Pointer Page - type 0x04

A pointer page is used internally to hold a list of all - or as may will fit on one pointer page - data pages (see below) that make up a single table. Large tables may have more than one pointer page but every table, system or user, will have a minimum of one pointer page. The RDB$PAGES table is where the Firebird engine looks to find out where a table is located within the physical database, however, RDB$PAGES is itself a table, and when the database is running, how exactly can it find the start page for RDB$PAGES in order to look it up?

The database header page contains the page number for RDB$PAGES at bytes 0x14 - 0x17 on the page. From experimentation, it appears as if this is always page 0x03, however, this cannot be relied upon and if you need to do this, you should always check the database header page to determine where RDB$PAGES is to be found.

The C code representation of a pointer page is:

```
struct pointer_page
{
   pag ppg_header;
   SLONG ppg_sequence;
   SLONG ppg_next;
   USHORT ppg_count;
   USHORT ppg_relation;
   USHORT ppg_min_space;
   USHORT ppg_max_space;
   SLONG ppg_page[1];
};
```

Ppg_header: A pointer page starts with a standard page header. In the header, the pag_flags field is used and is set to the value 1 if this is the final pointer page for the relation.

Ppg_sequence: Four bytes, signed. Offset 0x10 to 0x13 on the page. The sequence number of this pointer page in the list of pointer pages for the table. Starts at zero.

Ppg_next: Four bytes, signed. Offset 0x14 to 0x17 on the page. The page number of the next pointer page for this table. Zero indicates that this is the final pointer page.

Ppg_count: Two bytes, unsigned. Offset 0x18 and 0x19 on the page. This field holds the count of active slots (in the ppg_page[] array) on this pointer page, that are in use. As the array starts at zero, this is also the index of the first free slot on this pointer page.

Ppg_relation: Two bytes, unsigned. Offset 0x1a and 0x1b on the page. This field holds the RDB$RELATIONS.RDB$REALTION_ID for the table that this pointer page represents.

Ppg_min_space: Two bytes, unsigned. Offset 0x1c and 0x1d on the page. This indicates the first entry in the ppg_page array holding a page number which has free space in the page.

Ppg_max_space: Two bytes, unsigned. Offset 0x1e and 0x1f on the page. This was intended to indicate the last entry in the ppg_page array holding a page number which has free space in the page, but it has never been used. These two bytes are invariably set to zero.

Ppg_page: An array of four byte signed values, starting at offset 0x20. Each value in this array represents a page number where a part of the current table is to be found. A value of zero in a slot indicates that the slot is not in use. Deleting all the data from a table will result in all slots being set to zero.

Page fill bitmaps: At the end of each pointer page is a bitmap array of two bit entries which is indexed by the same index as the ppg_page array. These bitmaps indicate that the page is available for use in storing records (or record versions) or not. The two bits in the bitmap indicate whether a large object (BLOB?) is on this page and the other bit indicates that the page is full. If either bit is set (page has a large object or page is full, then the page is not used for new records or record versions.

The location of the bitmaps on each page is dependent on the page size. The bigger the page, the more slots in the ppg_page array it can hold and so the bitmap is bigger. A bigger bitmap starts at a lower address in the page and so on. From lookiing inside a few databases with a 4Kb page size, the bitmaps begin at offset 0x0f10 on the page.

back to top of page

You can find the pointer page for any table by running something like the following query in isql:

```
SQL> SELECT P.RDB$PAGE_NUMBER, P.RDB$PAGE_SEQUENCE, P.RDB$RELATION_ID
CON> FROM RDB$PAGES P
CON> JOIN RDB$RELATIONS R ON (R.RDB$RELATION_ID = P.RDB$RELATION_ID)
CON> WHERE R.RDB$RELATION_NAME = 'EMPLOYEE'
CON> AND P.RDB$PAGE_TYPE = 4;

RDB$PAGE_NUMBER RDB$PAGE_SEQUENCE RDB$RELATION_ID
=============== ================= ===============
            180                 0             131
```

The page number which has RDB$PAGE_SEQUENCE holding the value zero is the top level pointer page for this table. In the above example, there is only one pointer page for the EMPLOYEE table. If we now hexdump the pointer page for the EMPLOYEE table, we see the following:

```
000b4000  04 01 39 30 02 00 00 00   00 00 00 00 00 00 00 00   Standard
header
000b4010  00 00 00 00                                         Ppg_sequence
000b4014  00 00 00 00                                         Ppg_next
000b4018  02 00                                               Ppg_count
000b401a  83 00                                               Ppg_relation
000b401c  01 00                                               Ppg_min_space
000b401e  00 00                                               Ppg_max_space
000b4020  ca 00 00 00                                         Ppg_page[0]
000b4024  cb 00 00 00                                         Ppg_page[1]
000b4028  00 00 00 00                                         Ppg_page[2]
000b402c  00 00 00 00                                         Ppg_page[3]
...
000b4f10  01 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
|................|
000b4f20  00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

```
|. . . . . .|
```

Looking at the above, we can see at address 0x0b4f10 on the page, that the byte there has the value of 0x01. This is an indicator that the page in ppg_page[0] - page 0xca - is full to capacity (bit 0 set) and does not have any large objects on the page (bit 1 unset). The page at ppg_page[1] - page 0xcb - is, on the other hand, not full up yet (bit 2 is unset) and doesn't have a large object on the page either. This means that this page is avilable for us.

This is confirmed by checking the value in ppg_min_space which has the value 0x0001 and does indeed correspond to the first page with free space. The value in ppg_min_space is the index into the ppg_array and not the page number itself.