### 数据库文件的结构

主数据库文件以.db为后缀。数据库文件内部划分为若干个页。页的大小可以是512到65536之间任意的2次方中的一个，但是同一个数据库中所有页的大小是相同的。一个数据库文件最多拥有2147483646(2^31-2)个页。

页的编号从1开始，所以数据库文件偏移为0处开始到页大小值前一字节处的页页号为1。紧跟在页面1后面的页面按递增顺序依次指定页号。

可以使用工具SQlite Page Explorer查看数据库文件里面的页信息，工具在github的地址为<https://github.com/rubydongle/sqlite3_page_explorer>。

页号为1的页和其它的页有一些区别，其前100个字节中存储着数据库文件头信息。即数据库文件头100个字节中存储着该数据库文件的文件头信息。

数据库文件头中包含下面信息：

|  |  |  |
| --- | --- | --- |
| 偏移(Offset) | 大小(Size) | 描述(Description) |
| 0 | 16 | 头字符串”SQLite format 3\000” The header string: “SQLite format 3\000” |
| 16 | 2 | 数据库页大小，单位byte.必须是512~32768之间2^xxx,如果是1表示页大小为65536 |
| 18 | 1 | File format write version. 1 for legacy; 2 for WAL. |
| 19 | 1 | File format read version. 1 for legacy; 2 for WAL. |
| 20 | 1 | Bytes of unused “reserved” space at the end of each page. Usually 0. |
| 21 | 1 | Maximum embedded payload fraction. Must be 64. |
| 22 | 1 | Minimum embedded payload fraction. Must be 32. |
| 23 | 1 | Leaf payload fraction. Must be 32. |
| 24 | 4 | File change counter. |
| 28 | 4 | Size of the database file in pages. The “in-header database size”. |
| 32 | 4 | Page number of the first freelist trunk page. |
| 36 | 4 | Total number of freelist pages. |
| 40 | 4 | The schema cookie. |
| 44 | 4 | The schema format number. Supported schema formats are 1, 2, 3, and 4. |
| 48 | 4 | Default page cache size. |
| 52 | 4 | The page number of the largest root b-tree page when in auto-vacuum or incremental-vacuum modes, or zero otherwise. |
| 56 | 4 | The database text encoding. A value of 1 means UTF-8. A value of 2 means UTF-16le. A value of 3 means UTF-16be. |
| 60 | 4 | The “user version” as read and set by the user\_version pragma. |
| 64 | 4 | True (non-zero) for incremental-vacuum mode. False (zero) otherwise. |
| 68 | 4 | 通过命令PRAGMA application\_id设置的”Application ID” |
| 72 | 20 | 保留用作后续扩展，必须是0. |
| 92 | 4 | The version-valid-for number. |
| 96 | 4 | SQLite 版本号 |

SQLite数据库文件中的页有下列类型：

* B-树页(A b-tree page)
* 有效负荷溢出页(A payload overflow page)
* 空闲列表页(A freelist page)
* (A pointer map page)
* (The lock-byte page)

### B-树页

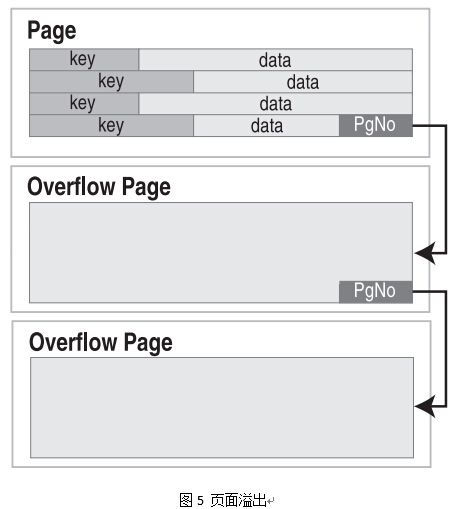
B-树页用来存储B-树信息，包括数据库表信息和数据库索引信息。B-树页分为下面四类

1. A table b-tree interior page
2. A table b-tree leaf page
3. An index b-tree interior page
4. An index b-tree leaf page

### 有效负荷溢出页(A payload overflow page)

有效载荷及其内容可有不同的大小。然而，页面大小是固定不变的。因此，给定的有效载荷总有可能超出单页装载大小。这种情况发生时，额外的有效载荷将添加到溢出页面的链接链表上。由此看来，有效载荷将在有序的链接链表中显示，如图所示。

图中第4个有效载荷超出了当前页所能装载的大小。因此，B-tree模块创建了溢出页来装载。实际上，一个溢出页也不能装载，因此，又链接了第二个溢出页。这实际上就是处理二进制大对象的方法。使用真正的大字段时，最后都采用页链接链表来存储。如果blob字段太大，这种方式效率很低，此时，可考虑创建外部文件来存储blob数据，并将外部文件名保存在记录中。



### 空闲列表页(Freelist Page)

空闲列表页有两种

1. A freelist trunk page
2. A freelist leaf page

A database file might contain one or more pages that are not in active use. Unused pages can come about, for example, when information is deleted from the database. Unused pages are stored on the freelist and are reused when additional pages are required.

The freelist is organized as a linked list of freelist trunk pages with each trunk page containing page numbers for zero or more freelist leaf pages.

A freelist trunk page consists of an array of 4-byte big-endian integers. The size of the array is as many integers as will fit in the usable space of a page. The minimum usable space is 480 bytes so the array will always be at least 120 entries in length. The first integer on a freelist trunk page is the page number of the next freelist trunk page in the list or zero if this is the last freelist trunk page. The second integer on a freelist trunk page is the number of leaf page pointers to follow. Call the second integer on a freelist trunk page L. If L is greater than zero then integers with array indexes between 2 and L+1 inclusive contain page numbers for freelist leaf pages.

Freelist leaf pages contain no information. SQLite avoids reading or writing freelist leaf pages in order to reduce disk I/O.

The number of freelist pages is stored as a 4-byte big-endian integer in the database header at an offset of 36 from the beginning of the file. The database header also stores the page number of the first freelist trunk page as a 4-byte big-endian integer at an offset of 32 from the beginning of the file.

### Pointer Map or Ptrmap Pages

Pointer map or ptrmap pages are extra pages inserted into the database to make the operation of auto\_vacuum and incremental\_vacuum modes more efficient. Other page types in the database typically have pointers from parent to child. For example, an interior b-tree page contains pointers to its child b-tree pages and an overflow chain has a pointer from earlier to later links in the chain. A ptrmap page contains linkage information going in the opposite direction, from child to parent.

### The Lock-Byte Page

The lock-byte page is the single page of the database file that contains the bytes at offsets between 1073741824 and 1073742335, inclusive. A database file that is less than or equal to 1073741824 bytes in size contains no lock-byte page. A database file larger than 1073741824 contains exactly one lock-byte page.

The lock-byte page is set aside for use by the operating-system specific VFS implementation in implementing the database file locking primitives. SQLite does not use the lock-byte page. The SQLite core will never read or write the lock-byte page, though operating-system specific VFS implementations may choose to read or write bytes on the lock-byte page according to the needs and proclivities of the underlying system. The unix and win32 VFS implementations that come built into SQLite do not write to the lock-byte page, but third-party VFS implementations for other operating systems might.

The lock-byte page arose from the need to support Win95 which was the predominant operating system when this file format was designed and which only supported mandatory file locking. All modern operating systems that we know of support advisory file locking, and so the lock-byte page is not really needed any more, but is retained for backwards compatibility.