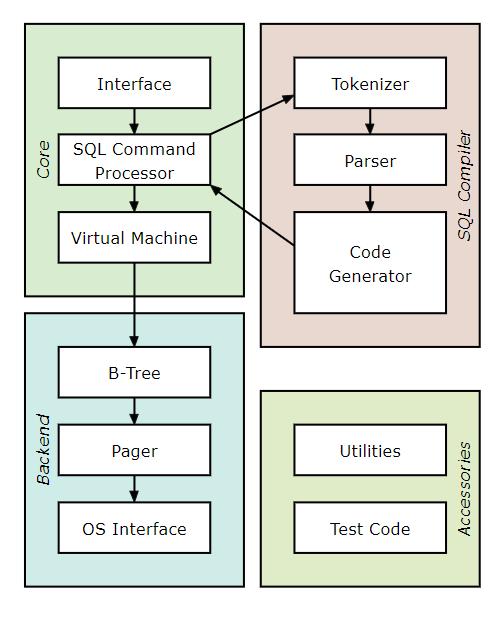
2-A:

1.a 來源: <https://sqlite.org/arch.html>



1.b 來源: <https://sqlite.org/arch.html>

Interface: interface是可以讓開發者處理資料庫的接口，最主要的是C interface，讓開發者可以直接使用C語言和資料庫互動，除此之外還有許多種，讓sqlite可以處理多種語言，並且Interface還會利用底線處理重複名稱的問題。

Tokenizer: tokenizer會將輸入的字串分解並且分別送到parser去判斷這行指令是什麼。可以在tokenize.c裡面看到，他是利用指標去控制現在要看哪個位置，197行的getToken，是去確認接下來的token是什麼type，並且依照讀到的type進行不同行為，並且在567行的sqliteRunParser去進行parser的部分，將token賦予行動。

Parser: parser將讀入的指令轉化為有效的行動，sqlite是用lemon parser generater，比較不容易出錯，並且在語法錯誤時不會發生memory leak，parser.y是文法檔，裡面是各個指令的語法。

Code generator: 產生bytecode來執行statement，有很多檔案都在code generator的範圍裡，不同的檔案去解決不同的statement，會視輸入到parser的指令為何調用不同檔案，例如select.c，就是在處理select statement，像是2858行的multiselect就是在處理巢狀的select。

Bytecode engine: 將先前在code generator裡產生的bytecode放到virtual machine執行。

B-tree: sqlite是用B-tree來儲存資料，btree.c就是在實作一個B-tree。在一個database裡會用到許多B-tree，而這些B-tree都存在同一個disk裡。

Page cache: 他是負責檔案相關的操作，Pager會將database檔案分成固定大小的page，並且讀入和寫入這些pages，除此之外，還會快取常用的pages讓操作更快，在pager.c中，就是在實作存取disk file，並且還會避免同時對同一個檔案操作。

OS interface: 針對不同的作業系統會有不同的存取指令，因此sqlite用VFS處理這些問題，VFS提供在不同作業系統下，各種操作檔案的方法。

Utilities: 各種utilities的function放在util.c，在有需要的時候可以直接呼叫裡面的function，如memory allocation, caseless string comparison routines等，除此之外，還有hash table在hash.c，sqlite自己的printf在print.c，也都算在utilities的範圍內。

Test code: 名字開頭有test的都是測試專用，於平常使用時不同，在測試時才會使用這些。

1.c 來源: <https://sqlite.org/about.html>

Sqlite不需要連接到伺服器，他可以直接讀寫檔案，並且還可以在不同的作業系統上運作，再轉換系統時可以不用修改程式，除此之外sqlite的大小很小，大約只有幾百KB，而且也不用下載或是提前設定，可以直接使用。

2.a 來源: <https://www.sqlite.org/howitworks.html>

<https://www.sqlite.org/opcode.html>

Tokenizer: 將輸入的文字分解成一個個token。

Parser: 將所有token除了不需要的空格和註解，放到parser裡面，parser會產生一個語法樹。

Code generator: 將得到的語法樹解析，先找到對應的table跟欄位，優化語法樹，再依照語法樹要求的指令選出合適的演算法，輸出bytecode。

Bytecode engine: 用sqlite3\_step()執行bytecode。

2.b.i 來源: <https://sqlite.org/optoverview.html>

Where: 先把where clause看成terms，如果是用and連接就將每項個別看成一個term，如果是用or連接就將整體看成一個term，在whereInt.h 217行。接著在whereexpr.c 132行，看這個term可不可以用index，在這裡是檢查他的operator。

