlite3BeginWriteOperation(pParse, pTrigger || hasFK, iDb);

  /* Allocate required registers. */
  if( !IsVirtual(pTab) ){
    /* For now, regRowSet and aRegIdx[nAllIdx] share the same register.
    ** If regRowSet turns out to be needed, then aRegIdx[nAllIdx] will be
    ** reallocated.  aRegIdx[nAllIdx] is the register in which the main
    ** table record is written.  regRowSet holds the RowSet for the
    ** two-pass update algorithm. */
```

->更新table紀錄->檢查有沒有衝突

```
  /* Start the view context. */
  if( isView ){
    sqlite3AuthContextPush(pParse, &sContext, pTab->zName);
  }

  /* If we are trying to update a view, realize that view into
  ** an ephemeral table.
  */
```

->處理view

```
#endif

  /* Resolve the column names in all the expressions in the
  ** WHERE clause.
  */
  if( sqlite3ResolveExprNames(&sNC, pWhere) ){
    goto update_cleanup;
  }
```

->處理where

```
    /* Open every index that needs updating. */
    if( eOnePass!=ONEPASS_OFF ){
      if( aiCurOnePass[0]>=0 ) aToOpen[aiCurOnePass[0]-iBaseCur] = 0;
      if( aiCurOnePass[1]>=0 ) aToOpen[aiCurOnePass[1]-iBaseCur] = 0;
    }

    if( eOnePass==ONEPASS_MULTI && (nIdx-(aiCurOnePass[1]>=0))>0 ){
      addrOnce = sqlite3VdbeAddOp0(v, OP_Once); VdbeCoverage(v);
    }
    sqlite3OpenTableAndIndices(pParse, pTab, OP_OpenWrite, 0, iBaseCur,
                               aToOpen, 0, 0);
    if( addrOnce ){
      sqlite3VdbeJumpHereOrPopInst(v, addrOnce);
    }
  }

  /* Top of the update loop */
```

->開始處理index

```
  }
}

/* If the rowid value will change, set register regNewRowid to
** contain the new value. If the rowid is not being modified,
** then regNewRowid is the same register as regOldRowid, which is
** already populated.  */
assert( chngKey || pTrigger || hasFK || regOldRowid==regNewRowid );
if( chngRowid ){
  sqlite3ExprCode(pParse, pRowidExpr, regNewRowid);
  sqlite3VdbeAddOp1(v, OP_MustBeInt, regNewRowid); VdbeCoverage(v);
}

/* Compute the old pre-UPDATE content of the row being changed, if that
** information is needed */
if( chngPk || hasFK || pTrigger ){
```

->以loop檢查row需不需要update

->結束

(3)updateVirtualTable:

```
** Generate code for an UPDATE of a virtual table.
**
** There are two possible strategies - the default and the special
** "onepass" strategy. Onepass is only used if the virtual table
** implementation indicates that pWhere may match at most one row.
**
** The default strategy is to create an ephemeral table that contains
** for each row to be changed:
**
**   (A)  The original rowid of that row.
**   (B)  The revised rowid for the row.
**   (C)  The content of every column in the row.
**
** Then loop through the contents of this ephemeral table executing a
** VUpdate for each row. When finished, drop the ephemeral table.
**
** The "onepass" strategy does not use an ephemeral table. Instead, it
** stores the same values (A, B and C above) in a register array and
** makes a single invocation of VUpdate.
*/
```

default way : 創建一個暫時表，包含三個row，原始的rowid，更改後的rowid，接著跑回圈，對每個row update，完成後就可以捨棄暫時表

```
  Parse *pParse,        /* The parsing context */
  SrcList *pSrc,        /* The virtual table to be modified */
  Table *pTab,          /* The virtual table */
  ExprList *pChanges,   /* The columns to change in the UPDATE statement */
  Expr *pRowid,         /* Expression used to recompute the rowid */
  int *aXRef,           /* Mapping from columns of pTab to entries in pChange
  Expr *pWhere,         /* WHERE clause of the UPDATE statement */
  int onError           /* ON CONFLICT strategy */
){
```

由上而下，傳給parsing，要改的虛擬表，建立的虛擬表，要被改的column，驗證rowid的expression，map pTab和pChanges，處理where，處理error

過程：

開始

```
  /* Start scanning the virtual table */
  pWInfo = sqlite3WhereBegin(pParse, pSrc, pWhere, 0,0,WHERE_ONEPASS_DESIRED
  if( pWInfo==0 ) return;

  /* Populate the argument registers. */
  for(i=0; i<pTab->nCol; i++){
    assert( (pTab->aCol[i].colFlags & COLFLAG_GENERATED)==0 );
    if( aXRef[i]>=0 ){
      sqlite3ExprCode(pParse, pChanges->a[aXRef[i]].pExpr, regArg+2+i);
    }else{
      sqlite3VdbeAddOp3(v, OP_VColumn, iCsr, i, regArg+2+i);
      sqlite3VdbeChangeP5(v, OPFLAG_NOCHNG);/* Enable sqlite3_vtab_nochange(
    }
  }
```

->獲取更新訊息

```
  /* There is no ONEPASS_MULTI on virtual tables */
  assert( eOnePass==ONEPASS_OFF || eOnePass==ONEPASS_SINGLE );

  if( eOnePass ){
    /* If using the onepass strategy, no-op out the OP_OpenEphemeral coded
    ** above. */
    sqlite3VdbeChangeToNoop(v, addr);
    sqlite3VdbeAddOp1(v, OP_Close, iCsr);
  }else{
    /* Create a record from the argument register contents and insert it int
    ** the ephemeral table. */
    sqlite3MultiWrite(pParse);
    sqlite3VdbeAddOp3(v, OP_MakeRecord, regArg, nArg, regRec);
#ifdef SQLITE_DEBUG
```

->選擇要用default way還是onepass way

->default的話就建立暫時表

```
    /* Begin scannning through the ephemeral table. */
    addr = sqlite3VdbeAddOp1(v, OP_Rewind, ephemTab); VdbeCoverage(v);

    /* Extract arguments from the current row of the ephemeral table and
    ** invoke the VUpdate method.  */
    for(i=0; i<nArg; i++){
      sqlite3VdbeAddOp3(v, OP_Column, ephemTab, i, regArg+i);
    }
  }
  sqlite3VtabMakeWritable(pParse, pTab);
  sqlite3VdbeAddOp4(v, OP_VUpdate, 0, nArg, regArg, pVTab, P4_VTAB);
  sqlite3VdbeChangeP5(v, onError==OE_Default ? OE_Abort : onError);
  sqlite3MayAbort(pParse);

  /* End of the ephemeral table scan. Or, if using the onepass strategy,
  ** jump to here if the scan visited zero rows. */
  if( eOnePass==ONEPASS_OFF ){
    sqlite3VdbeAddOp2(v, OP_Next, ephemTab, addr+1); VdbeCoverage(v);
    sqlite3VdbeJumpHere(v, addr);
    sqlite3VdbeAddOp2(v, OP_Close, ephemTab, 0);
  }else{
    sqlite3WhereEnd(pWInfo);
  }
```

->開始掃描暫時表內容

->掃描完成

->結束