Like: 在whereexpr.c 161行，like要可以optimize，他的右邊就必須事非萬用字元開頭的字串，左邊要是欄位且只能放NULL、字串、或BLOB。先檢查operator是不是like，接著算第一個萬用字元前有幾個字元，然後看可不可以optimize，要能optimize，他的pattern不能以萬用字元開頭，並且萬用字元前不能是illegal的字元，或是沒有跳脫字元。之後分成左右邊來看，左邊是欄位，如果左邊不是一般的文字，那他的上下界不能看起來像是文字，不然會被當成文字處理，也就是無法optimize；如果右邊是變數，且變數的值不需要用到like，需要加一個dummy op\_variable，不然會出問題。若like可以optimize，就可以直接查找index，不用去掃描整個table。

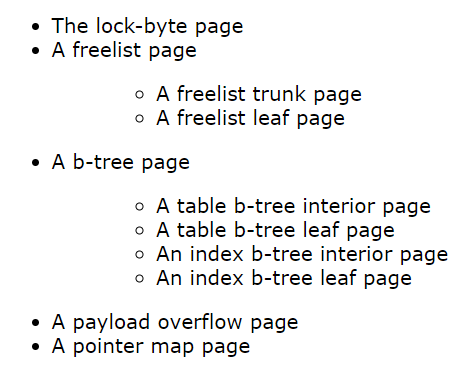
Between: 在whereexpr.c 1223行，他會創造兩個新的virtual terms，分別是他的上下界，在執行的時候，如果index符合新的這兩項，就不會用到between，反過來也是一樣。

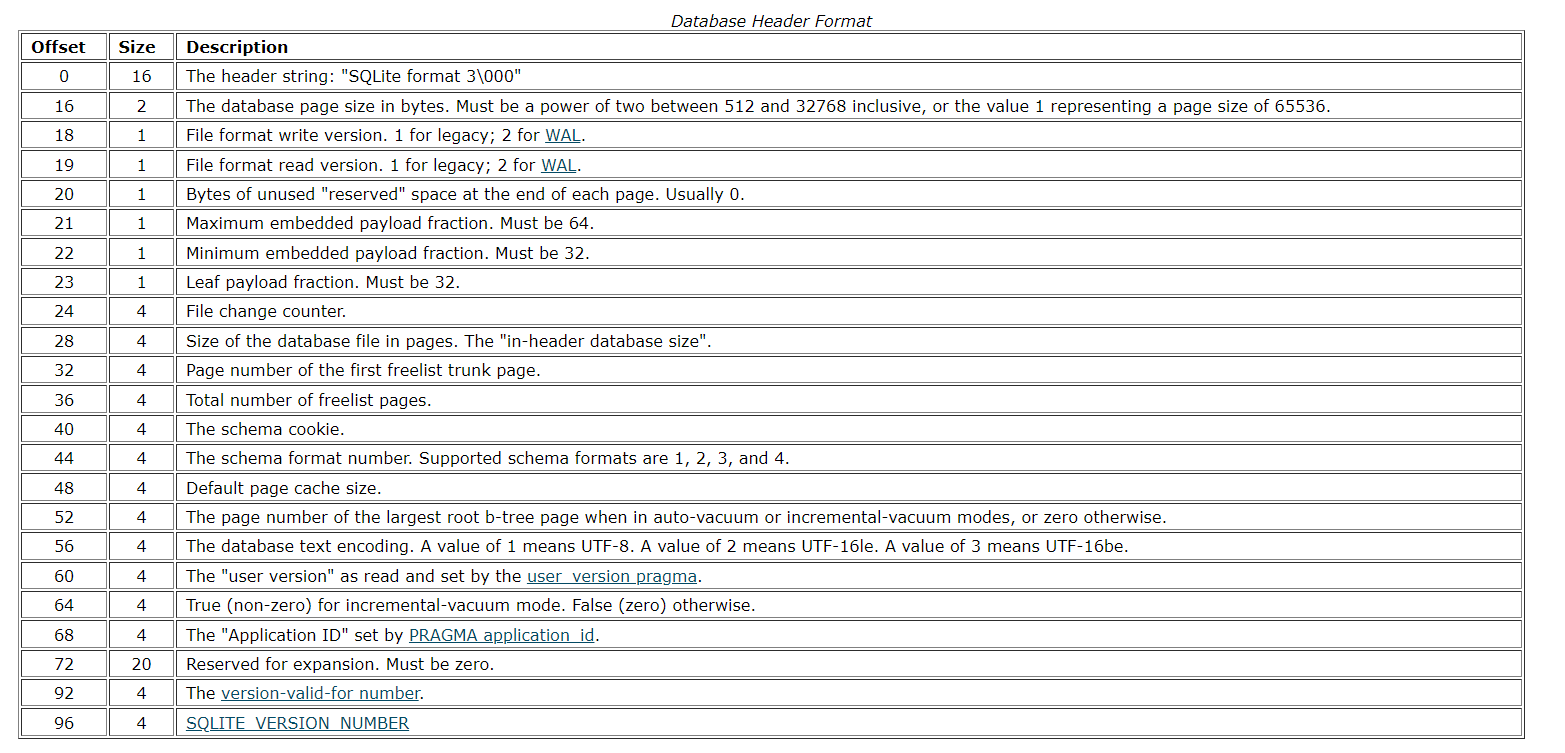
2.b.ii 來源: <https://sqlite.org/optoverview.html>

<https://sqlite.org/queryplanner.html>

Query planner就是要選出最有效率的方法來實作sql statement，他會考慮index、估計cost來找出方法。當接收到statement時，會先檢查他的結構，接著就是去database裡找table和columns，並且收集他們的數據，然後計算不同方法的cost，產出各種plans，找出最好的plan。

3.a 來源: <https://sqlite.org/fileformat2.html>

Database file是由一個或多個pages組成，每個page的大小是固定的，大小會從512到65536，總共可以有2^32-2頁，通常sqlite database會是kilobytes到gigabytes，每個page在main database裡只有一種用途分別是這些

Database file裡錢100個bytes是header，header被分為很多個部分，multibytes field是存在MSB。

3.b

在os.c 82行有sqlite3OsRead、sqlite3OsWrite、sqlite3OsClose，分別對應到read、write、close，在這三個function裡，都會呼叫sqlite3\_file的function，xRead、xWrite、xClose。

4.a 來源: <https://sqlite.org/lockingv3.html>

Locking: sqlite有5種locking state，database file可能是5種裡的其中1種。Unlock是指database file沒有鎖，不能讀也不能寫，所有的快取資料都是需要被驗證的狀態，是預設的狀態；Shared是可以讀但不能寫，可以同時很多人讀，啟動多個shared lock，但不論如何都不可寫入檔案；Reserved是指之後將要變成可以寫的狀態，但還仍然是只能讀不能寫，一次只能有一個reserved lock被啟動，但還是能有多個shard lock；Pending是指想要寫入檔案且越快越好，只要等到所有shared lock都清空，就可以取得exclusive lock，啟動pending lock時就不能再增加shared lock；Exclusive是指可以寫入檔案，一次只能有一個exclusive lock，並且不能有其他的lock存在。

4.b

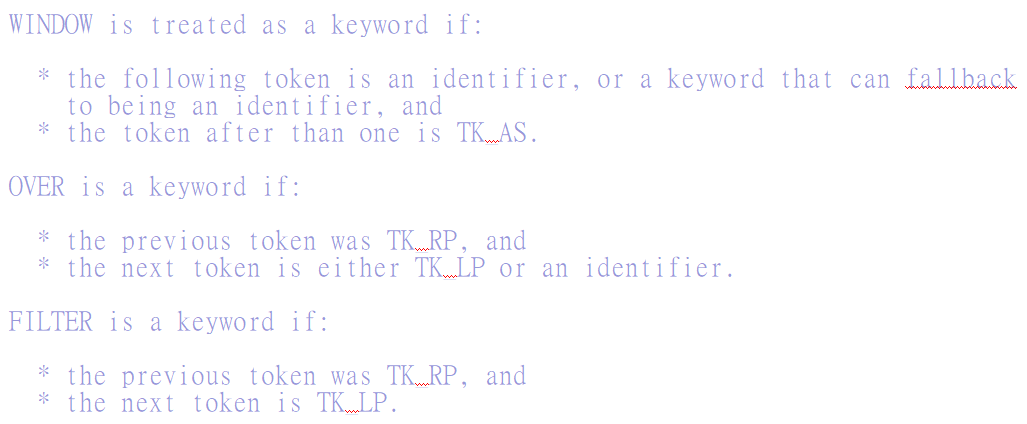
2-B:

Tokenizer: tokenizer在sqlite裡，會負責將輸入的字串分解一個個token，並且送到parser去判斷這行指令是什麼。

在tokenize.c 29行開始，是去定義每個符號的id，這些id會用在判斷這個token是甚麼。

第61行開始是他的table，利用對應table得知當前為哪個符號。

在第197行的getToken，是要得到token的id，也就是前面定義的。

第217行，當tokenizer讀到window、over、filter的時候，他要特別處理，因為他需要知道這個詞是keyword還是SQL identifier，因此這裡是去辨別這些詞是否為keyword，他們會分別在以下的情況為keyword。

在273行，sqlite3GetToken回傳的是這個token的size，並且還會將token的種類存在\*tokenType，裡面是用switch case實作，根據不同種類的token做出不同的事，而種類就是前面定義的那些符號。

最後在567行去執行parser，在tokenizer裡面去呼叫parser會比從parser裡面呼叫tokenizer還要快，並且比較安全。